YAYASAN PRIMA AGUS TEKNIK

Manajemen Proyek Scrum

Pada Pengembangan Perangkat Lunak, Produk dan Layanan

Dr. Agus Wibowo, M.Kom, M.Si, MM.

Manajemen Proyek Scrum

Pada Pengembangan Perangkat Lunak, Produk dan Layanan

Dr. Agus Wibowo, M.Kom, M.Si, MM.

BIO DATA PENULIS

Penulis memiliki berbagai disiplin ilmu yang diperoleh dari Universitas Diponegoro (UNDIP) Semarang. dan dari Universitas Kristen Satya Wacana (UKSW) Salatiga. Disiplin ilmu itu antara lain teknik elektro, komputer, manajemen dan ilmu sosiologi. Penulis memiliki pengalaman kerja pada industri elektronik dan sertifikasi keahlian dalam bidang Jaringan Internet, Telekomunikasi, Artificial Intelligence, Internet Of Things (IoT), Augmented Reality (AR), Technopreneurship, Internet Marketing dan bidang pengolahan dan analisa data (komputer statistik).

Penulis adalah pendiri dari Universitas Sains dan Teknologi Komputer (Universitas STEKOM) dan juga seorang dosen yang memiliki Jabatan Fungsional Akademik Lektor Kepala (Associate Professor) yang telah menghasilkan puluhan Buku Ajar ber ISBN, HAKI dari beberapa karya cipta dan Hak Paten pada produk IPTEK. Sejak tahun 2023 penulis tercatat sebagai Dosen luar biasa di Fakultas Ekonomi & Bisnis (FEB) Universitas Diponegoro Semarang. Penulis juga terlibat dalam berbagai organisasi profesi dan industri yang terkait dengan dunia usaha dan industri, khususnya dalam pengembangan sumber daya manusia yang unggul untuk memenuhi kebutuhan dunia kerja secara nyata.

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Manajemen Proyek Scrum Pada Pengembangan Perangkat Lunak, Produk dan Layanan

Penulis :
Dr. Agus Wibowo, M.Kom, M.Si, MM.

ISBN :

Editor :
Dr. Joseph Teguh Santoso, S.Kom., M.Kom.

Penyunting :
Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

Desain Sampul dan Tata Letak :
Irdha Yunianto, S.Ds., M.Kom

Penebit :
Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Anggota IKAPI No: 279 / ALB / JTE / 2023

Redaksi :
Jl. Majapahit no 605 Semarang
Telp. 08122925000
Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :
Universitas STEKOM
Jl. Majapahit no 605 Semarang
Telp. 08122925000
Fax. 024-6710144
Email : info@stekom.ac.id

KATA PENGANTAR

Dalam era digital yang terus berkembang, pengembangan perangkat lunak, produk, dan layanan telah menjadi pusat perhatian bagi banyak organisasi yang berusaha untuk tetap kompetitif dan relevan di pasar. Salah satu metode yang terbukti efektif dalam mengelola proyek-proyek ini adalah *Scrum*, sebuah kerangka kerja manajemen proyek berbasis Agile yang memberikan pendekatan iteratif dan kolaboratif untuk mencapai hasil yang optimal. Scrum memfasilitasi tim dalam menciptakan produk yang bernilai tinggi dengan cara yang adaptif terhadap perubahan dan selalu berfokus pada kebutuhan pelanggan.

Manajemen proyek menggunakan Scrum dalam pengembangan perangkat lunak, produk, dan layanan tidak hanya memungkinkan tim untuk bekerja lebih efisien, tetapi juga mendorong komunikasi yang lebih baik, transparansi yang lebih tinggi, dan kolaborasi yang lebih erat antara anggota tim dan pemangku kepentingan. Metode ini membangun dasar yang kuat untuk mencapai tujuan bersama, memprioritaskan pekerjaan yang paling bernilai, serta merespons perubahan dengan cepat dan tepat.

Buku ini membahas secara mendalam tentang prinsip dan praktik Scrum serta penerapan Agile dalam konteks pengembangan perangkat lunak. Bab pertama mengawali dengan pembahasan mengenai kelincahan di era Revolusi Industri Keempat, perbedaan dengan metodologi Waterfall, serta pengenalan Agile Manifesto yang menekankan pentingnya perangkat lunak yang berfungsi lebih daripada dokumentasi. Selain itu, dibahas pula 12 prinsip Agile yang menjadi dasar untuk pengembangan yang adaptif dan kolaboratif. Bab kedua mengulas kerangka kerja Scrum, termasuk penjelasan mengenai peran, visi, backlog, serta pentingnya Sprint dalam siklus pengembangan. Disini juga dijelaskan bagaimana menetapkan Definisi Selesai dan bagaimana Scrum mendukung iterasi dalam pengembangan responsif.

Bab ketiga fokus pada peran-peran dalam Scrum, mulai dari tiga pilar dasar Scrum hingga peran setiap individu seperti Scrum Master, Product Owner, dan pengembang dalam tim Agile. Terdapat juga penjelasan mengenai bagaimana Scrum Master dapat melatih dan memfasilitasi tim untuk bekerja lebih efektif. Bab keempat menggali lebih dalam tentang struktur tim Scrum, baik tim yang terkoordinasi maupun yang tersebar, serta konsep Scrum-of-Scrums untuk tim besar. Bab kelima membahas berbagai upacara dan artefak dalam Scrum, seperti Definisi Selesai, Bagan Burn-Down, retrospektif Sprint, dan penyempurnaan backlog yang menjadi kunci pengelolaan proyek yang baik.

Dalam bab keenam, penekanan diberikan pada tanggung jawab Scrum Master dalam membangun tim yang efektif, memfasilitasi perencanaan dan pelaksanaan Sprint, serta mengembangkan kriteria penerimaan yang jelas untuk cerita pengguna. Bab ketujuh mengulas interaksi antara Scrum Master dengan peran lainnya, seperti Product Owner, tim pengembang, serta pemangku kepentingan, yang penting untuk menciptakan kerjasama yang solid dalam tim. Bab kedelapan membahas tentang keterampilan lunak yang harus dimiliki Scrum Master, termasuk kemampuan untuk melatih tim, mengelola perilaku emosional, serta

melindungi tim dari gangguan eksternal. Sementara bab kesembilan menyoroti keterampilan teknis Scrum Master yang meliputi refactoring, pengembangan berbasis pengujian, serta menyempurnakan estimasi dan perencanaan Sprint.

Bab kesepuluh menyentuh keterampilan kontinjensi Scrum Master dalam menghadapi hambatan, scope creep, serta menjalankan tinjauan Sprint untuk memastikan keberhasilan proyek. Akhirnya, bab kesebelas menyatukan seluruh konsep yang telah dibahas, menekankan pentingnya keberhasilan tim dalam Scrum, peran pelatih untuk mendukung Scrum Master, serta penerapan DoD (Definisi Selesai) dan DoR (Definisi Siap) untuk memperjelas ekspektasi dan kriteria dalam setiap Sprint.

Semoga buku ini dapat menjadi panduan yang bermanfaat dan dapat diimplementasikan dengan sukses dalam manajemen proyek di berbagai bidang, serta memberikan dampak positif dalam proses pengembangan perangkat lunak, produk, dan layanan yang lebih baik. Terima Kasih.

Semarang, Januari 2025
Penulis

Dr. Agus Wibowo, M.Kom, M.Si, MM.

DAFTAR ISI

BAB 1
PRINSIP AGILE

1.1 KELINCAHAN DI ERA REVOLUSI INDUSTRI KEEMPAT

Selamat datang di era Revolusi Industri Keempat. Ketika saya masih di sekolah dasar, saya membaca buku yang memprediksi bahwa pada abad ke-21, hal-hal luar biasa seperti mobil terbang, pakaian berbahan aluminium foil, dan liburan ke Mars akan menjadi hal yang biasa. Meskipun prediksi tersebut belum sepenuhnya terwujud, banyak kemajuan teknologi yang luar biasa telah terjadi. Misalnya, mobil sekarang dapat parkir paralel secara otomatis, ponsel memiliki daya komputasi yang lebih besar daripada komputer besar pada masa lalu, dan printer 3D digunakan untuk membuat prostetik serta anggota tubuh robotik. Semua kemajuan ini didorong oleh perangkat lunak, yang kini menyentuh hampir setiap aspek kehidupan kita. Perangkat lunak tidak hanya dikembangkan oleh perusahaan teknologi, tetapi juga oleh sektor-sektor lain seperti bank, asuransi, dan bahkan industri otomotif, menjadikannya bagian integral dari banyak aspek kehidupan modern.

Revolusi Pertanian mengubah pola hidup manusia dengan memulai pertanian dan pemeliharaan hewan, memungkinkan manusia untuk menetap di satu tempat. Lalu, Revolusi Industri membuat masyarakat berpindah dari pertanian ke pabrik-pabrik, yang memunculkan kota-kota besar dan pekerjaan terstruktur. Kemudian, Revolusi Informasi menandai lahirnya pekerja yang berpengetahuan, yang bisa saja merupakan pengembang perangkat lunak, dokter, ilmuwan, atau guru. Pekerjaan jenis ini lebih berfokus pada perubahan dan inovasi yang berkelanjutan, yang berbeda dari pekerjaan sebelumnya yang lebih terstruktur.

Kini, kita berada dalam Revolusi Industri Keempat yang dipicu oleh teknologi disruptif yang tidak hanya mengubah cara kita bekerja dan hidup, tetapi juga cara kita berinteraksi. Keberhasilan kita dalam era ini sangat bergantung pada kemampuan untuk berkolaborasi dan beradaptasi dengan perubahan yang cepat. Teknologi sangat mempengaruhi masyarakat dan dunia bisnis, serta mengaburkan batas antara dunia fisik dan digital, yang membutuhkan ketangkasan untuk bertahan dan berkembang.

Saat saya menyebutkan kata "tangkas", kebanyakan orang mungkin berpikir tentang cheetah, yang merupakan hewan tercepat di darat. Namun, meskipun cheetah unggul dalam kecepatan, antelop Impala memiliki kelincahan yang luar biasa. Dengan kemampuannya untuk mengubah arah dengan cepat, impala dapat menghindari kejaran cheetah dan memaksanya untuk berhenti mengejar. Di dunia hewan, kelincahan menjadi kunci untuk bertahan hidup, dan hal ini juga berlaku di dunia kita yang terus berubah. Seperti impala yang tangkas, kita harus bisa beradaptasi dengan cepat untuk bertahan hidup dalam dunia yang penuh dengan inovasi teknologi ini.

Kelincahan adalah tentang Bergerak dan Beradaptasi dengan Cepat

Pada tahun pertama saya kuliah, saya rasa tahun 1981, pergelangan kaki saya retak pada hari pertama latihan dengan bantalan saat permainan pertama yang saya coba jalankan sebagai quarterback. Seperti yang saya pelajari, kelemahan pertahanan empat-empat adalah

di bagian tengah atau di sekitar ujung. Saya membawa bola di sekitar sisi kanan pertahanan untuk mendapatkan jarak 30 yard. Saya fokus pada udara yang bersiul melalui helm saya. Berlari seperti angin... sampai saya menginjak lubang gopher. Saat pertahanan mengejar saya, saya mendengar pelatih kepala berteriak, "Jika itu quarterback yang bagus, dia pasti akan mencetak gol!"

Pergelangan kaki saya butuh waktu lama untuk pulih, jadi saya tidak bermain selama tahun kedua saya. Selain itu, saya tidak bisa disebut anggun. Saya akan mengatakan bahwa saya jangkung, tidak terbiasa dengan tubuh saya yang setinggi enam kaki. Paman saya Frank, yang bermain untuk Universitas Nebraska dan berusaha membantu saya dengan karier sepak bola sekolah menengah saya yang malang, punya istilah lain yang ia gunakan untuk menggambarkan pola gerakan saya: Kaki Berat.

“Apa yang Anda lihat dilakukan petinju sepanjang waktu?" tanya paman saya. “Lompat tali,” jawab saya. “Benar sekali!” katanya. “Setiap ada kesempatan, mereka melompat tali. Itu karena gerak kaki penting bagi petinju. Sama halnya dengan sepak bola. Anda perlu melakukan banyak latihan kelincahan seperti lompat tali karena Anda tidak bisa menggerakkan kaki.” Saya lompat tali, angkat beban, dan lari, tetapi sepak bola tidak pernah berhasil bagi saya. Saya merasa agak ironis bahwa orang yang tidak bisa keluar dari jalannya sendiri sekarang melatih Agility. Tidak, bukan jenis ketangkasan itu. Jika Anda menantang saya untuk menjelaskan kerangka kerja Agile dalam satu kalimat, saya akan mengatakan ini: Agile adalah tentang beradaptasi dengan perubahan, bukan merencanakan semuanya di awal.

Agile adalah banyak hal. Melakukan iterasi lebih sering, membebaskan pengembangan untuk melakukan apa yang paling mereka kuasai, menyelaraskan bisnis dengan perangkat lunak, mencoba menghasilkan barang yang benar-benar ingin dibeli pelanggan kami, dan mengembangkan budaya perbaikan berkelanjutan. Semua hal hebat yang merupakan bagian dari Agile. Namun, bagi saya itu semua tentang menanggapi perubahan. Pada hari-hari awal saya di tim pengembangan, perubahan merupakan kekuatan yang mengganggu dan disambut seperti rakun yang mengamuk di garasi Anda.

Karena semua perencanaan dilakukan di awal proyek, semua jenis perubahan akan menimbulkan masalah. Sebuah tim harus menghindari perubahan dengan cara apa pun. Namun, pada kenyataannya perubahan yang kami hindari adalah pelanggan yang membutuhkan bantuan. Itu adalah cacat pada desain yang perlu ditangani, itu adalah memanfaatkan teknologi baru untuk membuat produk menjadi lebih baik. Persyaratan berubah begitu cepat sehingga bodoh untuk mencoba merencanakan rilis skala besar di awal. Tim harus menerima perubahan. Mereka harus mengubah cara mereka bekerja. Perubahan akan terjadi. Tim yang tangkas harus mendambakan perubahan. Berpegang pada rencana tidak akan berhasil lagi. Kekuatannya ada pada kemampuan untuk berputar dan beradaptasi dengan cepat. Begitulah cara kami menciptakan pelanggan yang senang.

1.2 MENGENAL METODOLOGI WATERFALL

Ketika saya mulai bekerja untuk vendor perangkat lunak, saya tidak tahu bahwa kami menggunakan metodologi Waterfall; saya hanya berpikir begitulah cara kami melakukan

sesuatu. Metodologi Waterfall memiliki tujuan untuk setiap fase pengembangan. Setelah fase pengembangan selesai, Anda beralih ke fase berikutnya. Anda tidak dapat kembali setelah melewati satu fase. Karena kita berbicara tentang air terjun, mari kita lihat air terjun yang mungkin paling terkenal di dunia, Air Terjun Niagara. Saat Anda pergi ke sana, keindahan dan keagungan tempat itu akan memberikan kesan. Volume air yang mengalir melalui air terjun setiap hari sungguh menakjubkan. Satu hal yang jelas adalah bahwa setelah air melewati tebing, Anda tidak dapat membawanya kembali ke atas gunung.

Sama halnya dengan pengembangan Waterfall. Seperti yang Anda lihat pada Gambar 1.1, Anda tidak dapat kembali.

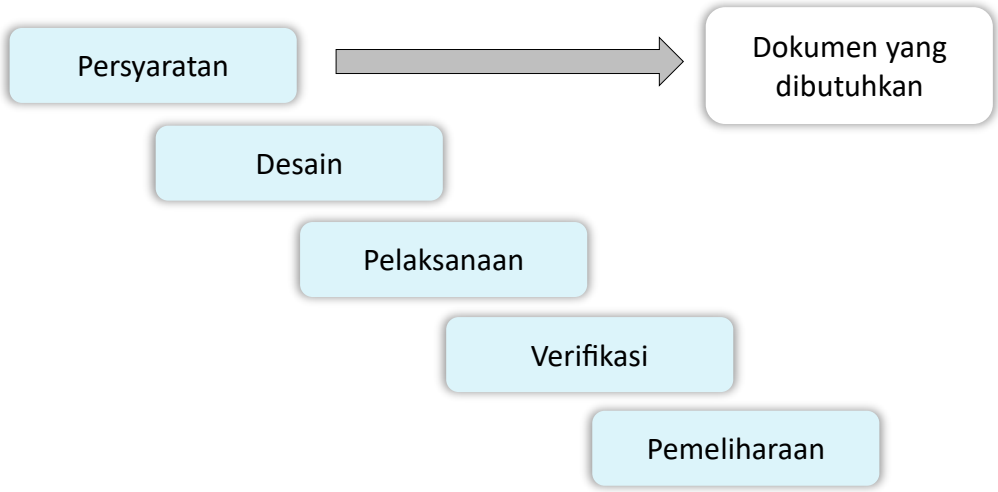
Gambar 1.1 Setelah Air Melewati Air Terjun, Tidak Ada Jalan Kembali

Saat Anda memikirkan cara membangun perangkat lunak, model Waterfall menggambarkan metode yang linier, berpagar, dan berurutan. Anda mengambil proyek pengembangan dan memecahnya menjadi beberapa tahap atau fase pengembangan. Setiap fase ini memiliki tujuan. Anda beralih ke fase berikutnya setelah mencapai tujuan. Saya pernah melihat Waterfall digambarkan menyerupai lomba lari estafet. Seorang pelari mengambil tongkat, berlari di bagian lomba, dan menyerahkan tongkat kepada pelari berikutnya. Gambar 1.2 menunjukkan fase-fase metode Waterfall.

Gambar 1.2 Fase Waterfall

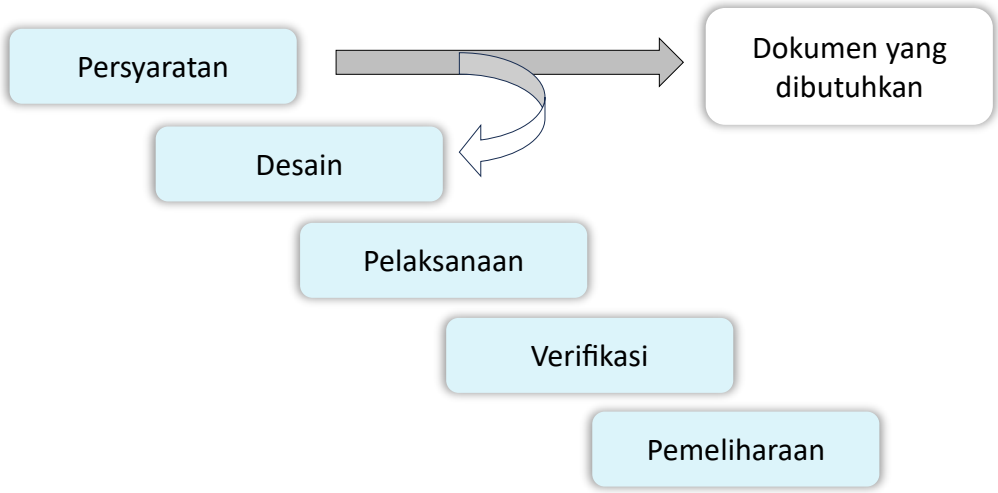
Fase Persyaratan

Sebuah tim akan memulai dengan fase persyaratan, seperti yang ditunjukkan pada Gambar 1.3. Di sinilah analis bisnis atau pemasaran produk mendefinisikan persyaratan.

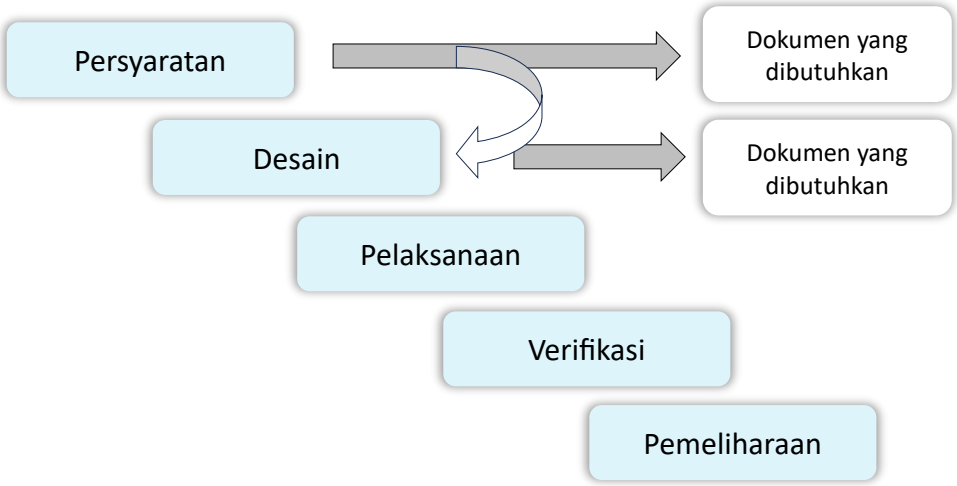


Gambar 1.3 Fase Persyaratan Waterfall

Dengan kata lain, mereka akan mencari tahu apa yang akan kita bangun, dan apa yang dibutuhkan pelanggan dan bisnis dari proyek ini. Sasaran dari fase persyaratan biasanya berupa semacam dokumen persyaratan yang merinci temuan tim. Seperti yang ditunjukkan pada Gambar 1.4, tim melakukan semua perencanaan dalam fase desain. Pekerjaan desain terdiri dari hal-hal seperti memutuskan bagaimana tim akan menulis kode. Mereka akan mencari tahu bahasa dan teknik yang akan digunakan, untuk memberikan fungsionalitas dan nilai terbaik, dan bahkan siapa yang akan mengerjakan apa. Hasil dari semua pekerjaan ini adalah staf pengembangan yang menghasilkan dokumen persyaratan internal, seperti yang ditunjukkan pada Gambar 1.5.



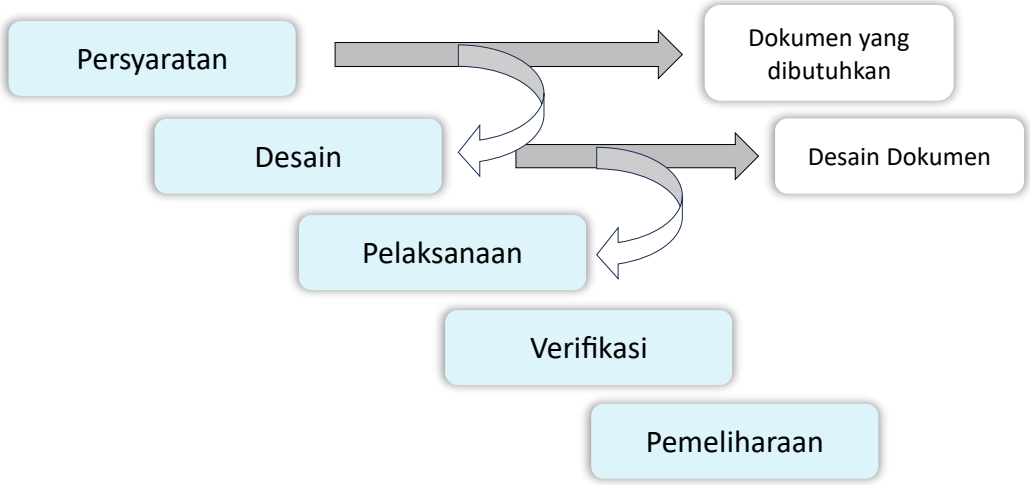
Gambar 1.4 Fase Desain Waterfall



Gambar 1.5 Dokumen Persyaratan Waterfall

Tahap Implementasi

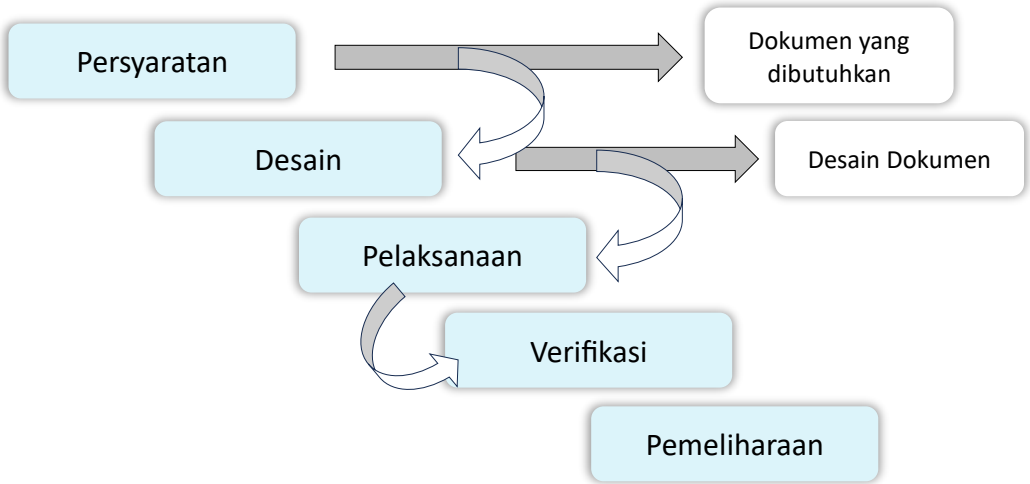
Tahap implementasi, seperti yang ditunjukkan pada Gambar 1-6, adalah saat pengembang melakukan hal terbaik yang dapat mereka lakukan. Mereka menulis kode dan mengubah ide menjadi produk fungsional. Pengembang mengambil persyaratan sebagaimana yang ditetapkan dalam tahap Persyaratan dan menciptakan nilai bagi pelanggan. Di akhir tahap implementasi, perangkat lunak diproduksi, dan perangkat lunak tersebut menyerupai produk yang layak.



Gambar 1.6 Tahap Implementasi Waterfall

Tahap Verifikasi

Kita memiliki perangkat lunak pada titik ini, dan itu adalah hal yang baik. Namun, perangkat lunak harus berfungsi dengan baik. Gambar 1-7 menunjukkan tahap Verifikasi, saat teknisi Jaminan Kualitas menguji perangkat lunak. Mereka bekerja untuk memastikan bahwa tim memberikan produk berkualitas tinggi kepada pelanggan. Teknisi QA mengambil dokumen desain yang diproduksi selama tahap desain dan membuat rencana pengujian dari dokumen tersebut. Mereka menjalankan rencana pengujian terhadap perangkat lunak yang diproduksi dalam tahap Implementasi. Staf pengembangan menangani setiap cacat yang ditemukan. Teknisi jaminan kualitas bekerja sebagai tim terpisah. Mereka tidak dianggap sebagai bagian dari pengembangan.



Gambar 1.7 Fase Verifikasi Waterfall

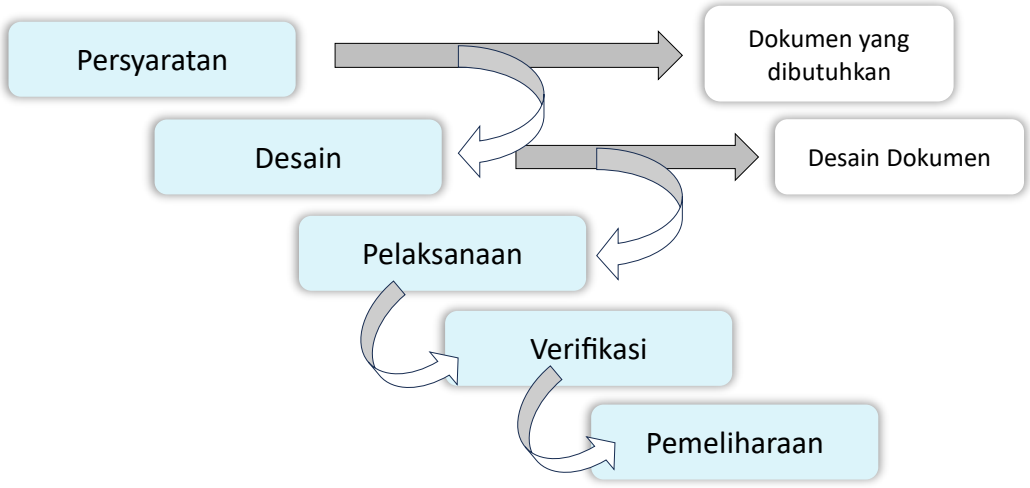
Setelah perangkat lunak diuji secara menyeluruh dan semua pihak setuju bahwa perangkat lunak berada dalam kondisi stabil. Rilis memasuki periode beta. Ini dapat dianggap sebagai fase Waterfall; namun, saya melihatnya sebagai bagian dari fase verifikasi. Selama pengujian Beta, rilis dikemas dan diberikan kepada sekelompok pelanggan tertentu untuk dicoba di

lingkungan mereka.

Fase Pemeliharaan

Setelah semua pihak setuju bahwa produk siap, produk tersebut dirilis, atau dianggap Tersedia Secara Umum (Generally Available/GA). Setelah produk "diluncurkan", rilis memasuki fase Pemeliharaan, seperti yang ditunjukkan pada Gambar 1.8. Teknisi atau pengembang yang melakukan pemeliharaan menangani setiap cacat yang ditemukan oleh pelanggan, dan pelaku bisnis memutuskan apakah rilis produk lainnya diperlukan. Masuk akal, bukan? Sekilas, Waterfall tampak seperti cara yang tepat untuk menghasilkan perangkat lunak berkualitas. Sejujurnya, saya terlibat dalam pembuatan beberapa produk terbaik menggunakan Waterfall.

Namun, ada beberapa kelemahan utama Waterfall yang sangat bermasalah di ekonomi berbasis aplikasi abad ke-21 ini. Jika persyaratan berubah dalam proyek Waterfall setelah fase persyaratan, semuanya harus dihentikan dan persyaratan baru perlu dievaluasi. Hal ini menyebabkan efek domino di mana semua fase lainnya "terdorong keluar." Hal ini menambah biaya proyek, dan membuat tanggal GA jauh lebih sulit dicapai. Seperti yang mungkin dapat Anda bayangkan, manajemen tidak akan senang dengan perubahan peristiwa ini. Kita sekarang hidup dan bekerja di dunia di mana persyaratan berubah dengan cepat, jadi ada banyak peluang bagi persyaratan untuk berubah cukup banyak selama rilis dua belas atau delapan belas bulan.



Gambar 1.8 Fase Pemeliharaan Waterfall

Anda mungkin bertanya pada diri sendiri, "Mengapa Waterfall membutuhkan waktu begitu lama? Dua belas hingga delapan belas bulan adalah waktu yang lama." Salah satu alasan terbesar yang dapat saya pikirkan adalah bahwa pelanggan terbiasa dengan irama itu. Mereka akan mencoba memasukkan sebanyak mungkin persyaratan ke dalam rilis berikutnya karena butuh waktu lama untuk merilisnya. Itu adalah tumpukan besar persyaratan yang mungkin relevan atau tidak saat fungsionalitas dikirimkan. Yang juga bermasalah adalah bahwa orang-orang Jaminan Kualitas tidak mendapatkan apa pun hingga kodenya selesai.

Begini, saya suka pengembang. Saya benar-benar menyukainya, tetapi semua orang perlu memahami bahwa setiap baris kode baru berpotensi cacat. Saya tidak menuduh siapa

pun menulis kode yang buruk. Meski begitu, kita semua manusia. Kesalahan pasti terjadi. Ditambah dengan fakta bahwa tim bekerja pada produk yang sangat rumit, dan Anda akan menemukan beberapa bug. Semakin awal Anda menemukan bug dalam proses pengembangan, semakin mudah untuk memperbaikinya. Biaya perbaikannya juga jauh lebih murah. Di Waterfall, bug tidak muncul hingga akhir proyek, dalam fase verifikasi.

Pikirkan itu sejenak. Bagaimana Anda memprediksi berapa banyak cacat yang akan ditemukan? Bagaimana Anda merencanakannya? Menurut pengalaman saya, fase verifikasi selalu mengancam untuk menunda tanggal GA karena jumlah waktu yang dibutuhkan untuk memperbaiki semua cacat. Hal ini mengakibatkan upaya keras yang penuh tekanan untuk mencapai tanggal GA. Teknisi dan pengembang jaminan kualitas juga merupakan dua tim yang terpisah. Hal ini menyebabkan sikap "kami versus mereka" terutama saat tingkat stres sedang tinggi.

Yang terpenting, pelanggan dan pemangku kepentingan tidak dapat melihat produk baru hingga Beta atau GA. Pelanggan akan menyampaikan persyaratan mereka, dan setahun kemudian, kami akan menghasilkan sesuatu yang baru dan mengilap. Namun, bagaimana kami tahu bahwa kami telah membuat sesuatu yang baru dan mengilap yang benar? Izinkan saya memberi Anda sebuah contoh. Saya bermain gitar bass di gereja saya. Saya sama sekali bukan seorang musisi, tetapi saya dapat bermain dengan kunci yang benar, sebagian besar waktu. Gambar 1.9 menunjukkan instrumen yang saya mainkan adalah rig empat senar Peavey Foundation lama. Alat musik itu tetap selaras, dan pas untuk saya.

Gambar 1.9 Gitar Bass Saya Tidak Mencolok; Namun, Saya Dapat Memindahkan Furnitur Ketika Saya Menekan Senar E Terbuka

Saya mengetahui bahwa putri saya Bailey akan membelikan saya gitar bass baru untuk Natal, dan saya menghentikannya. Saya tersentuh oleh kenyataan bahwa dia begitu memikirkan hadiah untuk saya, tetapi tidak mungkin dia bisa membelikan saya instrumen yang akan

membuat saya senang. Saya sangat teliti tentang cara bermain bass saya. Aksi, nuansa fretboard, bagaimana pickup ditempatkan pada bodi. Tidak mungkin saya tahu apakah saya akan menyukai gitar bass tertentu sampai saya memainkannya. Hal yang sama terjadi pada pelanggan kami. Tim mencoba membangun sesuatu yang memuaskan apa yang diminta, tetapi lebih sering daripada tidak, ini tidak terjadi.

Tentu, kami memiliki periode Beta, tetapi itu untuk memastikan perangkat lunak berfungsi, bukan untuk mengubah fungsionalitas. Pengembangan Waterfall dipengaruhi oleh perubahan persyaratan, tenggat waktu yang tidak realistis, dan estimasi yang buruk. Dampak terbesarnya adalah pada orang-orang. Perhatikan bahwa saya mengatakan orang, bukan sumber daya. Sumber daya adalah sesuatu yang Anda gunakan, seperti tisu toilet. Orang-orang luar biasa dan berbakat. Waterfall menyebabkan moral rendah, kelelahan, dan bahkan masalah kesehatan. Sesuatu harus dilakukan..

1.3 PENGENALAN AGILE MANIFESTO DAN PENGARUHNYA

Secara pribadi, saya tidak terlalu nyaman dengan kata manifesto. Karena saya adalah anak dari Perang Dingin, setiap kali seseorang mulai melontarkan kata manifesto, saya mulai berpikir tentang surga pekerja yang hebat yaitu bekas Uni Soviet atau Gulag. Karl Marx dan Friedrich Engels menulis Manifesto Komunis untuk menyatakan pemikiran mereka tentang seperti apa masyarakat yang sempurna itu. Itu adalah dokumen yang mengubah dunia, tetapi itu bukan satu-satunya. Menurut saya, manifesto adalah proklamasi cita-cita dan niat. Deklarasi Kemerdekaan adalah manifesto. Pidato Presiden Kennedy tentang Mendarat Manusia di Bulan adalah contoh lainnya.

Ketika saya remaja, saya tidak menginginkan apa pun selain menjadi monster. Tidak, tidak seperti Godzilla atau Frankenstein. Saya adalah anak yang diganggu. Menurut saya, otot dan kekuatan adalah perisai saya, dan saya sangat menginginkan keduanya. Dalam pikiranku, seseorang yang dapat merobek pintu dari engselnya dan kemudian harus memutarnya ke samping agar dapat melewati pintu tersebut secara otomatis dihormati oleh teman-temannya dan ditakuti oleh para pengganggu. Ada sebuah rahasia jalan menuju Nirvana yang berotot ini, dan aku akan menemukannya.

Perlu diingat, saat itu internet hanya ada dalam imajinasi Al Gore, dan informasi pelatihan tidak dapat ditemukan di mana pun. Akibatnya, teman-temanku dan aku membuang-buang waktu yang tak terhitung jumlahnya untuk mencoba-coba peralatan angkat beban plastik Sears dan Roebuck kami sambil menghabiskan uang hasil jerih payah kami untuk membeli suplemen yang tidak berharga. Lalu, itu terjadi. Saya tidak tahu apakah itu campur tangan ilahi, kebetulan, atau sekadar keberuntungan, tetapi paman seorang teman melindungi kami dan menunjukkan rahasianya. Kami mengambil apa yang diberikannya dan membuat semacam manifesto. Kami menuliskannya dan menempelkannya di dinding garasinya. Tepat di depan rak squat.

- Angkat beban berat. Tantang diri Anda untuk membuat rekor pribadi dalam setiap latihan.
- Jangan pernah melewatkan satu pengulangan pun. Kami diperkenalkan dengan

sesuatu yang disebut latihan istirahat/jeda, yang akan saya jelaskan nanti.

- Anda tumbuh saat beristirahat. Berolahraga selama dua jam sehari, setiap hari akan membuat Anda kecil dan lemah.
- Makan makanan asli. Banyak makanan asli. Tidak ada suplemen atau obat di dunia yang dapat menggantikan makanan asli.

Seperti yang saya katakan sebelumnya, manifesto adalah deklarasi publik. Itu adalah pernyataan tentang bagaimana kami bermaksud untuk melaksanakan apa yang kami lakukan mulai saat ini dan seterusnya. Itu mungkin tidak mengubah dunia, tetapi itu mengubah dunia saya. Pada tahun 2001, sekelompok orang yang sangat pintar berkumpul di sebuah resor ski. Sejauh yang saya pahami, kebanyakan dari mereka adalah pakar lean, atau pemrograman ekstrem (XP) yang berkumpul untuk membicarakan metodologi yang ringan. Mereka memunculkan apa yang sekarang kita kenal sebagai Manifesto Agile.

Individu dan Interaksi atas Proses dan Alat

Seperti lelucon lama jika Anda menempatkan sekelompok monyet di sebuah ruangan dengan mesin ketik, pada akhirnya dengan keberuntungan Anda akan mendapatkan karya Shakespeare yang lengkap. Konon, ini adalah kemungkinan matematis. Saya meragukannya. Secara matematis mungkin atau tidak. Ide yang sama di sini. Proses yang sangat terdefinisi dengan baik, alat-alat terbaik, fasilitas terbaik untuk bekerja, makan siang yang disediakan setiap hari, meja pingpong tidak ada yang penting jika tim tidak dapat bekerja sama.

Namun, jika salah satu dari rantai manajemen saya membaca ini, makan siang yang disediakan adalah ide yang bagus. Hanya sekadar memberi tahu. Cara yang lebih baik adalah dengan menempatkan tim dalam satu ruangan dan membiarkan mereka mencari cara terbaik untuk menyelesaikan pekerjaan. Idenya adalah bahwa tim perlu berbicara satu sama lain. Ketika saya mengatakan berbicara satu sama lain, yang saya maksud adalah semua orang berbicara satu sama lain.

Saya merasa senang bekerja di bisnis perangkat lunak selama lebih dari 20 tahun. Itu berarti saya telah mengikuti banyak rapat. Dalam rapat-rapat tersebut, saya mendengar banyak pembicaraan dari pengembang senior, arsitek, manajer pengembangan, direktur, dan sebagainya. Tidak banyak masukan dari staf junior, tim QA, atau teknisi Pengalaman Pengguna. Dugaan saya adalah mereka takut untuk berbicara di luar kesempatan karena mereka merasa akan diberhentikan atau dianggap kurang mampu.

Jauh lebih sehat untuk terlibat dalam rapat tim yang kolaboratif. Idenya adalah untuk membuat semua orang dalam tim berbicara satu sama lain, dan untuk memahami perspektif yang mereka miliki. Misalnya, saya selalu mengatakan bahwa saya dapat mengatakan hal terbodoh yang pernah Anda dengar dalam sebuah rapat, tetapi tetap memberikan masukan yang berharga. Artinya, apa yang saya katakan tidak begitu berharga, tetapi itu membuat Anda melihat dari mana saya berasal. Dengan kata lain, Anda memahami perspektif saya, dan memiliki gambaran yang cukup baik tentang seperti apa proses berpikir saya.

Interaksi antara orang-oranglah yang membangun perangkat lunak yang hebat, memecahkan masalah, dan memberikan nilai. Orang-orang harus mampu berkomunikasi dan bekerja sama. Jika Anda tidak melakukannya dengan benar, Anda tidak akan berhasil. Alat dan

proses masih dapat memberikan nilai, tetapi keduanya bukanlah peluru ajaib. Ambil contoh telepon pintar. Entah mengapa, semakin kita terhubung di abad ke-21, semakin sedikit koneksi yang sebenarnya kita miliki. Kita tampaknya ingin bersembunyi di balik telepon kita dan tidak berinteraksi dengan orang yang berdiri di sebelah kita. Ketika putri saya dan saya berada di mal dan dia melihat seseorang yang tidak ingin dia ajak bicara, dia mengeluarkan teleponnya dan mulai mengirim pesan teks atau tweet atau apa pun yang dilakukan anak-anak keren saat ini.

Saya tidak memahaminya. Jika Anda meninggalkan saya sendirian di mal selama setengah jam, saya akan berbicara dengan setidaknya tiga orang dan mengetahui segalanya tentang mereka. Saya bahkan berusaha untuk memulai percakapan dengan orang-orang ketika mereka berada di lift bersama saya. Terkadang, saya pikir saya membuat mereka takut. Saya kira seluruh hal tentang monster itu menjadi bumerang bagi saya dalam kasus ini. Sebagian besar waktu, orang-orang bersikap agak terkejut karena penghuni planet Bumi lainnya mengakui keberadaan mereka. Saya kira saya orang yang suka bersosialisasi.

1.4 PERANGKAT LUNAK LEBIH UTAMA DARIPADA DOKUMENTASI

Jika Anda mengalami masalah tidur, mulailah membaca beberapa dokumentasi teknis. Pikirkanlah kapan Anda membaca dokumentasi yang menyertai sesuatu yang Anda beli? Apakah hal pertama yang Anda lakukan saat membeli mobil adalah mencari kursi yang nyaman dan membaca buku panduan pemiliknya? Saya bahkan tidak tahu di mana buku panduan pemiliknya untuk mobil saya saat ini. Sebagian besar pengembang yang sangat baik yang saya kenal berorientasi pada detail dan sangat mendalami proses. Pengembang mendapat manfaat dari karakteristik ini, tidak diragukan lagi, tetapi itu juga berarti mereka dapat dengan mudah teralihkan dari apa yang seharusnya menjadi fokus mereka:

Menulis perangkat lunak yang menghasilkan nilai bagi pelanggan

Tim perlu mempertimbangkan perspektif pengguna saat membangun perangkat lunak. Ingat, tujuannya adalah membangun sesuatu yang benar-benar diinginkan pelanggan. Dokumentasi tetap diperlukan, tetapi tidak dalam skala besar. Saat pembuatan dokumentasi menjadi prioritas daripada menghasilkan nilai, semuanya menjadi kacau. Saya tidak mengatakan bahwa kita berhenti membuat dokumentasi. Kuncinya adalah memiliki dokumentasi yang cukup untuk mendukung perangkat lunak yang berfungsi. Perhatikan bahwa tertulis perangkat lunak yang berfungsi. Ini berarti tidak ada cacat. Saat bug ditemukan, bug tersebut segera diatasi bahkan jika itu berarti tim berhenti mengerjakan fitur baru untuk melakukannya.

Kolaborasi Pelanggan daripada Negosiasi Kontrak

Pelanggan menggunakan perangkat lunak kita dengan cara yang tidak pernah kita pikirkan untuk memecahkan masalah yang tidak pernah kita dengar. Kenyataannya adalah bahwa pengembang bukanlah pengguna. Tim scrum perlu berkolaborasi dengan pelanggan untuk mendapatkan pemahaman yang kuat tentang apa yang mereka cari. Agar tim dapat membuat perangkat lunak yang diinginkan pelanggan, Anda perlu mengetahui apa yang mereka, Anda tahu inginkan.

Kontrak tidak menggantikan komunikasi. Saya pernah menjadi bagian dari proyek yang

disukai manajemen tetapi tidak pernah terwujud di pusat data karena orang-orang yang benar-benar mengerjakannya tidak melihat nilainya. Tim tidak hanya harus berkolaborasi dengan pelanggan, mereka juga harus berkolaborasi dengan orang-orang yang "paling dekat dengan kaca", orang-orang yang menggunakan produk.

Ingat, persyaratan berubah dengan cepat. Pelanggan mungkin tidak tahu apa yang mereka inginkan pada awalnya. Mereka akan berubah pikiran. Mereka mungkin tidak berkomunikasi dengan baik. Kolaborasi yang konstan memungkinkan tim mengatasi hambatan ini dan memberikan sesuatu yang benar-benar berharga bagi pelanggan. Kolaborasi memungkinkan tim untuk terus berfokus pada tujuan akhir, dan bukan hal lain.

Menanggapi Perubahan Daripada Mengikuti Rencana

Perubahan itu mengganggu. Perubahan itu berantakan. Hindari perubahan dengan cara apa pun. Begitulah perubahan dipandang sebelum Agile Manifesto. Itu dulu. Ini sekarang. Ekonomi aplikasi mengganggu cara pelanggan menjalankan bisnis mereka, dan bagaimana hal itu mengubah apa yang mereka butuhkan dari produk yang mereka beli. Saat ini, persyaratan berubah begitu cepat sehingga bodoh untuk mencoba merencanakan rilis skala besar di awal. Tim Agile perlu merangkul perubahan. Ini berarti mereka perlu mengubah cara kerja. Kita tahu perubahan akan terjadi.

Kita harus mengantisipasinya. Bagian dari rencana rilis kita harus beradaptasi dengan perubahan. Daripada berkata, "Lihat apa yang saya buat untuk Anda. Keren, bukan?" kita harus berpikir tentang cara menyampaikan potongan-potongan kecil dari apa yang kita rencanakan untuk dibangun secara berkala. Dengan cara ini, pelanggan bisa mendapatkan apa yang kita bangun. Mereka bisa mencoba fungsionalitas baru dan memberi kita umpan balik. Dan mengharapkan perubahan yang mengganggu datang dari umpan balik pelanggan yang kita dapatkan.

Itu berarti kita tidak merencanakan 10 bulan ke depan dengan detail yang rumit. Rencana rilis tim dan prioritas Backlog harus terus berubah dan berkembang karena perubahan persyaratan dan umpan balik. Kekuatannya ada pada kemampuan untuk berputar dan beradaptasi dengan cepat. Ini memastikan bahwa apa yang dibangun tim adalah apa yang diinginkan pelanggan. Bahkan jika itu bukan yang direncanakan semula.

1.5 12 PRINSIP AGILE YANG PENTING

Selain manifesto, penulis mendefinisikan 12 prinsip. Kebanyakan orang tidak merujuk pada prinsip-prinsip panduan ketika mereka berbicara tentang Agile Manifesto. Saya tidak yakin mengapa. Saya pikir prinsip-prinsip panduan sama pentingnya dengan apa yang terkandung dalam Agile Manifesto itu sendiri, jika tidak lebih. Prinsip-prinsip ini cukup sederhana untuk dipahami, tetapi juga memiliki kompleksitas yang lebih dalam yang memunculkan makna Agile yang lebih dalam.

1. Prioritas tertinggi kami adalah memuaskan pelanggan sejak awal dan terus menghadirkan perangkat lunak yang berharga

Dulu ketika saya bekerja di bagian dukungan pelanggan, kami menggunakan frasa "puaskan pelanggan." Pikirkan itu sejenak. Ketika Anda puas dengan sesuatu, itu adalah hal

yang baik. Tidak ada yang salah dengan rasa puas, tetapi ketika Anda merasa senang, itu berarti membawanya ke tingkat yang sama sekali baru. Jika saya selesai menyantap hidangan besar di restoran mewah, saya mungkin merasa puas. Saya akan tersenyum, menepuk perut, dan pada umumnya senang dengan apa yang baru saja saya makan. Jika hidangan itu menyenangkan saya, saya tidak sabar untuk memberi tahu semua teman saya tentang hidangan lezat yang saya santap di restoran itu.

Saya menggunakan media sosial secepat mungkin, memberi tahu semua orang tentang pengalaman saya. Ini menjadi semacam mantra jangan hanya memuaskan pelanggan, buat mereka senang. Itulah yang seharusnya kita lakukan senangkan pelanggan. Bukan menyenangkan manajemen. Senangkan pelanggan kita. Jika tim berfokus pada hal lain selain pelanggan, maka fokusnya salah. Tujuannya bukan hanya untuk menghasilkan sesuatu yang diinginkan pelanggan, tetapi sesuatu yang tidak sabar untuk dimiliki pelanggan. Hal ini mengharuskan pengembang untuk mengubah cara mereka bekerja. Perlu untuk mengerjakan ulang cara penulisan kode. Hal ini mungkin mengharuskan tim untuk mengubah cara mereka bekerja sama. Hal ini dapat menjadi perjuangan, terutama jika tidak semua orang dalam tim merasa nyaman dengan ide ini. Tim perlu memahami bahwa lebih baik melakukan kesalahan di awal dan memiliki waktu untuk mengubah dan memberikan apa yang benar-benar diinginkan pelanggan.

Pikirkan hal ini seperti kue pengantin. Katakanlah Anda membayar untuk sebuah pernikahan. Ya, itu sangat menakutkan. Percayalah, saya memiliki dua orang putri. Saya pikir semua orang akan setuju bahwa di samping gaun pengantin, kue pengantin adalah keputusan terpenting yang harus dibuat. Mengetahui hal ini, sebagai orang yang membeli kue, Anda ingin kuenya sempurna. Saya yakin bahwa koki kue yang membuat kue ingin menyenangkan Anda. Namun, jika kue besar bertingkat tujuh muncul pada hari pernikahan dan pengantin wanita tidak menyukainya, ada kemungkinan besar kita memiliki Bridezilla yang hidup dan bernapas di tangan kita.

Percayalah; tidak ada yang menginginkan itu, jadi kita akan mencicipinya. Kita bisa mencicipi kue dan melihat seperti apa bentuknya nanti. Saya tidak berharap melihat kue besar di sini. Saya mengharapkan Produk Minimal yang Layak. Versi lebih kecil dari produk jadi yang berdiri sendiri. Kita bisa mencicipi kue, lapisan gula, isian, dan apa pun. Kita mungkin juga bisa melihat beberapa teknik dekorasi yang direncanakan, tetapi kita tidak melihat kue yang sudah jadi. Dengan cara ini, kita bisa memberikan umpan balik dan memberi tahu koki kue apa yang kita inginkan. Mungkin kita ingin isian stroberi di antara lapisan, tetapi sekarang kita tidak menyukai ide itu, atau kita lebih suka bunga daripada burung, atau fondant daripada lapisan gula krim mentega.

Idenya adalah kita mendapatkan gambaran tentang seperti apa produk akhir dengan mengerjakan prototipe kecil. Kita kemudian dapat memberikan umpan balik dan mendapatkan apa yang kita inginkan. Dalam contoh kue pengantin, kita mungkin hanya melakukan ini satu atau dua kali. Saat mengembangkan perangkat lunak, tim memberikan hasil lebih awal dan sering sehingga ada banyak kesempatan untuk umpan balik dari orang-orang yang akan menggunakan perangkat lunak tersebut. Itu memastikan bahwa apa yang Anda

bangun bukanlah hal yang salah, tetapi hal yang benar. Yang membawa kita ke poin yang paling penting. Pada akhirnya, yang dipedulikan pelanggan hanyalah bahwa apa yang Anda hasilkan adalah sesuatu yang berharga bagi mereka. Itulah yang perlu difokuskan oleh tim.

2. Terima perubahan persyaratan bahkan di tahap akhir pengembangan

Agile adalah tentang beradaptasi dengan perubahan, bukan merencanakan semuanya di awal. Jika Anda belum menyadarinya, saya telah berbicara cukup banyak tentang perubahan. Perubahan adalah alasan tim perlu mengubah cara kerja, tetapi lebih dari itu. Pelanggan tidak dapat menyampaikan persyaratan mereka, karena persyaratan tersebut terus berubah. Tim Agile bekerja dengan cara yang mengharapkan dan menerima perubahan. Dengan cara ini, persyaratan pelanggan dapat terpenuhi bahkan jika berubah di akhir proyek.

3. Kirimkan perangkat lunak yang berfungsi secara berkala, dari beberapa minggu hingga beberapa bulan, dengan preferensi pada jangka waktu yang lebih pendek

Saya ingat ketika saya menantang tim saya untuk mengirimkan apa yang sedang kami bangun dalam peningkatan kecil. "Kami tidak dapat mengirimkan semua fungsionalitas itu dalam waktu sesingkat itu," kata mereka. "Bagaimana jika Anda mengerjakan beberapa fungsionalitas dan membuat tiruan sisanya?" Saya menyarankan. Raut wajah mereka tak ternilai harganya. Seolah-olah saya menyarankan agar kita semua memotong jari dan mencoba menjualnya kepada mereka. Serius, mereka ngeri.

Sudah menjadi sifat manusia untuk tidak ingin memamerkan sesuatu sampai selesai. Namun, tidak masuk akal untuk menahan sesuatu sampai kita menganggapnya, yah... bagus. Sebuah tim membutuhkan umpan balik sedini mungkin dalam proyek, dan kemudian secara berkala. Cara terbaik bagi pelanggan untuk memberikan umpan balik yang bermanfaat adalah dengan mencoba apa yang telah dibangun. Suatu tim dapat membuat tayangan slide, diagram teknis, dan bahkan demo fungsionalitas yang akan mereka hasilkan, tetapi tidak ada yang lebih baik daripada perangkat lunak yang berfungsi.

Suatu tim bekerja dalam iterasi, atau Sprint, yang berlangsung selama dua hingga empat minggu. Sasaran Sprint adalah untuk menghasilkan produk minimal yang layak (MVP) yang dapat dicoba oleh pelanggan. Setelah mencoba perangkat lunak, pelanggan dapat memberikan umpan balik yang sangat mereka dambakan kepada tim. Bonus tambahannya adalah status proyek tersebut setransparan mungkin. Pelanggan tahu persis apa yang sedang diproduksi oleh tim dan berapa lama waktu yang dibutuhkan. Berbicara tentang itu, suatu tim harus berusaha membuat Sprint sesingkat mungkin. Itu masuk akal jika Anda memikirkannya. Semakin pendek Sprint, semakin banyak umpan balik yang dapat diberikan pelanggan.

4. Pelaku bisnis dan pengembang harus bekerja sama setiap hari selama proyek berlangsung

Ide di balik proyek perangkat lunak adalah untuk memuaskan bisnis. Dengan kata lain, untuk menghasilkan uang. Mudah bagi suatu tim untuk berfokus pada hal yang salah. Bila Anda menjaga tim tetap fokus pada bisnis, mereka akan memiliki perspektif yang tepat. Dengan cara ini, tim akan belajar lebih banyak tentang bisnis daripada dengan membahas daftar persyaratan. Tim perlu didorong untuk berinteraksi dengan pelanggan sebanyak mungkin. Mungkin tidak mungkin bagi tim dan pelanggan untuk berinteraksi setiap hari, tetapi kontak harus dilakukan sesering mungkin.

5. Bangun proyek di sekitar individu yang termotivasi. Berikan mereka lingkungan dan dukungan yang mereka butuhkan, dan percayalah kepada mereka untuk menyelesaikan pekerjaan

Tim adalah entitas yang paling penting. Mungkin mustahil untuk membangun "tim impian", tetapi sejujurnya hal itu tidak sepenting yang Anda pikirkan. Meskipun demikian, manajemen tidak dapat mengumpulkan sekelompok programmer junior dan mengharapkan keajaiban terjadi. Kuncinya adalah membentuk tim yang dapat bekerja sama, lalu biarkan mereka mengerjakannya. Pada masa Waterfall, manajer pengembangan akan memberi tahu tim apa yang harus dilakukan dan bagaimana melakukannya. Pelanggan memanfaatkan setiap kesempatan untuk mencoba membuat tim memasukkan fungsionalitas yang mereka inginkan ke dalam produk.

Tenaga penjualan akan mengganggu tim agar meninggalkan semuanya dan menyediakan fungsionalitas yang akan membantu menutup penjualan. Tim juga diminta untuk memberikan demo, membantu dukungan, dan banyak hal yang saya lupa. Tim Agile terisolasi dari semua ini. Alih-alih manajer pengembangan, tim memiliki Pemilik Produk yang bertanggung jawab untuk berbicara dengan pelanggan, memahami cara mereka bekerja dan nilai apa yang akan mereka peroleh dari proyek, dan membangun Backlog pekerjaan yang diprioritaskan. Tim menggunakan Backlog untuk membangun apa yang diinginkan pelanggan. Scrum Master (peran Agile lainnya) bertanggung jawab untuk melindungi tim. Semua gangguan hilang, memungkinkan tim untuk fokus.

6. Metode yang paling efisien dan efektif untuk menyampaikan informasi kepada dan di dalam Tim Pengembangan adalah percakapan tatap muka

Komunikasi tatap muka paling efektif karena semua aspek komunikasi berperan. Anda memiliki ekspresi wajah, nada, bahasa tubuh, dan kata-kata yang sebenarnya diucapkan. Salah satu pengalaman melatih favorit saya adalah ketika ibu seorang pemain berjalan menghampiri pelatih kepala dan saya. Saya dapat melihat dari raut wajahnya bahwa dia kesal tentang sesuatu. Sebenarnya, marah seperti induk ayam yang basah mungkin merupakan deskripsi yang lebih baik.

Dia mendekati pelatih kepala dengan tidak nyaman dan berkata, "Saya tidak menghargai nada email Anda." Dia kemudian berbalik dan pergi. “Bagaimana Anda mendapatkan nada dari sebuah email?” kata pelatih kepala dengan ekspresi heran di wajahnya. Komunikasi sangatlah penting. Agile tidak akan berjalan tanpanya, jadi kita harus berusaha menggunakan metode yang paling efektif. Jika Anda mencoba berkomunikasi menggunakan email atau semacam Instant Messenger, Anda kehilangan satu atau beberapa aspek dari proses komunikasi.

Dan itu lebih cepat. Jika kita berkomunikasi melalui email, Anda harus menunggu hingga saya membaca email asli dan menanggapi untuk mengetahui apakah saya benar-benar memahami apa yang Anda coba katakan, apalagi perasaan saya tentang subjek tersebut. Bertemu langsung, itu terjadi dengan segera. Hasilnya, pemahaman mengalir dengan mudah.

7. Perangkat lunak yang berfungsi adalah ukuran utama kemajuan

Perangkat lunak yang berfungsi harus menjadi fokus tim. Bukan pada dokumentasi, atau desain, atau apa pun. Tim Pengembangan tampaknya ingin fokus pada penyelesaian pengembangan. Pengembangan yang selesai dan perangkat lunak yang berfungsi belum tentu merupakan hal yang sama. Dengan kata lain, perangkat lunak tidak selesai jika dikompilasi atau dirakit. Perangkat lunak harus "berfungsi", artinya dapat diuji dan dapat dibuktikan. Mari kita kembali ke contoh kue pengantin.

Jika koki pastry datang ke acara mencicipi kue dengan membawa resep di tangannya dan memberi tahu Anda bahwa ia telah menyempurnakan resep untuk pernikahan Anda tetapi tidak memiliki kue untuk Anda uji, apakah Anda akan senang? Menurut saya tidak. Agar berhasil, koki pastry perlu fokus pada kue yang lezat. Tim perlu fokus pada perangkat lunak yang berfungsi.

8. Proses Agile mendorong pembangunan berkelanjutan. Sponsor, pengembang, dan pengguna harus dapat mempertahankan kecepatan yang konstan tanpa batas waktu

Kesibukan untuk menyelesaikan proyek hingga tanggal GA membutuhkan waktu yang lama. Tim yang saya ikuti bekerja sama dan menyelesaikan pekerjaan, tetapi itu tidak menyenangkan. Saya yakin kecepatan itu tidak dapat dipertahankan lebih dari beberapa minggu. Lihatlah dari sudut pandang ini. Pelari maraton tercepat di dunia berlari sejauh 26,2 mil dengan kecepatan enam menit per mil. Kecepatan itu dapat dipertahankan bagi mereka, tetapi itu jauh dari kata sprint habis-habisan. Tidak seorang pun yang waras akan mencoba berlari sprint sepanjang maraton.

Tim Pengembangan juga tidak boleh. Ketika saya masih dalam pelatihan dasar, sersan pelatih gemar mengatakan bahwa jika kami seharusnya memiliki keluarga, itu akan diberikan kepada kami dan dihukum di sisi kanan loker kaki kami. Ini seharusnya bukan sikap yang berlaku pada orang-orang yang bekerja dengan Anda. Keseimbangan kehidupan kerja itu penting. Kecepatan yang berkelanjutan mendukung keseimbangan kehidupan kerja yang sehat.

9. Perhatian berkelanjutan terhadap keunggulan teknis dan desain yang baik meningkatkan ketangkasan

Kode yang bersih lebih mudah digunakan daripada kode spaghetti. Itu seharusnya jelas. Saya ragu ada orang yang pernah bekerja sebagai pengembang yang bangun suatu hari dan berkata, "Hari ini saya akan menulis kode yang rumit, berbelit-belit, dan sulit dipahami." Itu terjadi karena tekanan yang diperlukan untuk menyelesaikan proyek dan memenuhi tenggat waktu. Tim Agile harus selalu memperhatikan kode yang mereka tulis. Begitu kode menjadi berantakan, Anda kehilangan kesempatan untuk berputar cepat dan menanggapi umpan balik. Dengan kata lain, Anda kehilangan ketangkasan.

Begitu pula dengan desain. Mengapa harus membuang waktu dan membuat desain yang rumit untuk keseluruhan proyek jika Anda mengharapkan perubahan? Desain harus bersih, efisien, dan dilakukan dengan cara yang menerima perubahan. Saya suka mengatakan bahwa pengodean dan desain harus menerima perubahan dengan lapang dada.

10. Kesederhanaan seni memaksimalkan jumlah pekerjaan yang tidak dilakukan sangat penting

Hal ini tampaknya berlawanan dengan intuisi. Bagaimanapun, Tim Pengembangan ada terutama untuk menulis perangkat lunak. Ya, itulah idenya. Tim harus fokus membangun hal-hal yang luar biasa asalkan mereka membangun hal-hal yang tepat. Hal ini mengharuskan pengembangan untuk membingkai ulang cara mereka berpikir tentang penulisan kode. Jika pengembang menulis fungsi sederhana, mereka akan berasumsi bahwa mereka perlu melihat semua skenario yang mungkin dan melakukan semua jenis pekerjaan yang sebenarnya tidak perlu.

"Yah, ini mungkin terjadi, jadi sebaiknya saya membangun sesuatu ke dalam fungsi untuk itu". "Sementara saya melakukannya, saya mungkin juga memanggil rutinitas lain ini untuk memastikan datanya bagus. Dan saya rasa saya harus mempersiapkan hal lain ini juga." Tim Agile harus membuat fungsi sesederhana mungkin. Masuk, lakukan pekerjaan sesederhana mungkin, dan keluar. Saya bersikap sederhana di sini (saat membahas kesederhanaan, tidak kurang), tetapi ini adalah cara tim perlu melihat pengembangan.

Semua hal lain yang Anda pikir "mungkin" Anda perlukan adalah buang-buang waktu. Jika Anda perlu menambahkannya di masa mendatang, tambahkan saat itu juga. Saya pernah mendengar orang mengatakan bahwa sebagian besar fitur yang dibangun hampir tidak pernah digunakan. Kompleksitas menyebabkan ketidakandalan. Tim Agile berusaha untuk kesederhanaan dengan hanya membangun apa yang dibutuhkan saat itu.

11. Arsitektur, persyaratan, dan desain terbaik muncul dari tim yang mengorganisasi diri sendiri

Saya melatih lari lintas alam selama 5 tahun saat putri saya berlari. Olahraga lari lintas alam persis seperti namanya. Anda berlari di hutan. Gaya kepelatihan saya biasanya adalah dengan menunjukkan lintasan kepada para pelari dan berkata, "Lakukan saja." Kemudian, saya akan berdiri di garis finis sambil memegang stopwatch dan menunggu mereka. Orang tua selalu bertanya kepada saya apakah saya akan berlari bersama anak-anak atau memberi mereka petunjuk tentang cara berlomba. Jawaban saya selalu sama: "Mereka akan menemukan jawabannya sendiri".

Sebenarnya, saya akan lebih banyak mengacaukan daripada memperbaikinya jika saya terlalu terlibat. Berlari itu sederhana; kita semua bisa melakukannya. Berlomba membutuhkan dedikasi dan keinginan. Seseorang seperti saya yang berteriak kepada Anda untuk berlari lebih cepat bukanlah motivasi yang tepat. Ingin mengalahkan anak dengan kaus warna berbeda adalah motivasi yang tepat. Saya selalu merasa heran bahwa setelah lomba pertama, tim berhenti berjalan dan mulai berlari dengan keras. Lampu lalu lintas menyala.

Hal yang sama berlaku untuk tim Agile. Mereka tidak memerlukan manajer pengembangan yang memberi tahu mereka cara melakukan segalanya. Sama seperti pelari, tim mencari tahu seiring berjalannya waktu. Alih-alih mencoba merancang segalanya di awal, tim membiarkan desain muncul saat mereka membangun fungsionalitas. Beri mereka tujuan dan jangan ikut campur. Mereka akan menemukan cara terbaik untuk mencapainya.

12. Secara berkala, tim merenungkan cara menjadi lebih efektif, lalu menyesuaikan dan menyempurnakannya

Agile adalah tentang memeriksa dan beradaptasi dengan fokus pada eksperimen.
Bob Carpenter, CSM, ICP-ACC, CSP

Agile adalah tentang peningkatan berkelanjutan. Tim Agile memeriksa semua yang dilakukannya dan secara rutin membuat perubahan dalam upaya untuk mencoba menjadi lebih baik. Melakukan post-mortem setelah proyek selesai memang bagus, tetapi proyek itu sudah berakhir. Kapal itu telah berlayar, dan pelajaran yang dipelajari hanya dapat diterapkan pada proyek mendatang.

Sebaliknya, tim Agile meluangkan waktu untuk melakukan retrospeksi secara berkala selama proyek berlangsung. Dalam retrospeksi ini, tim mengidentifikasi cara untuk melakukan perbaikan dan mencoba menerapkannya dalam cara kerja mereka. Tim dapat mencoba hal-hal baru, mempertahankan apa yang berhasil, dan membuang apa yang tidak. Dengan berfokus pada perbaikan yang berkelanjutan, tim tidak dapat tidak melakukan perbaikan. Peningkatan yang konstan tersebut menciptakan tim yang berkinerja tinggi.

Bab ini memberikan gambaran umum tentang pengembangan perangkat lunak Waterfall, manifesto Agile, dan dua belas prinsip perangkat lunak Agile. Bab berikutnya akan membahas Scrum, kerangka kerja ringan yang mendukung pengembangan berulang seperti yang dijelaskan dalam Manifesto Agile.

BAB 2
KERANGKA KERJA SCRUM

Ada sekitar selusin metode Agile yang digunakan saat ini. Di antara semuanya, kerangka kerja Scrum adalah yang paling populer. Agility berasal dari peningkatan siklus umpan balik dan tanggapan terhadap umpan balik tersebut. Kerangka kerja Scrum mendorong siklus umpan balik yang pendek, menurunkan risiko, dan memungkinkan laba atas investasi (ROI) terlihat lebih cepat.

2.1 PERAN, VISI, DAN BACKLOG

Scrum membutuhkan tiga peran yang mungkin tidak Anda kenal pemilik produk (PO), Master Scrum, dan anggota tim. Saya akan membahas lebih rinci tentang peran-peran ini di bab berikutnya. Beberapa organisasi memiliki peran tambahan yang berada di luar kerangka kerja Scrum, tetapi sangat penting bagi keberhasilan tim, yang disebut manajer produk (PM). PM bertanggung jawab untuk mendefinisikan visi tentang seperti apa produk ini nantinya. PM bertanggung jawab atas apa yang disebut sebagai peta jalan produk. Dengan kata lain, seperti apa produk ini dalam lima tahun, tiga tahun, satu tahun dari sekarang?

Bagaimana produk ini akan berperan di pasar? Visi adalah yang mendanai proyek dan memberi semua orang alasan untuk bangun pagi dan bekerja. Visi tersebut berubah menjadi Backlog. Ingat dokumen persyaratan dari Waterfall? Backlog seperti itu, kecuali terus-menerus didefinisikan dan diprioritaskan. Backlog adalah daftar hal-hal yang perlu dilakukan untuk mengubah visi menjadi kenyataan. Apa pun dapat masuk ke dalam Backlog. Hal-hal seperti cacat, utang teknis, permintaan peningkatan, bahkan risiko. Persyaratan fungsional dan nonfungsional. Satu-satunya kendala adalah bahwa item Backlog harus dinyatakan dalam istilah bisnis (nonteknis).

Cerita dan Epik

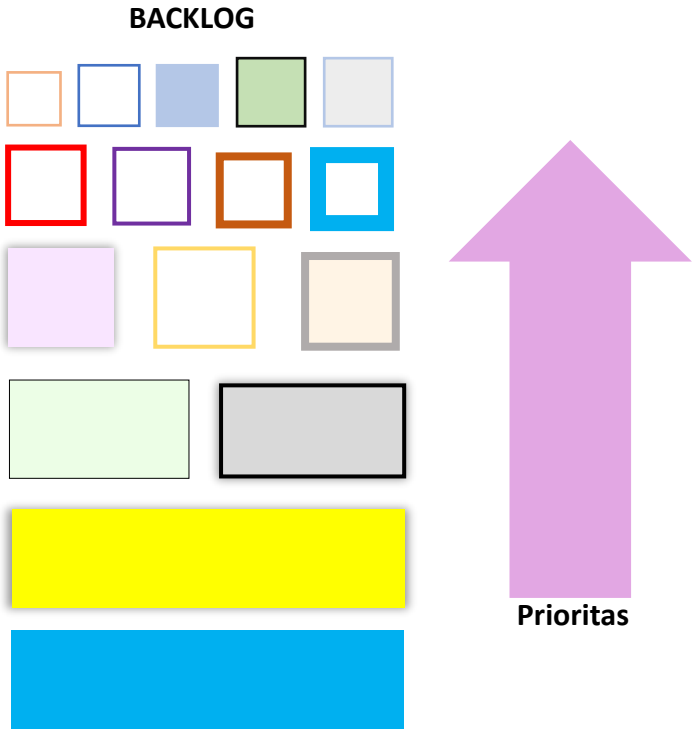
Item dalam Backlog disebut sebagai cerita pengguna. Ditulis dari perspektif pengguna, cerita menyampaikan potongan kecil fungsionalitas atau nilai bagi tim untuk dikerjakan. Cerita adalah deskripsi sederhana yang ditulis dengan cara nonteknis. Idenya adalah mengalihkan fokus dari menulis tentang fitur menjadi membicarakannya. Epik adalah cerita pengguna yang sangat besar yang perlu dipecah, atau kumpulan cerita pengguna. Epik adalah cara mudah untuk mengatur Backlog dengan cara yang jelas tanpa harus terlalu mendalami kompleksitas. Pikirkan ini dalam konteks musik.

Anda dapat mengurutkan musik berdasarkan genre, artis, album, dan lagu. Setiap level menjadi sedikit lebih spesifik dan kompleks. Anda dapat melihat dan mengamati bahwa Anda memiliki banyak musik dalam genre Heavy Metal, lalu memecahnya dan melihat berapa banyak lagu yang ada berdasarkan album dan/atau artis. Jika Anda ingin mengelompokkan musik Anda berdasarkan tema tertentu, Anda dapat melakukannya. Hal yang sama berlaku untuk Backlog yang menggunakan epik.

Product Owner Memiliki Backlog

Product owner (PO) bertanggung jawab atas Backlog. Saya suka mengatakan bahwa PO tinggal di Backlog. PO memahami bisnis dan produk yang akan dibuat oleh tim. Mereka tahu seperti apa produk itu seharusnya, apa yang seharusnya dilakukan, dan yang terpenting nilai apa yang akan diberikannya kepada pelanggan. Memiliki Backlog yang terdefinisi dengan baik memungkinkan PO untuk mendorong tim untuk menciptakan sesuatu yang bernilai. Penting untuk dicatat bahwa ketika item pertama kali ditempatkan di Backlog, item tersebut bisa sangat umum.

Saat epik dipecah menjadi cerita dan cerita disempurnakan, item tersebut menjadi lebih kecil dan lebih terperinci. Gambar yang sering saya gunakan adalah cerita yang "naik" dalam Backlog, seperti yang ditunjukkan pada Gambar 2.1. Pada saat sebuah cerita mencapai puncak tumpukan (diprioritaskan paling tinggi dan siap untuk dikerjakan), cerita tersebut seharusnya sudah terdefinisi dengan baik dan siap untuk dikerjakan oleh tim Scrum.



Gambar 2.1 Backlog Yang Sehat

Penting untuk dipahami bahwa kita tidak berharap untuk mengetahui semua persyaratan di awal. Dengan kata lain, PO tidak diharapkan untuk membangun Backlog yang sempurna sebelum tim mulai membuat kode. Backlog akan tumbuh dan berkembang seiring berjalannya proses Scrum dan persyaratan berubah dan/atau menjadi lebih jelas. PO menambahkan cerita ke Backlog dan memprioritaskannya menurut nilai bisnis. Nilai bisnis tersebut dapat berubah dengan cepat, jadi PO harus terus memprioritaskan Backlog. Cerita yang memberikan nilai bisnis tertinggi saat ini harus berada di urutan teratas daftar. PO memahami bagaimana pelanggan menggunakan perangkat lunak serta nilai apa yang mereka dapatkan darinya.

Mereka peduli dengan apa dan mengapa, bukan bagaimana. Terserah tim untuk

mencari tahu cara menyampaikan apa yang diminta PO. Pemilik produk tidak hanya harus mampu mengartikulasikan dengan jelas apa yang diinginkan, tetapi juga mengapa itu penting. Orang-orang tidak benar-benar berinvestasi pada apa yang Anda inginkan, tetapi ketika Anda memberi tahu mereka alasannya tujuannya, penyebabnya mereka akan terhubung dan bersemangat untuk melakukan pekerjaan tersebut. Para pemangku kepentingan (baik internal maupun eksternal) ingin ikut menentukan bagaimana produk dibangun. Para pemangku kepentingan eksternal adalah pelanggan, alias orang-orang yang memberi kita uang untuk apa yang kita bangun.

Contoh pelanggan internal adalah dukungan produk. Seorang pemangku kepentingan internal sangat tertarik dengan apa yang sedang dibangun oleh tim tetapi merupakan bagian dari perusahaan yang sama. PO bertindak sebagai penyangga antara semua orang ini dan tim Scrum. Misalnya, jika departemen hukum mengharuskan tim untuk menerbitkan perjanjian lisensi pengguna akhir untuk beberapa perangkat lunak pihak ketiga yang mereka gunakan, mereka tidak langsung menghubungi tim. Mereka menghubungi PO, yang akan memasukkan persyaratan itu ke dalam Backlog dan memastikannya diprioritaskan dengan benar.

EULA (Perjanjian Lisensi Pengguna Akhir) itu mungkin tidak harus diselesaikan sekarang, tetapi harus diselesaikan sebelum produk dirilis. Hal ini memungkinkan tim untuk fokus pada tugas yang ada dan tidak terganggu oleh pengaruh eksternal. Tim terlindungi karena PO menerima permintaan mereka dan membuat Backlog cerita pengguna.

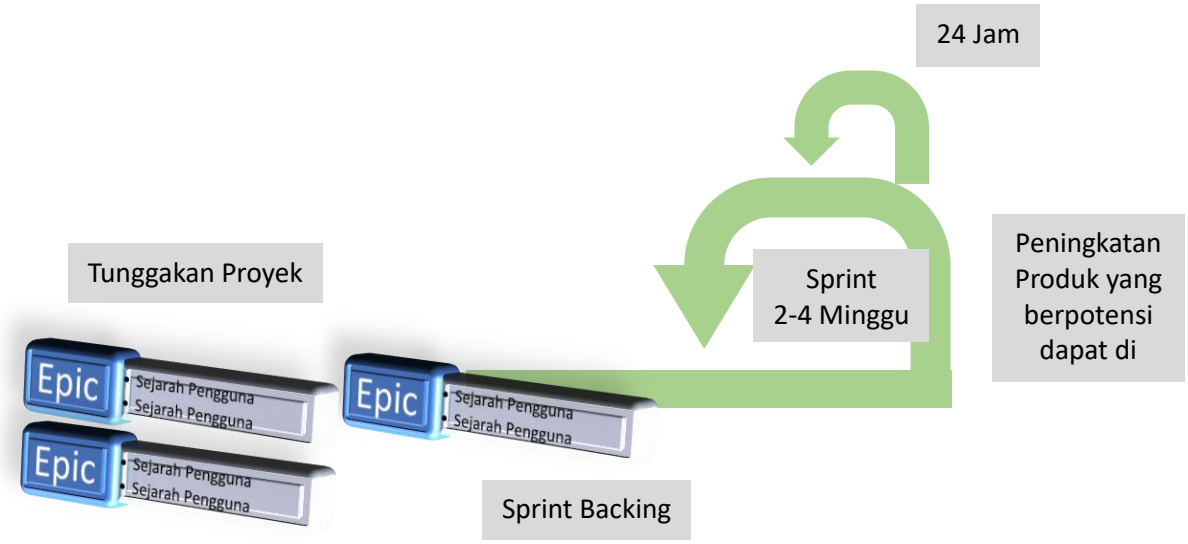
2.2 SPRINT

Berbicara tentang tim, tim Scrum adalah tim lintas fungsi yang bekerja dengan cara yang sangat berbeda dari zaman Waterfall. Tim berkomitmen untuk mencapai sejumlah kecil sasaran dalam waktu singkat. Tim melakukan apa pun untuk mencapai sasaran tersebut. Ketika saya mengatakan "apa pun," yang saya maksud bukan sekadar tidur di meja kerja dan tidak pernah bertemu keluarga. Yang saya maksud adalah fakta bahwa anggota tim melakukan apa yang perlu dilakukan tanpa memandang peran. Misalnya, pengembang dapat melakukan beberapa pekerjaan QA untuk menyelesaikan sesuatu dan mencapai sasaran, atau sebaliknya.

Tim akan mengambil Product Backlog, dan dengan bantuan Scrum Master, memasukkan cerita ke dalam Sprint Backlog selama rapat perencanaan Sprint. Pada rapat ini, PO membahas apa yang ingin dicapainya dalam Sprint, dan tim berkomitmen untuk menyelesaikan sejumlah cerita untuk mencapai sasaran Sprint tersebut. Penting untuk dicatat bahwa setelah Sprint Backlog ditetapkan, tidak seorang pun dapat mengubahnya. Kata "komitmen" akhir-akhir ini tidak disukai, tetapi menurut saya kata itu terlalu kuat untuk tidak digunakan. Yang menurut saya kuat adalah bahwa tim tidak lagi bekerja untuk memenuhi tuntutan yang tidak realistis yang diajukan oleh manajemen.

Mereka masih memiliki tekanan untuk menyelesaikan sesuatu pada akhir Sprint, tetapi hanya apa yang telah mereka komitmenkan. Lihatlah seperti ini: Jika saya memberi tahu Anda bahwa saya akan melakukan sesuatu, saya akan berinvestasi di dalamnya. Tidak sama jika Anda memberi tahu saya untuk melakukannya. Sprint adalah iterasi 2.4 minggu di mana tim bekerja untuk menghasilkan peningkatan produk yang berpotensi dapat dikirim (produk yang layak

minimal) seperti yang ditunjukkan pada Gambar 2.2.



Gambar 2.2 Iterasi Scrum

Mengapa 2 sampai 4 minggu? Karena itu adalah periode waktu yang mudah dikontrol. Kebanyakan manusia sangat pandai menunda sesuatu hingga benar-benar harus diselesaikan. Membatasi "kotak waktu" tim hingga 24 minggu memaksa mereka untuk menyelesaikan tugas yang ada. Tidak ada waktu yang terbuang. Di akhir Sprint, tim mengadakan rapat tinjauan Sprint tempat tim menunjukkan kepada para pemangku kepentingan apa yang telah mereka buat dan meminta umpan balik. Tim kemudian menerima umpan balik tersebut, menyesuaikannya, dan mengulanginya lagi dengan Sprint 2 sampai 4 minggu lainnya.

Melakukan Sprint berulang-ulang dan menanggapi umpan balik yang diberikan tentang apa yang dihasilkan sangatlah hebat. Saya mengartikan secara umum, tetapi anggaplah Sprint sebagai rilis kecil yang termuat dalam kurun waktu 2 sampai 4 minggu. Tim terus-menerus merilis perangkat lunak yang berfungsi kepada para pemangku kepentingan di setiap batas Sprint dan memungkinkan para pemangku kepentingan memberikan pendapat mereka tentangnya. Tugas utamanya adalah menerima umpan balik tersebut dan menggunakannya untuk menciptakan sesuatu yang bernilai bagi pelanggan. Saat mengerjakan Sprint, tim melakukan iterasi setiap hari (mengulangi dalam iterasi. Saya pusing di sini) dengan mengadakan rapat standup, atau rapat Scrum harian. Rapat standup bukanlah rapat status, atau rapat pengembangan, atau rapat staf.

Rapat ini ditujukan bagi tim untuk saling memberi tahu tentang apa yang telah mereka lakukan pada hari sebelumnya, apa yang akan mereka lakukan hari ini, dan apakah mereka memerlukan bantuan apa pun. Hal ini dimaksudkan agar tim dapat berkolaborasi dan bekerja secara efisien, serta menangani masalah secepat mungkin. Saya juga harus mencatat bahwa setiap bagian dari fungsionalitas harus selesai pada akhir Sprint. Itu termasuk semua QA, dokumentasi, dan sebagainya. Di akhir setiap Sprint, tim perlu menyediakan "produk yang berpotensi dapat dikirim". Berbicara tentang selesai, bagaimana tim tahu bahwa mereka telah selesai? Yang saya maksud dengan "selesai" adalah selesai. Bukan hanya kode yang lengkap.

Tidak selesai kecuali untuk beberapa pengujian otomatis yang perlu ditulis. Selesai, lengka, siap dikirim.

2.3 MENETAPKAN DEFINISI SELESAI DALAM SCRUM

Tim mengembangkan, merancang, dan menyetujui definisi selesai (DoD). DoD adalah daftar periksa hal-hal yang perlu diselesaikan sebelum cerita dianggap "selesai." Ini adalah konsep yang sangat sederhana tetapi mungkin merupakan hal terpenting yang dapat dilakukan tim Scrum untuk mencapai keberhasilan. Ketika DoD ditetapkan, baik tim maupun Pemilik Produk dapat merasa nyaman dengan fakta bahwa semua hal dalam daftar tersebut telah terpenuhi. Daftar tersebut harus berisi tonggak-tonggak penting seperti kode yang telah lengkap dan diperiksa ke manajemen sumber. Skrip QA otomatis telah ditulis, dan QA manual telah selesai. Dokumentasi telah diterbitkan, dan seterusnya.

Tim bekerja untuk memenuhi kriteria penerimaan DoD dan cerita pengguna. Kriteria penerimaan ditulis sebagai bagian dari cerita pengguna. Kriteria tersebut merupakan kesepakatan antara PO dan tim mengenai apa yang perlu dilihat PO untuk menerima cerita tersebut. Saya ingin menjelaskannya seperti ini: Katakanlah saya ingin secangkir kopi, dan saya meminta Anda untuk mengambilkannya. Dengan krimer. Pada titik ini, saya benar-benar tidak peduli bagaimana Anda mendapatkan kopinya. Saya hanya ingin secangkir. Dalam istilah Agile, saya baru saja mendefinisikan sebuah cerita. Jadi sekarang tugas Anda adalah mengeluarkan cerita ini dari Backlog dan mewujudkannya, tetapi bagaimana Anda tahu kapan Anda telah memenuhi permintaan saya? Bagaimana Anda tahu bahwa Anda telah membuat saya senang?

Dalam istilah Agile, inilah mengapa kita memiliki definisi kriteria selesai dan penerimaan. DoD mendorong tanggung jawab tim Scrum siapa yang memiliki keterampilan untuk menyelesaikan hal ini, dan tugas apa yang perlu diselesaikan agar kita sebagai tim dapat mengatakan bahwa kita telah "selesai"? DoD bukanlah kriteria penerimaan yang melekat pada sebuah cerita. Dalam contoh kita, kriteria penerimaan saya adalah secangkir kopi panas muncul di meja saya. Saya tidak peduli apakah Anda pergi ke dapur di lantai tiga, atau ke kafetaria di lantai bawah, atau masuk ke mobil Anda dan pergi ke Starbucks atau Dunkin Donuts.

Saya hanya peduli dengan hasil akhirnya; itulah yang dimaksud dengan kriteria penerimaan. Secara teknis, kriteria penerimaan akan mencakup semua kondisi kepuasan pemilik produk. Definisi saya tentang "selesai" akan terlihat seperti ini:

- Cangkir terisi kopi.
- Kopi panas.
- Kopi segar.
- Krim telah ditambahkan.
- Kopi telah diantar.

Sekarang, Anda mungkin berpikir, "Di mana saya mendapatkan cangkirnya?" Rincian seperti itu akan tercakup dalam tugas-tugas yang kemudian Anda tambahkan ke cerita. Dengan DoD yang didefinisikan dengan baik, tim Scrum dapat menjadi efisien tim bekerja untuk memuaskan DoD. Hal ini memungkinkan tim dan pemilik produk untuk memahami bahwa

sebuah cerita telah selesai. Setelah cerita selesai, PO dapat menentukan apakah cerita tersebut memenuhi kriteria penerimaan: apakah cerita tersebut memberikan fungsionalitas yang dijanjikan?

Bahkan, langkah tersirat pertama dari DoD adalah bahwa cerita tersebut memenuhi kriteria penerimaan. Kembali ke contoh saya, saya dapat percaya bahwa kopi saya akan diantar dalam keadaan segar dan panas. Saya tidak perlu meminta apa pun dari Anda, dan Anda dapat yakin bahwa saya akan senang dengan apa yang telah Anda berikan kepada saya karena kriteria penerimaan saya (apa yang saya harapkan) telah didefinisikan dengan jelas.

Tidak ada lagi manajer pengembangan yang menugaskan pekerjaan kepada tim. Tim memutuskan apa yang ingin mereka kerjakan. Mereka mengemudi sendiri. Idenya adalah bahwa orang-orang yang mengembangkan perangkat lunak adalah para ahli. Mereka tahu cara terbaik untuk membangun apa yang dibutuhkan. Saat bekerja, tim bersikap setransparan mungkin sehingga setiap orang menyadari apa yang sedang terjadi, dan hal-hal dapat dengan mudah diperbaiki jika timbul masalah.

Di akhir Sprint, tim, PO, dan manajer produk mengadakan rapat tinjauan sprint. Mereka mengundang para pemangku kepentingan, menunjukkan kepada mereka fungsionalitas yang telah mereka selesaikan selama Sprint (sebaiknya dengan melakukan demo), dan mendapatkan umpan balik tentang apa yang telah dilakukan tim. Para pemangku kepentingan juga diundang untuk memasang produk yang layak secara minimal (jika memungkinkan) dan mencobanya untuk "uji coba". Dengan melakukan ini, tim mendapatkan umpan balik selama proses pengembangan, tidak hanya di akhir.

Hal ini memungkinkan tim untuk bereaksi terhadap umpan balik pemangku kepentingan dan benar-benar memberikan nilai. Jika para pemangku kepentingan tidak menyukai sesuatu, tim dapat bereaksi dengan cepat dan membuat perubahan. Jika para pemangku kepentingan menyukai produk tersebut, mereka mungkin ingin produk tersebut segera dirilis. Setelah rapat tinjauan Sprint, tim akan mengadakan retrospeksi. Ini pada dasarnya adalah post-mortem tentang Sprint yang baru saja diselesaikan, yang mencakup apa yang telah dilakukan tim, dan apa yang perlu mereka lakukan untuk menjadi lebih baik pada Sprint berikutnya.

Dan kemudian semua orang melakukannya lagi pada iterasi berikutnya. Di mana Scrum Master selama seluruh proses ini? Tugas Scrum Master adalah untuk mendorong proses Agile. Scrum Master memastikan semua anggota tim menjalankan nilai-nilai dan praktik Agile. Scrum Master menyiapkan dan mengadakan rapat, menyingkirkan hambatan, melerai pertengkaran, memoderasi ketidaksepakatan, dan membeli donat. Baiklah serius, Scrum Master melakukan segala yang mungkin untuk memfasilitasi produktivitas.

2.4 PENERAPAN SCRUM DAN ITERASI UNTUK PENGEMBANGAN YANG RESPONSIF

Jadi seperti yang Anda lihat, dengan melakukan iterasi secara berkala, mendapatkan banyak masukan pada setiap iterasi, dan mengambil tindakan berdasarkan masukan tersebut, kami menyiapkan lingkungan tempat kami menghadirkan fitur-fitur yang benar-benar diinginkan oleh orang-orang yang menggunakan perangkat lunak kami. Bekerja dengan cara

ini memungkinkan kami untuk berputar cepat dan bereaksi terhadap perubahan persyaratan dan permintaan pelanggan. Scrum memungkinkan bisnis untuk mendorong pengembangan, karena PO harus memikirkan bisnis tersebut karena Backlog diprioritaskan. Namun, tim Scrum juga diizinkan untuk berubah pikiran saat mereka mempelajari hal-hal baru. Ini adalah cara kerja baru yang merangkul Agile Manifesto. Izinkan saya untuk merangkum:

- Pemilik produk membuat Backlog yang disempurnakan dan diprioritaskan yang terdiri dari cerita pengguna.
- Tim Scrum bekerja dari Backlog. Mereka berkomitmen pada seberapa banyak pekerjaan yang dapat mereka lakukan dalam Sprint, dan memindahkan cerita-cerita tersebut ke dalam Sprint Backlog.
- Tim melakukan Sprint, atau iterasi setiap 2–4 minggu. PO dan tim sepakat untuk mencapai tujuan setiap Sprint, dan tim bekerja untuk mencapai tujuan tersebut dengan menyelesaikan cerita dalam Sprint Backlog.
- Setiap 24 jam, tim berkumpul untuk membahas kemajuan mereka dalam rapat stand-up. Mereka menjawab tiga pertanyaan: Apa yang saya lakukan kemarin? Apa yang saya rencanakan untuk dilakukan hari ini? Apa yang menghalangi saya?
- Segala hal yang menghalangi tim disingkirkan secara agresif oleh Scrum Master.
- Di akhir setiap Sprint, tim menyelenggarakan rapat tinjauan Sprint tempat mereka menunjukkan kepada para pemangku kepentingan apa yang telah mereka hasilkan dan mengumpulkan umpan balik.
- Sebelum memulai Sprint berikutnya, tim mengadakan retrospeksi tempat mereka merenungkan Sprint sebelumnya dan cara untuk meningkatkannya.

Scrum adalah kerangka kerja yang sangat sederhana yang sangat sulit diterapkan. Ia membutuhkan perubahan budaya yang membutuhkan cara berpikir baru. Dalam bab ini, kita melihat secara mendetail kerangka kerja Scrum. Dalam bab berikutnya, kita akan melihat lebih dekat peran Scrum dan bagaimana peran tersebut memengaruhi kerangka kerja.

BAB 3
PERAN SCRUM

Menggunakan Agile bukanlah perubahan proses; melainkan perubahan budaya. Percayalah, perubahan budaya adalah hal yang akan membuat Anda terjaga di malam hari. Agar Agile berhasil, orang-orang harus menerima peran baru dan mengubah cara mereka bekerja sehari-hari. Sebelum saya membahas peran Agile, sekarang adalah waktu yang tepat untuk merinci apa yang saya yakini sebagai tiga pilar yang perlu dipahami oleh setiap orang yang bekerja di lingkungan Agile. Tiga pilar Scrum adalah transparansi, inspeksi, dan adaptasi.

3.1 TIGA PILAR SCRUM

Saya sedang melatih beberapa orang dari organisasi pendukung. Saya terus mengarahkan pembicaraan kembali ke fakta bahwa semuanya benar-benar menjadi lebih baik jika setiap orang beroperasi secara transparan. Salah satu orang di manajemen bertanya, "Apa itu transparansi?". Saya terkejut dengan pertanyaan ini. Maksud saya, bukankah ini sudah jelas? Ketika Anda transparan, Anda ya, transparan. Mungkin ini tidak begitu jelas. Transparansi berarti bahwa semua hal yang berkaitan dengan proyek harus diketahui semua orang. Saya lebih suka menggambarkannya sebagai berbagi informasi secara berlebihan.

Semua keberhasilan dan kegagalan tim harus diketahui semua orang. Tim beroperasi secara transparan dan berbagi informasi; pemilik produk juga bersikap transparan dan berbagi. Scrum Master memposting semua informasi tim yang relevan pada radiator informasi sehingga para pemangku kepentingan dapat dengan mudah mengetahui bagaimana Sprint berjalan. Radiator adalah tampilan (atau dasbor) yang diletakkan di area dengan lalu lintas tinggi. Alih-alih menyembunyikan informasi, tim Scrum menyiarkan semua hal tentang apa yang sedang mereka lakukan. Ini adalah sesuatu yang mungkin menjadi masalah bagi orang-orang pada awalnya.

Manusia memiliki kecenderungan untuk ingin menyembunyikan sesuatu. Seberapa sering Anda mendengar orang tua Anda berkata "Urus saja urusanmu sendiri" atau "Simpan untuk dirimu sendiri" ketika Anda tumbuh dewasa? Sebagian besar dari kita tidak tumbuh dalam lingkungan yang mendorong kita untuk bersikap terbuka dan transparan—terutama tentang kegagalan. Tim pengembang tentu ingin menyembunyikan masalah hingga saat-saat terakhir. Pada masa Waterfall, tim akan menyembunyikan masalah dari manajer pengembangan, dengan harapan dapat memperbaikinya sebelum masalah tersebut terungkap. Scrum mengharuskan pekerjaan dilakukan secara terbuka sehingga masalah dapat ditangani sesegera mungkin.

Kakak saya melatih lini ofensif di sekolah menengah tempat kami berdua lulus. Saya ikut angkat beban bersama tim beberapa kali, tetapi saya tidak menganggap diri saya sebagai bagian dari tim atau apa pun. Itulah mengapa saya sangat gembira ketika kakak saya memberi saya kaus "ironman" tim. Setiap anggota tim yang berpartisipasi dalam latihan di luar musim mendapat satu. Di bagian depan tertulis "Viking Iron-men" di sekeliling gambar maskot tim

seorang Viking yang besar dan berotot memegang barbel yang terisi. Di bagian belakang terdapat 10 "perintah," yang pada dasarnya merupakan nilai-nilai inti dari program tersebut.

Saya sangat senang dengan kaus tersebut. Saya memakainya sepanjang waktu. Namun, saya tidak menceritakan hal ini kepada Anda karena saya sangat menyukai kaus tersebut. Saya memberi tahu Anda karena cara beberapa orang bereaksi terhadapnya. Saat melihat kemeja baru saya, salah satu saudara ipar saya berkata, "Saya tidak setuju dengan nomor enam". Nomor enam berkata, "Anda harus kalah untuk menang". Di permukaan, itu tidak masuk akal. Budaya kita mencintai pemenang, jadi ada premi yang sangat besar yang diberikan untuk menjadi benar. Ini hampir seperti pertunjukan permainan raksasa: "Saya akan selalu benar untuk 500!".

Karena itu, ada rasa takut yang nyata dan nyata untuk melakukan kesalahan. Jika kita berjalan dengan rasa takut untuk melakukan kesalahan, kita tidak akan pernah berkembang. Saya selalu mengatakan bahwa jika Anda menunjukkan kepada saya seseorang yang tidak pernah gagal, Anda sebenarnya menunjukkan kepada saya seseorang yang gagal belajar. Fokuslah pada hasil positif dari kegagalan. Tidak, tidak ada yang suka gagal. Saya lebih suka menganggap sesuatu yang tidak berhasil sebagai cara yang luar biasa untuk mempelajari apa yang berhasil dan tidak. Tim Scrum harus bersedia mencoba hal-hal baru bahkan jika itu berarti mereka akan gagal melakukannya.

Anda Harus Kalah untuk Menang

"Anda harus kalah untuk menang" adalah motto latihan untuk tim angkat beban. Program yang kita semua mulai bertahun-tahun lalu di garasi sangat sederhana secara konseptual, tetapi sangat sulit untuk diterapkan karena mengharuskan pengangkat beban untuk terus-menerus berusaha membuat rekor pribadi (PR) dalam sesuatu. Programnya tampak seperti ini:

Minggu 1: Lakukan set terbaik Anda sebanyak 15 repetisi.
Minggu 2: Lakukan set terbaik Anda sebanyak 5 repetisi.
Minggu 3: Lakukan set terbaik Anda sebanyak 3 repetisi.
Minggu 4: Lakukan set terbaik Anda sebanyak 1 repetisi.
Minggu 5: Istirahat.

Kemudian ulangi siklusnya.

Gambar 3.1 Ini adalah Susi Susanti Pebulu tangkis Wanita pertama di Indonesia yang menyabet gelar Olimpiade

Seperti yang saya katakan, ini sederhana tetapi sangat sulit karena Anda terus-menerus menguji diri sendiri. Ketika Anda gagal, ada dua pilihan. Tingkatkan repetisi yang gagal atau turunkan beban dan mulai lagi. Misalnya Anda mencoba bench press dengan beban 225 pon sebanyak 15 repetisi, dan Anda gagal pada repetisi ke-12. Anda dapat menaikkan beban, duduk tegak, dan menghitung sampai sepuluh (maksud saya "satu Mississippi, dua Mississippi di sini.") Anda perlu cukup pulih untuk memiliki kesempatan menyelesaikan set Anda, tetapi tidak sepenuhnya pulih.

Ketika Anda mencapai sepuluh, berbaringlah kembali di bangku dan lakukan yang terbaik untuk menyelesaikan lima repetisi. Ya, kami menambahkan satu. Tidak ada alasan teknis yang dapat saya pikirkan mengapa kami menambahkan repetisi. Kami melakukannya begitu saja. Pilihan Anda yang lain adalah menurunkan beban, katakanlah menjadi 220, dan mendapatkan 15 repetisi. Satu-satunya peringatan adalah bahwa bebannya harus lebih besar dari terakhir kali Anda menguji diri sendiri selama 15 repetisi. Bicara tentang peningkatan berkelanjutan! Seperti yang Anda lihat, pengangkat beban selalu ditantang.

Ada tekanan konstan untuk memberikan hasil. Sama seperti Sprint dalam Scrum. Budaya perintah dan kontrol memperbesar rasa takut untuk melakukan kesalahan atau kegagalan. Scrum menyarankan agar Anda benar-benar transparan sehingga Anda dapat memeriksa apa yang sering Anda lakukan. Jika tim gagal dalam sesuatu, tentukan alasannya dan sesuaikan dengan tepat. Pada hari terakhir setiap Sprint, tim Scrum harus mengadakan retrospeksi pada dasarnya rapat tim tempat tim dan master Scrum berkumpul, melihat bagaimana keadaan selama Sprint, dan kemudian mencari tahu cara untuk menjadi lebih baik.

Beroperasi secara transparan adalah hal yang bagus untuk dilakukan; namun, jika Anda tidak pernah melihat apa yang sedang terjadi dan menentukan cara untuk menjadi lebih baik, bersikap transparan tidak akan memberikan nilai tambah apa pun. Melakukan rapat retrospeksi secara teratur menciptakan, atau setidaknya membantu mewujudkan, budaya

perbaikan berkelanjutan. Dengan terus mencari cara untuk meningkatkan, Anda tidak dapat tidak menjadi lebih baik. Transparansi, inspeksi, dan adaptasi disebut sebagai "tiga pilar Scrum." Untuk menjadi hebat dalam suatu hal, Anda harus secara objektif meninjau kembali apa yang telah Anda lakukan. Pertahankan apa yang berhasil, dan buang apa yang tidak.

3.2 PERAN SCRUM

Pemilik Produk, Master Scrum, dan tim Scrum adalah tiga peran yang ditetapkan oleh kerangka kerja Scrum. Peran-peran ini adalah yang membuka kekuatan Scrum dan memungkinkan nilai untuk disampaikan kepada pelanggan dan pemangku kepentingan dengan cepat.

Pemilik Produk

Saya pernah mendengar peran Pemilik Produk (PO) dijelaskan dalam beberapa cara. Penjaga Backlog Produk. Pendukung pelanggan. Satu-satunya yang bisa dikekang. Satu hal yang pasti, peran PO penuh tantangan. PO adalah titik kontak antara tim Scrum, pelanggan, dan bisnis. Pemilik Produk berbicara dengan pelanggan dan mengumpulkan persyaratan. Mereka membangun backlog cerita pengguna yang digunakan tim Scrum untuk mengubah persyaratan tersebut menjadi perangkat lunak yang berfungsi. PO juga harus memaksimalkan laba atas investasi (ROI). Mereka harus terus berpikir untuk memuaskan bisnis dan pelanggan.

Setiap keputusan yang dibuat PO ditimbang berdasarkan dampaknya terhadap bisnis sekaligus memberikan nilai kepada pelanggan. Mereka memiliki wewenang untuk menerima atau menolak pekerjaan yang telah selesai, dan bahkan dapat mengubah arah proyek di akhir setiap Sprint. Ya, itu tanggung jawab yang besar. PO memiliki pengaruh terhadap kebahagiaan tim yang membangun perangkat lunak dan orang-orang yang menggunakannya.

Advokat Pelanggan

Pemilik Produk harus memiliki keahlian tentang cara pelanggan menggunakan perangkat lunak serta nilai yang mereka dapatkan darinya. Peran tersebut menjadikan mereka sebagai satu-satunya titik kontak antara tim dan semua pemangku kepentingan. Yang saya maksud dengan "pemangku kepentingan" adalah siapa saja yang tertarik dengan apa yang dihasilkan tim: para eksekutif ingin tahu berapa laba atas investasi; perwakilan dukungan ingin tahu cara men-debug fitur baru. Tim lain mungkin ingin tahu cara menghubungkan produk mereka dengan produk Anda.

Pemangku kepentingan sangat pandai menyampaikan permintaan mereka. Mungkin itu berlebihan. Mereka pandai memberi tahu siapa pun yang mau mendengarkan apa yang mereka inginkan. Yang tidak mereka sadari adalah bahwa yang mereka lakukan hanyalah mengalihkan perhatian orang-orang yang seharusnya fokus pada tugas yang sedang dikerjakan. Misalnya, saya punya teka-teki gambar yang rumit dan saya menawarkan Anda Rp.1.500.00 untuk menyusunnya bagi saya. Masalahnya adalah saya hanya memberi Anda waktu 3 jam untuk mengerjakannya.

Saya berani bertaruh bahwa Anda tidak akan dapat menyelesaikan tugas tersebut jika orang-orang terus menyela Anda dengan pertanyaan yang mengharuskan Anda mengalihkan fokus dari teka-teki tersebut. Dan pelanggan senang sekali berbicara dengan pengembang

tentang persyaratan mereka. Saya ingat menghadiri konferensi pengguna dan berpikir serius bahwa kurungan isolasi bukanlah hal yang buruk. Serius, saya membicarakan hal ini sejak saya meninggalkan hotel di pagi hari hingga tidur di malam hari. Kenyataannya adalah bahwa pelanggan sangat menginginkan fitur tertentu, dan tidak tahu cara mendapatkannya. Mereka membutuhkan cara untuk menyampaikan kebutuhan mereka dengan cara yang berarti.

Itulah sebabnya pemilik produk sangat penting. Antarmuka PO dengan pelanggan dan pemangku kepentingan ada dua. Berbicara dengan PO adalah cara pasti untuk memasukkan "permintaan" Anda ke dalam Backlog. Pelanggan tidak perlu mencoba memasukkan fungsionalitas melalui pintu belakang; mereka memiliki akses penuh ke pintu depan. Pemilik Produk. Bertemu dengan PO dan memastikan bahwa ada pemahaman tentang persyaratan dan urgensi kebutuhan adalah cara terbaik untuk memasukkan apa yang Anda inginkan ke dalam produk. Ini juga membantu agar tim tidak terganggu. Pemilik Produk adalah satu-satunya titik kontak untuk tim Scrum. Tim tidak lagi diganggu oleh pemangku kepentingan mana pun. Semuanya dilakukan melalui PO. Ini memungkinkan tim Scrum melakukan yang terbaik menciptakan perangkat lunak yang berharga.

Memaksimalkan ROI

Saya belum pernah mendengar anggaran TI pelanggan bertambah besar. Itulah sebabnya manifesto menyatakan "negosiasi kontrak yang berlebihan." Seorang CIO (atau siapa pun yang melakukan negosiasi) akan selalu mengatakan bahwa anggaran mereka menyusut dan mereka tidak dapat membayar apa yang diminta untuk produk tersebut. Itulah sebabnya kita perlu memproduksi perangkat lunak yang tidak dapat diabaikan oleh pelanggan. Hal-hal yang harus mereka miliki. Pemilik Produk perlu bermitra dengan pelanggan untuk mendapatkan pemahaman mendalam tentang cara mereka menggunakan perangkat lunak dan apa yang mereka anggap berharga.

Pengembang tidak bekerja di pusat data; mereka menulis kode. Mereka mungkin pernah bekerja di lingkungan yang saya sebut sebagai lingkungan pelanggan, tetapi sekarang peran mereka berbeda. Mereka ahli dalam apa yang mereka lakukan, tetapi mereka perlu memahami bagaimana pelanggan menggunakan perangkat lunak untuk menciptakan sesuatu yang berharga. Seperti yang saya katakan sebelumnya, pelanggan menggunakan perangkat lunak kami dengan cara yang tidak pernah kami pikirkan untuk memecahkan masalah yang tidak pernah kami dengar.

Saya ingat belajar tentang seorang pelanggan yang menggunakan produk yang sedang saya kerjakan saat itu untuk memantau lemari es. Saya bekerja di tim yang mendukung produk otomasi tempel. Produk tersebut memiliki kemampuan untuk mendapatkan akses ke berbagai host dan mengambil tindakan terhadap kejadian. Host yang dapat dipantau termasuk mainframe, server Windows, dan IBM iSeries, tetapi tidak termasuk lemari es. Pelanggan memiliki kebutuhan dan menemukan cara untuk melakukannya, dan memperoleh nilai sebagai hasilnya.

Suara pelanggan harus datang dari, ya pelanggan. Tugas pemilik produk adalah mengambil suara pelanggan itu dan membuat tim tetap fokus padanya. PO harus dapat berkomunikasi secara efektif mereka adalah sumber informasi bagi tim Scrum. Jika ada

pertanyaan tentang apa yang sedang dibangun, bagaimana persyaratan tercermin dalam cerita, atau bahkan bagaimana pelanggan akan menggunakan apa yang sedang dibangun, tim akan mengharapkan PO memiliki jawabannya. PO harus dapat mengisi peran pelanggan saat tim memiliki pertanyaan.

3.3 PENGUASA BACKLOG

Pemilik Produk bertanggung jawab atas Backlog Produk. Anggap Backlog sebagai daftar "yang harus dilakukan". Setiap hari, saya membuat daftar yang harus dilakukan untuk menyelesaikan sesuatu. Urutan item dalam daftar saya berubah seiring dengan perubahan prioritas sepanjang hari (terutama karena hari saya hampir tidak pernah berjalan sesuai harapan), tetapi hal itu tidak terlalu memengaruhi pekerjaan saya. Saat saya menyelesaikan item, saya kembali dan menghapus item berikutnya dari daftar. Dengan risiko dianggap terlalu sederhana, begitulah cara kerja Backlog. Daftar persyaratan diprioritaskan dan disempurnakan, lalu akhirnya diubah menjadi hasil akhir.

PO mungkin tidak menulis setiap cerita dalam Backlog, tetapi mereka bertanggung jawab atas semua hal dalam Backlog. Mungkin tidak realistis mengharapkan mereka untuk menulis setiap cerita secara fisik. Anggota tim dapat membantu; namun, PO harus sangat menyadari semua hal yang masuk ke dalam Backlog. Dengan menyadari setiap cerita dalam Backlog dan terus mengikuti perkembangan bisnis, PO dapat memprioritaskan Backlog secara efektif dan membuat semua orang mengetahui apa yang akan terjadi selanjutnya. Pemilik Produk juga merupakan orang yang berwenang untuk menerima atau menolak cerita pengguna saat tim telah selesai mengerjakannya. Ia memutuskan apakah cerita tersebut memenuhi kriteria penerimaannya atau tidak.

Kriteria penerimaan, seperti yang dijelaskan dalam Bab 2, adalah daftar kondisi yang harus dipenuhi sebelum PO menerima cerita tersebut. Daftar tersebut merupakan sesuatu yang disetujui oleh PO dan tim Scrum, dan memastikan bahwa tujuan cerita terpenuhi. Dokumen kriteria penerimaan yang ditulis dengan baik akan memfokuskan tim pada hasil yang diinginkan dari cerita tersebut. Dokumen tersebut merupakan bentuk kontrak kerja antara tim Scrum dan PO. Daftar kriteria penerimaan dibuat oleh PO, kemudian disempurnakan dan disetujui oleh tim. Saya pernah mendengar bahwa Pemilik Produk bertanggung jawab untuk menerima cerita ke dalam Backlog dan menerima cerita saat tim menyelesaikannya. Semacam liputan dari awal hingga akhir.

PO bertanggung jawab penuh atas proyek tersebut, jadi mereka harus terlibat di dalamnya. Permintaan akan selalu lebih besar daripada kapasitas. Tugas PO adalah mengajukan pertanyaan yang tepat untuk memastikan bahwa Backlog diprioritaskan dengan tepat. Backlog yang diprioritaskan dengan tepat akan menghasilkan tim yang menghasilkan tingkat nilai yang tepat yang akan memuaskan sebanyak mungkin pemangku kepentingan. Saya akan membahasnya lebih rinci nanti di buku ini. Sebagai rekapitulasi, Pemilik Produk bertanggung jawab untuk membangun Backlog cerita pengguna yang diprioritaskan yang digunakan tim untuk menciptakan nilai.

Cerita-cerita tersebut menciptakan tujuan yang jelas untuk dicapai oleh tim. Jika tim

Scrum memiliki pertanyaan tentang cerita pengguna, PO-lah yang memberikan jawabannya. PO-lah yang menyempurnakan cerita dengan bantuan tim Scrum dan membawanya ke titik di mana cerita-cerita tersebut dapat ditarik ke dalam SprintSprint. PO adalah titik kontak bagi pelanggan dan pemangku kepentingan. Karena itu, tim Scrum bebas dari gangguan dan dapat fokus untuk memberikan nilai pada setiap SprintSprint.

Tim Scrum

Tim Scrum sedikit berbeda dari yang biasa Anda lihat. Karena didefinisikan sebagai tim yang mengatur diri sendiri dan lintas fungsi, tidak seharusnya ada peran yang ditentukan. Tidak seharusnya ada. Kenyataannya adalah tidak ada peran yang ditentukan dalam tim Scrum, tetapi kebanyakan orang datang ke tim dengan peran yang ditentukan. Tim Scrum harus dapat menulis, menguji, mendokumentasikan, dan mengirimkan perangkat lunak yang berfungsi setiap Sprint. Peran yang ditentukan membuat hal ini sangat sulit.

Lakukan Apa yang Diperlukan untuk Mencapai Tujuan

Tim bola basket memiliki lima pemain di lapangan. Setiap pemain memiliki peran, tetapi akan melakukan apa pun yang diperlukan untuk memenangkan permainan. Ketika saya melatih (Ya, saya juga melatih tim bola basket putri saya), hal yang paling saya sesalkan adalah rebounding. Rebounding adalah saat tembakan meleset, dan Anda maju dan mengambil bola. Tim yang "membersihkan kaca" memiliki keuntungan tersendiri. Saya akan memberi tahu para pemain saya untuk membaca skor pertandingan basket, sekolah menengah, perguruan tinggi, atau NBA.

Tim yang memiliki rebound terbanyak biasanya adalah pemenangnya. Rebound adalah pekerjaan yang sulit. Anda perlu mendapatkan posisi di depan lawan, melompat lebih tinggi dari mereka, dan mendapatkan bola. Itu benar-benar pekerjaan yang sulit, dan tentu saja tidak ada kemewahan di dalamnya. Dapatkah Anda bayangkan bagaimana jadinya jika point guard menolak untuk mengejar rebound karena itu adalah tugas center? Pengembang menulis kode, penguji menguji, penulis dokumen menulis, dan pemilik produk mengumpulkan persyaratan. Tekanan yang dibawa oleh iterasi mengharuskan tim untuk sedikit "mengaburkan batasan".

Pengembang mungkin perlu melakukan beberapa pengujian. Penguji mungkin perlu menulis dokumentasi. Tim mungkin perlu membantu PO mengumpulkan persyaratan dengan berbicara kepada pelanggan. Yang tidak Anda inginkan adalah penguji memiliki lebih banyak pekerjaan daripada yang dapat mereka tangani karena pengembang terus mengeluarkan kode. Tim perlu fokus dan menyelesaikannya. Pelanggan tidak membeli kode mereka membeli fitur. Mereka berharap perangkat lunak bebas cacat dan terdokumentasi dengan baik.

Tim Scrum perlu memiliki keterampilan yang diperlukan untuk menyelesaikan pekerjaan serta wewenang untuk melakukannya. Tidak ada peran manajer pengembangan dalam Scrum. Tim memiliki wewenang untuk menentukan cara terbaik untuk memberikan nilai. Mereka bertanggung jawab atas lingkungan, kecepatan, dan bahkan beban kerja mereka sendiri. Sebelum saya membahas tipe orang yang mungkin Anda temukan di tim Scrum, ada baiknya untuk melihat nilai-nilai Scrum. Berikut ini adalah nilai-nilai tim yang terorganisasi sendiri:

- Komitmen: Tim Scrum tidak diberi tahu apa yang harus dilakukan setiap iterasi. Mereka

berkomitmen pada apa yang mereka pikir dapat mereka lakukan, dan mereka berkomitmen untuk memenuhi DoD dan kriteria penerimaan apa pun. Ini memberi tim rasa kepemilikan atas pekerjaan mereka.

- Rasa hormat: Ini adalah hal yang penting. Tim perlu menghormati masukan Pemilik Produk tentang pekerjaan apa yang harus dilakukan dan dalam urutan apa. PO harus cukup menghormati tim untuk percaya bahwa mereka akan memenuhi komitmen mereka.
- Keberanian: Scrum Master perlu memiliki keberanian untuk menentang manajemen dan pemangku kepentingan sambil melindungi fokus tim. Tim perlu keberanian untuk menolak ketika PO menginginkan mereka untuk berkomitmen berlebihan selama Sprint, dan PO perlu memiliki keberanian untuk memberi tahu pemangku kepentingan tentang prioritas item dalam Backlog.
- Keterbukaan: Anda akan bosan mendengar saya berbicara tentang transparansi setelah Anda selesai membaca buku ini. Keterbukaan dapat membuat beberapa orang merasa tidak nyaman, tetapi itu benar-benar salah satu nilai inti Agile. Terlibatlah sepenuhnya dan terbuka jangan pernah menyembunyikan apa pun dari bisnis atau pemangku kepentingan.
- Fokus: Tim Scrum harus benar-benar fokus untuk menyelesaikan sesuatu. Lakukan multitugas sesedikit mungkin dan fokuslah untuk mencapai tujuan Sprint. Scrum Master harus memastikan bahwa semua hambatan telah disingkirkan dan tim terlindungi dari pengaruh eksternal.

Jadi, bagaimana peran tradisional cocok dengan tim lintas fungsi? Mari kita lihat setiap peran, dan bagaimana peran tersebut berubah dalam Scrum.

Insinyur Jaminan Kualitas

Oh ya, orang QA yang mungkin merupakan orang yang paling dibenci dalam proses pengembangan. Jangan salah paham; orang QA adalah kelompok yang cukup baik di masa Waterfall. Namun, hubungan mereka dengan pengembangan paling banter bersifat permusuhan. Itu benar-benar badai yang sempurna dari kegagalan yang tak terelakkan yang dipadukan dengan harapan yang tidak realistis. Ketika sebuah tim mengerjakan Waterfall, tidak ada yang lebih penting daripada merilisnya tepat waktu. Keberhasilan atau kegagalan dinilai berdasarkan tanggal yang ditentukan. Dan siapa yang biasanya menghalangi tercapainya tanggal GA yang sangat penting itu? Teknisi QA.

Bukan karena staf QA memiliki keinginan untuk bekerja keras atau karena mereka menetapkan standar kualitas pengembangan yang mustahil. Masalahnya adalah cara Waterfall mendikte pekerjaan. Waterfall adalah kerangka kerja yang berurutan dan tertutup. Anggap saja ini sebagai serangkaian acara di mana Anda tidak dapat melanjutkan ke acara berikutnya hingga Anda menyelesaikan acara yang sedang Anda ikuti. Teknisi QA memiliki tanggung jawab yang tidak menguntungkan karena menjadi garis pertahanan terakhir dalam hal kualitas produk, yang masuk akal bagaimanapun juga, mereka semua peduli dengan kualitas... benar? Jangan terburu-buru, kawan.

Masalahnya adalah dalam metodologi Waterfall, teknisi QA mendapatkan kode

terakhir, setelah semua pengembangan selesai. Mereka adalah yang terakhir dalam antrean. Mereka dianggap sebagai orang-orang yang menghalangi pengembang memenuhi tenggat waktu yang sangat penting. Masalah besar lainnya adalah staf QA tidak diberi cukup waktu untuk menyelesaikan semua pengujian. Itulah sifat Waterfall. Anda harus memenuhi tenggat waktu dengan segala cara, dan pengembangan memakan waktu lebih lama dari yang diharapkan. Karena QA berada di urutan terakhir, hal itu menjadi kendala. Teknisi QA sering diminta untuk melakukan hal yang bertentangan dengan sifat mereka. Mengabaikan masalah yang mereka temukan.

Hal ini membuat mereka merasa seperti korban. Orang-orang yang selalu harus menyampaikan berita buruk. Dalam Agile, teknisi QA bukan lagi korban. Mereka adalah pemimpin. Mereka memiliki kualitas dalam lingkungan di mana nol cacat adalah kebijakan, bukan target. Pengujian harus dilakukan sedini dan sesering mungkin. Tim Agile diharuskan untuk merilis tanpa cacat. Ini berarti persis seperti yang terlihat: QA harus ditanggapi dengan serius. Yang penting adalah merilis produk dengan kualitas terbaik. Insinyur QA memiliki pengetahuan paling banyak tentang pengujian dalam tim, jadi tugas mereka adalah memastikan bahwa pekerjaan QA dilakukan sedini dan sesering mungkin.

Dalam tim lintas fungsi, siapa pun dapat melakukan pekerjaan QA, jadi insinyur QA perlu berkonsultasi dengan anggota tim lainnya tentang pengujian QA, membantu penulisan otomatisasi pengujian, dan membantu PO dengan kriteria penerimaan. Semakin awal pengujian dimulai dalam proses, semakin besar risiko yang dihilangkan dari proses tersebut. Ketika saya pertama kali mulai bekerja dengan tim, kami melakukan apa yang saya sebut sebagai "Agile basah". Kami pada dasarnya membagi pekerjaan Waterfall menjadi Sprint, dengan banyak rapat yang sangat canggung.

Itu sangat tidak menyenangkan bagi para penguji, karena staf pengembangan akan membuat kode dan membuat kode dan membuat kode di awal Sprint, lalu memberikan banyak pekerjaan kepada para penguji di akhir Sprint. Itu seperti Waterfall dalam kotak waktu yang lebih kecil. Pengujian harus dimulai sedini mungkin. Pengembangan yang digerakkan oleh pengujian mengharuskan pengujian ditulis terlebih dahulu. Anda menjalankan pengujian, mencatat kegagalannya, lalu menulis kode untuk memastikan bahwa pengujian berhasil.

3.4 PERAN PENGEMBANG DAN SCRUM MASTER DALAM TIM AGILE

“Kapan saya mulai menulis kode?”. Saya bekerja dengan tim Scrum pertama saya, dan saya akui saya sangat kacau. Seorang pengembang mengatakan ini selama rapat standup yang tidak berjalan dengan baik. Saya punya pepatah kecil yang suka saya ulangi kepada tim yang saya latih sesekali. Peternak sapi perah memerah susu sapi, tukang cuci jendela membersihkan jendela, dan tim Scrum menghasilkan nilai. Perhatikan bahwa saya tidak mengatakan baris kode. Tim Scrum perlu keluar dari cara berpikir seperti itu. Agile mendikte bahwa tim tidak menghasilkan kompleksitas. Agile menuntut kesederhanaan.

Pengembang biasanya adalah orang terpintar di ruangan itu. Saya memiliki hak istimewa untuk bekerja dengan beberapa pengembang yang luar biasa selama bertahun-tahun. Orang-orang yang dapat melihat masalah dan menciptakan solusi dari ketiadaan.

Semua orang mengagumi seseorang seperti itu; namun, bahkan seorang superstar perlu menyadari bahwa mereka tidak dapat melakukannya sendiri.

Pemain basket dengan keterampilan ofensif paling hebat tidak dapat memenangkan permainan sendiri tidak peduli seberapa keras mereka mencoba. Coba pikirkan, Michael Jordan tidak pernah memenangkan kejuaraan hingga Scottie Pippen ditukar ke Chicago Bulls. Di dunia yang baru, berani, dan gesit ini, bukan tentang menulis kode yang hebat sendiri. Melainkan tentang membuat orang-orang di sekitar Anda menjadi lebih baik. Pengembang harus bekerja sama dan membimbing.

Kenyataannya adalah bahwa tim hanya secerdas orang yang paling pendiam di ruangan itu. Alih-alih mendominasi, staf pengembangan harus memfasilitasi diskusi yang kuat di antara semua anggota tim. Dengan cara ini, kebijaksanaan tim akan muncul. Kolaborasi bukanlah pemikiran kelompok, melainkan konflik yang sehat. Setiap orang dalam tim harus merasa nyaman mengemukakan sudut pandang mereka dan menemukan kesamaan tentang apa yang sedang dibahas. Jika bukan kesamaan, setidaknya capailah titik di mana setiap orang dapat menerima keputusan tersebut. Yang tidak Anda inginkan adalah anggota tim hanya mengikuti apa yang terjadi, tetapi keluar dari rapat dengan harapan gagal. Nilai tim Scrum bagi organisasi bukanlah membangun sesuatu. Fokusnya bukan lagi pada mengetik baris kode; melainkan pada memberikan nilai.

Scrum Master

Kami menyelenggarakan acara di kantor untuk sekelompok anak-anak dari Yayasan Best of the Batch, yang didirikan oleh mantan quarterback Detroit Lion dan Pittsburgh Steelers Charlie Batch. Yayasan ini menyediakan kesempatan bagi kaum muda yang mengalami kesulitan keuangan dan keluarga mereka untuk memberikan upaya terbaik mereka dalam semua hal yang mereka lakukan sepanjang hidup mereka. Kami menunjukkan nilai Teknologi Informasi, dan mengapa mereka harus mempertimbangkan karier di bidang TI.

Di penghujung acara, saya menunjukkan sebagian bakat yang hanya dimiliki oleh pria dengan hobi seperti saya. Saya merobek pelat nomor menjadi dua, menggulung wajan penggorengan, dan membengkokkan sepotong baja canai dingin di mulut saya, seperti yang ditunjukkan pada Gambar 3.2. Saya senang melakukan hal-hal seperti ini untuk anak-anak, dan mereka pun kagum dengan apa yang dilakukan oleh "orang kuat yang ahli". Saya rasa Anda tidak melihat orang menggulung wajan penggorengan setiap hari, apalagi di laboratorium pengembangan perangkat lunak.

Setelah semua perayaan itu, salah satu anak bertanya kepada saya apa peran saya di CA. Ketika saya menjawab bahwa saya adalah Scrum Master, raut wajahnya sungguh tak ternilai. Setelah beberapa detik berpikir keras, dia bertanya apakah saya petugas kebersihan. Ini bukan kejadian yang terisolasi. Ketika saya memberi tahu orang-orang apa yang saya lakukan, biasanya mereka akan menanggapinya dengan tatapan bingung, cekikikan, atau permohonan klarifikasi.

Gambar 3.2 Membengkokkan Batang Baja Canai Dingin Di Mulut

Saya ingat saat saya sedang sarapan di restoran cepat saji bersama teman saya Russ Clear. Russ adalah orang yang mengajari saya cara melakukan aksi orang kuat. Dia adalah mantan anggota Hell's Angels yang menjadi orang kuat evangelis dan melakukan demonstrasi di gereja-gereja dan sekolah-sekolah. Russ adalah pria bertubuh besar dengan kepala yang dicukur, bertato, dan berotot. Dia bukan tipe orang yang hanya pamer dan tidak bisa berbuat apa-apa. Saya sendiri menyaksikan dia memecahkan 10 bantalan semen yang ditumpuk dengan kepalanya. Russ adalah orang yang mengajari saya cara melakukan aksi orang kuat seperti yang ditunjukkan pada Gambar 3.3.

Gambar 3.3 Russ Mengajari Saya Cara Merusak Wajan Penggorengan Yang Masih Bagus

Dan berat badan saya saat itu mencapai 300 pon. Saya yakin kami cukup menarik perhatian, tetapi kami tidak menyadari semua orang yang menatap kami. Akhirnya seorang pria menghampiri kami dan bertanya, "Apakah kalian pegulat?". Russ menatapnya dengan pandangan mengancam dan berkata, "Tidak." Dia bukan orang yang banyak bicara saat merasa kesal, dan dikira sebagai pegulat profesional membuatnya kesal. “Jadi, kalian ini apa?” Saya rasa orang ini tidak mengerti maksudnya. “Baiklah, saya seorang Scrum Master,” kata saya. Itu mengakhiri percakapan.

Memberikan gambaran yang jelas tentang apa yang sebenarnya dilakukan oleh seorang Scrum Master memang sulit; namun, Scrum Master sangat penting untuk keberhasilan tim Scrum. Saya akan berusaha sebaik mungkin untuk menjelaskan perannya di sini. Seorang Scrum Master adalah pemimpin pelayan tim Scrum. Perhatikan bahwa saya tidak mengatakan manajer. Seorang Scrum Master bukanlah manajer pengembangan dengan nama baru yang keren. Dalam Scrum, tim seharusnya mengelola diri sendiri. Dengan kata lain, mereka mengelola diri mereka sendiri. Saya tahu, itu agak berlebihan, tetapi saya perlu menekankan hal itu.

Scrum Master ada untuk melakukan apa pun yang diperlukan untuk membuat tim Scrum berhasil sampai pada titik tertentu. Menjadi seorang pemimpin pelayan tidak berarti Anda meremehkan diri sendiri. Itu berarti Anda meremehkan diri sendiri. Dalam karier saya, saya menghabiskan banyak tahun sebagai insinyur pendukung. Dalam istilah awam, saya memperbaiki bug. Saya punya kecenderungan alami untuk ingin terjun langsung dan memperbaiki berbagai hal. Hal itu membuat istri saya kesal, dan saya harus melawan diri sendiri untuk tidak melakukannya dengan tim tempat saya bekerja. Mengapa? Karena jika saya terus memperbaiki masalah untuk mereka, mereka akan selalu meminta saya untuk memperbaikinya. Scrum Master perlu membantu tim untuk menemukan solusi sendiri bukan melakukannya untuk mereka.

Gambar 3.4 Hal Ini Dijamin Akan Membuat Orang Senang. Beberapa Orang Akan Mengeluh Tentang Kalori Atau Bahwa Donat Tidak Sehat; Namun, Hal Itu Akan Hilang Pada Akhir Rapat.

Untuk lebih jelasnya, bagian dari pekerjaan Scrum Master adalah menyingkirkan hambatan. Bukan membantu pekerjaan pengodean atau desain, atau hal-hal semacam itu. Meski menggoda, Anda tidak ada di sana untuk membantu tim melakukan pekerjaan mereka. Anda ada di sana untuk memungkinkan mereka melakukan pekerjaan mereka. Salah satu masalah dengan menjadi pemimpin pelayan adalah mudah untuk menjadi pemimpin. Seorang pelayan, tidak semudah itu. Menjadi pelayan adalah pola pikir, keputusan sadar untuk mengutamakan kebutuhan tim di atas kebutuhan Anda sendiri. Tim adalah yang utama. Dan Scrum Master membeli donat, seperti pada Gambar 3.4. Begitulah adanya.

3.5 PERAN SCRUM MASTER DALAM MENERAPKAN AGILE DAN MELATIH TIM

Seorang Scrum Master adalah penjaga api Agile. Seorang Scrum Master perlu memastikan bahwa tim menerapkan kerangka kerja Agile dengan benar—khususnya Scrum. Sangat penting bagi seorang Scrum Master untuk menjadi orang yang gila tentang Agile. Oke, mungkin sebagian besar Scrum Master tidak akan sampai ke tingkat itu, tetapi penting untuk memastikan bahwa tim tidak "menjalankan Agile". Mereka perlu menjalankan Manifesto Agile setiap hari. Ini mungkin mengharuskan Scrum Master untuk dapat melatih orang-orang tentang cara menerapkan prinsip-prinsip Agile.

Ketika saya mengatakan melatih, saya tidak bermaksud bahwa Anda berteriak dan melempar kursi. Maksud saya adalah mengajar, mendengarkan, dan mengajukan pertanyaan yang kuat, dan membantu tim membuat keputusan tanpa membuat keputusan untuk mereka. Salah satu hal yang selalu dikatakan istri saya adalah, "Kamu tidak mendengarkan saya". Dia benar. Faktanya, kebanyakan dari kita tidak mendengarkan dengan efektif, bahkan ketika kita berusaha sangat keras.

Masalahnya adalah, bahkan ketika saya mendengarkan seseorang dengan saksama, saya masih memiliki percakapan internal di kepala saya. Saya mencoba memecahkan masalah Anda, atau memikirkan sesuatu yang cerdas untuk dikatakan, atau memikirkan makan siang atau semacamnya. Kenyataannya, saya pikir saya mendengarkan, tetapi saya melakukan segalanya kecuali berfokus pada apa yang dikatakan.

Menyadari hal ini, tingkat mendengarkan berikutnya adalah ketika saya menyadari apa yang terjadi di dalam kepala saya dan menyadari kapan saya tidak fokus pada percakapan. Menghentikan diri saya dari "menyimpang" memungkinkan terjadinya komunikasi yang sebenarnya. Hal yang keren adalah ketika Anda menjadi ahli dalam hal ini, Anda mulai melihat gambaran utuh. Hal-hal seperti bahasa tubuh dan ekspresi wajah. Hal ini memungkinkan saya menggunakan segalanya untuk memahami apa yang coba dikomunikasikan oleh orang yang berbicara kepada saya.

Teknik pembinaan yang ampuh adalah penggunaan pertanyaan-pertanyaan yang ampuh. Pertanyaan yang kuat langsung ke intinya. Pertanyaan itu memberi orang yang dilatih kesempatan untuk mengklarifikasi, atau menemukan bahwa mereka tahu solusinya. Kembali ke masa-masa saya melatih bola basket. Entah mengapa, tim saya terpesona dengan sudut kanan lapangan basket. Setiap kali terjadi turnover atau fast break (setiap kali kami tidak menjalankan permainan yang ditetapkan), gadis yang membawa bola akan menuju sudut

kanan lapangan. Saya biasa bercanda bahwa pasti ada anak anjing di sana, tetapi ini bukan hal yang bisa ditertawakan.

Saat berada di sudut itu, garis dasar dan garis samping bertindak sebagai dua pemain bertahan tambahan. Hal ini memudahkan tim lawan untuk menjebak pemain yang membawa bola. Saya biasa merasa sangat frustrasi setelah mengatakan "jangan pergi ke sudut kanan" sekitar satu miliar kali selama latihan. Saya seharusnya mencoba pertanyaan langsung. Pertanyaannya akan terlihat seperti ini.

"Jorden," kata saya (ya, saya mengejek putri saya), "Menurutmu, apakah membawa bola ke sudut kanan lapangan akan memudahkan untuk mencetak gol?" Mengenal putri saya, dia akan menunjukkan ekspresi yang sangat serius di wajahnya dan berkata, "Saya rasa tidak..." "Mengapa?"

"Karena kita tidak bisa memasukkan bola ke dalam keranjang."

"Baiklah," kata saya. "Bagaimana jadinya jika kita bisa memasukkan bola dengan mudah ke dalam keranjang?"

"Saya rasa kita harus memasukkan bola ke dalam area pertahanan." "Menurutmu, apa cara termudah untuk melakukannya?" "Menendang bola ke bagian atas ring?" tanyanya. "Ya, itu akan lebih baik daripada di sudut ring."

"Menggunakan pertanyaan yang kuat akan menarik jawaban dari orang lain, jadi alih-alih memberi tahu mereka, Anda membantu mereka mencari tahu sendiri. Hal ini memberi orang yang sedang dilatih rasa kepemilikan yang biasanya membuat mereka lebih mudah menerapkan ide tersebut.

Selama menjadi Scrum Master, saya harus menjelaskan berulang kali mengapa mengadakan rapat standup harian merupakan ide yang bagus. Akhirnya, saya bosan mengulang-ulang dan bertanya, "Apa yang membuat rapat standup itu berharga?". Orang yang saya ajak bicara berdiri di sana selama beberapa menit dengan ekspresi bingung di wajahnya dan berkata, "Apa maksudmu?" Saya berkata, "Jika Anda tidak ingin menghadiri rapat standup harian, Anda pasti tidak merasa ada nilainya. Bagaimana kita bisa membuat rapat itu berharga bagi Anda?".

Itu berujung pada percakapan yang sehat yang membuat tim membuat beberapa perubahan yang secara langsung mengarah pada peningkatan dinamika tim. Salah satu ucapan saya adalah, "Jadilah Manifesto Agile yang berwajah tegas." Tidak semua orang akan memahami hal-hal Agile ini, apalagi menerimanya. Scrum Master perlu menjadi pelatih yang dengan lembut menuntun semua orang menuju kelincahan yang lebih baik. Salah satu hal yang paling saya jengkelkan adalah ketika seseorang mengatakan kita "menjalankan Agile." Menurut saya, Anda tidak "melakukan" Agile.

Seorang gelandang tidak muncul di latihan dan berkata, "Saya akan menjadi agile hari ini." Ia terus-menerus berusaha untuk itu. Ia perlu mengembangkan ketangkasannya dan terus-menerus berusaha untuk meningkatkannya karena seperti yang biasa dikatakan oleh pelatih hoki hebat Herb Brooks, "kaki memberi makan serigala". Jika Anda tidak dapat menggerakkan kaki, Anda tidak akan banyak bermain. Anda tidak membalik tombol dan langsung menjadi Agile. Itu lebih seperti memutar tombol. Sedikit demi sedikit, semakin

banyak setiap hari. Scrum Master ada di sana untuk memimpin transformasi Agile dan untuk fokus pada keberhasilan (yang saya sebut sebagai suar) yang menerangi jalan menuju ketangkasan.

Seorang Scrum Master Harus Menjadi Sumber Energi dan Motivasi

Saya suka menggambarkan diri saya sebagai "sangat positif." Tipe orang yang melihat gelas setengah penuh. Kepositifan secara langsung mengarah pada kinerja; itu adalah fakta yang terbukti. Apa, Anda tidak percaya kepada saya? Anggaplah Anda sedang membawa semangkuk penuh sup panas ke seberang ruangan. Jika Anda terus berkata kepada diri sendiri, "Saya akan menumpahkan ini!", kemungkinan besar Anda akan menumpahkannya. Sebut saja ini ramalan yang terwujud dengan sendirinya atau sekadar keberuntungan; sikap Anda akan menentukan hasilnya.

Saya tidak yakin dari mana asal cerita ini. Sejujurnya, saya pertama kali mendengarnya dari pendeta saya, jadi saya akan mengakuinya. Ada sepasang anak laki-laki kembar. Secara fisik, mereka normal dan sehat, tetapi ibu mereka khawatir bahwa mereka mengembangkan kepribadian yang sangat berbeda, karena salah satu anak laki-laki sangat optimis tentang segala hal dan saudara laki-lakinya tampak sangat pesimis. Perilaku ini sampai pada titik di mana ibu anak laki-laki itu membawa mereka ke psikiater untuk melihat apakah ada yang bisa dilakukan untuk mereka.

Setelah mengamati mereka beberapa saat, psikiater mencoba bekerja dengan anak-anak laki-laki itu. Ia mulai dengan anak yang lebih pesimis. Ia menggandeng tangannya dan menunjukkan sebuah ruangan yang penuh dengan mainan baru hingga ke langit-langit. Ia mengatakan kepadanya bahwa semua barang di ruangan itu miliknya. Alih-alih melompat kegirangan, anak laki-laki itu malah berlutut dan mulai menangis.

"Ada apa?" kata psikiater. "Apakah kamu tidak suka mainan-mainan itu?"

"Oh, Tuan, mainan-mainan itu tampak bagus dan saya ingin sekali memainkannya," kata anak laki-laki itu di sela-sela isak tangisnya. "Tetapi saya mungkin akan merusaknya saja."

Psikiater itu kemudian membawa anak laki-laki yang optimis itu ke sebuah ruangan yang penuh dengan kotoran kuda hingga ke langit-langit. Yang mengejutkan semua orang, anak laki-laki itu berlari ke dalam ruangan dan mulai dengan panik menggali kotoran itu dengan tangannya.

"Apa yang kamu lakukan?" teriak psikiater itu...

Anak laki-laki itu menoleh ke arahnya dan berkata, "Tuan, pasti ada kuda poni di sini!"

Scrum Master harus menjadi kekuatan positif untuk perubahan. Teruslah mencari kuda poni itu.

Peran Scrum Master Dalam Memfasilitasi Tim

Scrum Master memfasilitasi dan menjalankan upacara Agile Scrum. Tentu saja, Scrum

Master menjadwalkan rapat; namun, idenya bukan hanya untuk mengadakan rapat. Segala hal yang mungkin harus dilakukan untuk membuat rapat bermakna dan efektif. Pikirkanlah. Rapat itu mahal. Lihat saja sekeliling ruangan selama rapat Anda berikutnya. Semua orang itu dibayar untuk hadir. Mari kita coba untuk tidak membuang-buang waktu. Untuk mencapai keberhasilan, rapat harus terfokus dan tidak menjadi mangsa "jejak kelinci" yang ditakuti.

Keluhan terbesar yang saya dengar tentang Scrum, terutama di awal, adalah tentang semua rapat. Saya akui bahwa jika Scrum Master tidak menjaga agar rapat tetap terfokus, rapat dapat menjadi, yah kacau balau. Idenya adalah untuk membuat rapat menjadi berharga. Perhatikan bahwa saya tidak mengatakan bahwa Scrum Master mengelola rapat. Mereka memastikan bahwa rapat berlangsung dan dilaksanakan dengan benar.

Untuk mencapai tujuan ini, Scrum Master harus membuat perjanjian kerja dengan tim. Saya akan membahasnya lebih lanjut di buku ini, tetapi berikut ini contoh yang bagus. Sebagai fasilitator, Scrum Master harus mendorong tim untuk mendefinisikan apa itu konsensus. Saya suka hal seperti ini: “Saya bisa menerimanya dan mendukungnya”. Anda tidak harus menyukainya, atau bahkan sangat menyukainya. Ketika tim mengambil keputusan, saya berharap bahwa ketika semua orang meninggalkan ruangan, mereka akan mendukungnya. Kita semua pernah mengikuti rapat yang buruk. Di mana Anda duduk di ruangan dan tidak ada yang selesai. Di mana orang-orang saling berteriak atau di mana satu orang mendominasi ruangan atau lebih buruk lagi, di mana semua orang bersikap hati-hati dan takut untuk berbicara.

Seorang fasilitator melawan rapat jenis itu. Dengan menggunakan teknik fasilitasi, seorang Scrum Master dapat mengikuti rapat yang penuh semangat. Di mana semua orang bersedia berpartisipasi dan keputusan dibuat. Di mana semua orang setuju bahwa rapat itu bernilai. Tujuan sebenarnya adalah untuk memunculkan kebijaksanaan tim. Seorang fasilitator menciptakan lingkungan yang aman di mana komunikasi tatap muka yang sesungguhnya dapat terjadi. Seorang Scrum Master menyingkirkan hambatan. Dalam standup harian, seorang Scrum Master harus mendorong tim untuk menjawab pertanyaan mereka:

- Apa yang Anda Lakukan Kemarin?
- Apa yang Anda Rencanakan untuk Dilakukan Hari Ini?
- Apakah Ada yang Menghambat Anda?

Ngomong-ngomong, Anda benar-benar harus meminta semua orang untuk benar-benar berdiri selama rapat standup. Oksigen yang masuk ke otak Anda sepuluh persen lebih banyak saat Anda berdiri. Seorang Scrum Master akan melakukan apa pun untuk membuat tim Scrum sukses. Anggota tim Scrum hanya perlu memikirkan tentang menghasilkan nilai. Jika komputer seseorang mati, atau mereka butuh bantuan dengan departemen hukum, Scrum Master harus turun tangan dan menyelesaikannya. Itu tidak berarti bahwa Scrum Master harus mengurus semuanya sendiri.

Mereka memiliki masalah dan jika perlu melibatkan orang lain. Scrum Master harus mendorong kerja sama yang erat di semua peran dan fungsi, mencari hambatan dan kurangnya transparansi. Scrum Master bertanggung jawab atas keberhasilan proses tim dan harus mencari cara untuk menumbuhkan kinerja tinggi. Apa pun yang menghalangi tim untuk berfungsi penuh dan produktif perlu ditangani dan/atau dihilangkan. Misalnya, seorang Scrum

Master melindungi tim dari gangguan eksternal.

Selalu ada hal-hal yang tampaknya muncul bagi anggota tim yang bukan bagian dari pekerjaan Sprint saat ini. Seorang Scrum Master perlu melangkah maju dan memblokir gangguan ini. Tim harus dapat fokus pada tujuan Sprint saat ini. Fokus Scrum Master harus selalu membantu tim mempercepat kemajuan mereka. Untuk menjadi lebih baik setiap hari. Dengan mengalihkan fokus dari tim, Scrum Master memungkinkan tim untuk mengerjakan apa yang penting.

Scrum Master Adalah "Ibu Tim"

Menjadi Scrum Master bisa jadi pekerjaan yang tidak menyenangkan. Jika tim Scrum belum matang dalam hal Agile dan Scrum, seseorang perlu membantu mereka menjadi maniak Agile yang berkinerja tinggi. Ada sisi manusiawi dalam hal ini. Terus terang saja, terkadang anggota tim panik. Mungkin mereka kurang tidur, atau perjalanan ke kantor sangat buruk. Mungkin mereka pergi membeli kopi dan disambut dengan teko kosong, atau mereka sedang dalam suasana hati yang buruk.

Seorang Scrum Master perlu memiliki keterampilan interpersonal untuk menangani semua ini. Seorang Scrum Master harus tahu kapan harus terjun ke dalam suatu situasi, atau menjauh sama sekali. Terkadang pekerjaan ini mengharuskan Anda untuk melerai pertengkaran dan atau argumen; di lain waktu pekerjaan ini mengharuskan Anda untuk memberikan kata-kata penyemangat. Intinya adalah bahwa seorang Scrum Master bertanggung jawab atas kebahagiaan tim Scrum.

Pemilik Produk bertanggung jawab untuk mengomunikasikan "Mengapa," dan Master Scrum bertanggung jawab untuk membantu tim menemukan "Bagaimana" sehingga mereka dapat membangun "Apa." Berharap untuk melakukan hal yang mustahil untuk menjamin bahwa semuanya berjalan sebaik mungkin. Dalam bab ini, kita mengeksplorasi tiga pilar Scrum, peran Scrum, dan beberapa dinamika intra-tim. Saya juga merinci program angkat beban lama saya dan memberikan sedikit saran pembinaan. Dalam bab berikutnya, kita akan melihat beberapa struktur tim Scrum.

BAB 4
STRUKTUR TIM SCRUM

Saya menemukan bahwa kebanyakan orang tidak menganggap pekerjaan sehari-hari mereka dalam konteks beroperasi dalam struktur tim, apalagi menjadi tim yang berkinerja tinggi. Sebelum Agile, tempat kerja adalah tempat yang sangat kompetitif, di mana setiap orang berfokus pada diri mereka sendiri. Orang-orang bekerja secara terpisah di bilik-bilik kerja dan hanya berkumpul untuk rapat status di mana manajemen mencari siapa yang akan ditegur karena mereka tertinggal dalam pekerjaan mereka.

Sekarang fokusnya harus pada tim dan kolaborasi. Mereka yang terdampak oleh keputusan membuat keputusan tersebut. Kegagalan dipandang sebagai peluang untuk berkembang, bukan untuk menyalahkan orang lain. Informasi dan pengetahuan dibagikan secara bebas, dan kualitas lebih ditekankan daripada memenuhi tanggal tertentu. Itu adalah perubahan yang cukup besar dalam dinamika tim.

4.1 TIM YANG TERKOORDINASI

Anggota Tim Scrum perlu berkomunikasi dan berkolaborasi. Seperti yang saya katakan sebelumnya, kolaborasi adalah konflik yang sehat. Komunikasilah yang membuatnya begitu sehat. Jika Anda memikirkannya, semua upacara Scrum dirancang untuk mendorong kolaborasi. Dalam setiap pertemuan, tim berkumpul dan didorong untuk berbagi berbagai sudut pandang. Tim yang bekerja sama memiliki lebih sedikit hambatan dalam berkomunikasi dan pada dasarnya lebih produktif.

Jika Anda perlu berbicara dengan seseorang, Anda cukup mendatangi mejanya dan mengobrol. Itulah yang saya sebut sebagai "momen sejuta dolar". Budaya kolaborasi terbentuk ketika orang-orang dapat dengan mudah berkumpul bersama. Saya lebih suka melihat empat atau lima anggota tim berkumpul di sekitar meja seseorang mendiskusikan apa yang sedang mereka bangun daripada orang-orang yang bekerja keras, mengetik kode, dan bekerja di ruang kerja yang sempit. Diskusi kelompok menuai manfaat dari pengalaman dan sudut pandang setiap orang. Informasi mengalir bebas dalam skenario ini.

Jika memungkinkan, Tim Scrum harus berada di lokasi yang sama. Sebaiknya dalam jarak lima puluh kaki dari satu sama lain. Mengapa lima puluh kaki? Itu sesuatu yang saya amati melalui pengalaman saya sendiri. Saya dulu bekerja di gedung yang memiliki pusat kebugaran. Sangat nyaman untuk dapat berolahraga saat makan siang, dan saya memanfaatkannya sepenuhnya. Saya kebanyakan melakukan latihan kardio. Ya, saya melakukan kardio bersamaan dengan angkat beban dan menekuk batang besi dan semacamnya. Saya bergidik ketika membayangkan betapa gemuknya saya jika tidak melakukannya.

Ada pintu akses luar di depan pusat kebugaran. Pintu ini berjarak sekitar lima puluh kaki dari pintu utama gedung. Orang-orang terus menggunakan pintu pusat kebugaran untuk memasuki gedung. Pada bulan-bulan yang lebih hangat, ini bukan masalah besar. Di musim dingin, ini mengerikan. Saya akan bekerja keras, berkeringat, dan saya akan terkena hembusan

udara dingin setiap kali pintu terbuka. Itu menyebalkan, dan akhirnya pintu itu terkunci.

Yang menurut saya lucu tentang kejadian ini adalah orang-orang yang menggunakan pintu pusat kebugaran tidak menghemat lima puluh langkah. Untuk mencapai lift, seseorang harus berjalan melalui pusat kebugaran dan menaiki lorong. Lift yang sama itu berada tepat di belakang pintu utama. Orang-orang menganggap bahwa pintu pusat kebugaran lebih dekat; itulah sebabnya mereka menggunakannya. Jika orang merasa terlalu sulit untuk menemui seseorang, mereka tidak akan pergi ke sana. Jika mereka duduk terlalu berjauhan, mereka tidak akan bekerja sama. Ini berarti bahwa sebuah tim tidak hanya harus berada di kantor yang sama, tetapi juga benar-benar duduk bersama.

Bukan di bilik atau kantor berdinding tinggi, tetapi di area terbuka yang mendorong upaya kolaboratif. Anggota tim dapat dengan mudah menyampaikan pendapat, keyakinan, dan saran mereka satu sama lain. Tim dapat menyelesaikan perbedaan mereka dengan cepat dan efisien karena mereka duduk bersama dan membahas masalah tersebut secara langsung, yang secara alami meminimalkan kebingungan dan kesalahpahaman. Ketika tim berbagi ruang kerja, kolaborasi menjadi hal yang wajar. Tidak ada yang lebih efektif daripada dapat berbalik dan berbicara dengan seseorang.

Memposting informasi tentang proyek di dinding di lokasi tim meningkatkan visibilitas ke dalam proyek dan memudahkan semua orang untuk memahami status proyek. Tim membutuhkan area umum untuk membuat posting informasi proyek menjadi praktis. Seluruh tim mungkin tidak perlu membawa laptop mereka ke ruang konferensi besar dengan flip chart dan papan tulis (meskipun saya telah melatih tim yang telah melakukan hal ini). Yang penting adalah mengeluarkan tim dari pertanian bilik. Cari cara untuk menghilangkan dinding di antara para anggota. Ketika mereka tidak lagi terisolasi, mereka dapat dengan mudah berkomunikasi.

Tim adalah jantung Scrum. Menurut saya, tidak ada yang lebih hebat daripada Tim Scrum yang termotivasi. Sekelompok individu yang mengerjakan hal yang sama tidak akan berhasil. Tim yang sesungguhnya bersifat kolaboratif dan berdedikasi pada tujuan bersama. Tim Scrum tidak seharusnya mengenali peran pengembangan perangkat lunak tradisional (pengembang, penguji, dan sebagainya). Jangan terpaku pada hal itu. Pahami bahwa pada awalnya cara terbaik bagi tim untuk bekerja adalah dalam peran yang membuat mereka nyaman. Tim dapat dilatih untuk menjadi lebih lintas fungsi saat mereka berupaya menjadi Tim Scrum yang berkinerja tinggi.

Lagi pula, dalam lingkungan Tim Scrum, tidak ada yang namanya pekerjaan Anda dan pekerjaan saya. Yang ada hanyalah pekerjaan yang harus diselesaikan oleh tim, jadi itu semua adalah pekerjaan kita. Tim berkomitmen pada pekerjaan setiap Sprint, dan melakukan apa pun yang diperlukan untuk menyelesaikannya. Ini menciptakan semangat korps tertentu di antara Tim Scrum. Ketika saya masih di militer, frasa itu cukup sering dilontarkan. Semangat korps pada dasarnya berarti perasaan kesetiaan dan persahabatan di antara anggota tim. Katakan apa pun yang Anda mau, tetapi saya tahu bahwa ada ikatan yang kuat antara saya dan orang-orang yang pernah bertugas bersama saya yang masih ada hingga saat ini. Ikatan itu tercipta ketika tim bekerja bersama di tempat yang sama hari demi hari.

4.2 TIM YANG TERSEBAR

Baiklah, bagaimana jika tidak mungkin bagi tim untuk berada di lokasi fisik yang sama? Terkadang keahlian yang dibutuhkan tidak berada di lokasi yang sama. Beberapa orang bekerja dari rumah. Beberapa orang tinggal di berbagai belahan dunia. Tentu saja, Anda ingin tim berada di tempat yang sama. Bagaimana Anda menciptakan rasa tujuan dan komitmen bersama yang sama saat tim tidak bersama? Gunakan teknologi sebanyak mungkin untuk membuat tim berkomunikasi secara langsung sebanyak mungkin. Gunakan ruang konferensi video untuk rapat dan upacara Agile.

Sebagian besar komputer laptop memiliki webcam terintegrasi. Gunakanlah. Lakukan apa pun untuk memastikan bahwa tim berkomunikasi secara efektif. Ada banyak alat kolaborasi yang dapat dipilih. Jangan biarkan hal itu melumpuhkan tim. Pilih sesuatu, mulailah menggunakannya, periksa, dan adaptasi. Saya tidak akan mendukung produk tertentu, karena setiap tim berbeda. Hanya karena sesuatu berhasil untuk tim yang saya latih tidak berarti itu akan berhasil untuk orang lain. Terserah masing-masing tim untuk menentukan apa yang berhasil bagi mereka. Teknologi tidak dapat menggantikan bertemu seseorang di ruang kopi dan mengajukan pertanyaan, tetapi itu bisa mendekati.

Waspadalah terhadap kecenderungan alami "kami lawan mereka" yang akan muncul saat tim tidak berada di lokasi geografis yang sama. Jika memungkinkan, saya ingin menyatukan tim di tempat fisik yang sama dan melakukan pertemuan awal atau semacam team building. Meskipun anggota tim hanya bertemu langsung satu kali, hal itu membuat mereka merasa lebih nyaman untuk saling menghubungi. Tim perlu bersatu dan mencari cara untuk bekerja sama. Tekankan komunikasi dan kolaborasi.

Sejujurnya, saya pernah melihat orang-orang di ruangan yang sama tidak berkomunikasi. Bagian dari pekerjaan Scrum Master sebagai fasilitator adalah menciptakan lingkungan yang memungkinkan kolaborasi dan mendorong anggota tim untuk berkomunikasi. Cari situasi di mana kolaborasi akan membantu dan tunjukkan kepada tim. Itu tidak mudah, tetapi kolaborasi dapat terjadi di seluruh kantor, kota, bahkan planet ini.

Tim Paruh Waktu

Pikirkan tim teknik berkelanjutan. Mereka menghabiskan sebagian besar waktu mereka untuk memperbaiki cacat pelanggan. Ini berarti bahwa mereka mengetahui kode dan memiliki kemampuan pengembangan yang dapat membantu proyek Scrum. Masalahnya adalah mereka digerakkan oleh peristiwa. Sebagian besar pelanggan tidak peduli bahwa teknisi berkelanjutan membantu Tim Scrum. Mereka ingin bug diperbaiki. Bagi saya, ini adalah latihan sederhana dalam perencanaan kapasitas. Sebagian besar tim berkelanjutan yang pernah saya tangani tidak sibuk 100 persen sepanjang waktu.

Mereka dapat mendedikasikan, katakanlah 30 persen dari kapasitas mereka untuk pekerjaan Scrum dan masih dapat memenuhi komitmen teknik berkelanjutan mereka. Transparansi adalah yang terbaik di sini. Jangan terlalu banyak mengalokasikan sumber daya paruh waktu sehingga masalah pelanggan yang berprioritas tinggi merusak Sprint. Bersikaplah realistis tentang apa yang dapat dikomitmenkan oleh tim paruh waktu untuk Sprint.

Scrum Skala Besar

Sampai saat ini, kami telah berfokus pada Tim Scrum. Sebuah tim yang terdiri dari antara, katakanlah enam dan sepuluh orang. Saya berharap bahwa pada titik ini dalam buku ini Anda dapat melihat nilai dari tim kecil yang memberikan nilai dalam iterasi singkat. Namun, apa yang Anda lakukan jika Anda membutuhkan lebih dari sepuluh orang? Anda tidak ingin membuat tim lebih besar. Ketika Tim Scrum menjadi terlalu besar, komunikasi menjadi terganggu. Bagi saya, masuk akal untuk memecah tim besar menjadi sekelompok tim yang lebih kecil.

Tim Fitur

Masuk akal untuk mengatur tim di sekitar fitur. Rasanya alami. Ketika Anda memikirkannya, pelanggan membeli fitur. Fitur memetakan fungsionalitas pengguna yang lengkap. Tim fitur diorganisasikan untuk menyediakan fitur, bukan berfokus pada persyaratan. Tim fitur menyediakan perangkat lunak yang lengkap, berfungsi, dan diiris vertikal setiap iterasi sama seperti Tim Scrum. Perbedaannya adalah bahwa tim fitur mengerjakan fitur dari awal hingga akhir, fitur demi fitur. Dengan kata lain, tim mengerjakan fitur hingga selesai, menyediakan irisan vertikal setiap iterasi. Setelah fitur tersebut selesai, mereka beralih ke fitur berikutnya. Tim memiliki semua yang dibutuhkan untuk melengkapi fitur. Tim memiliki keterampilan, alat, dan wewenang yang diperlukan.

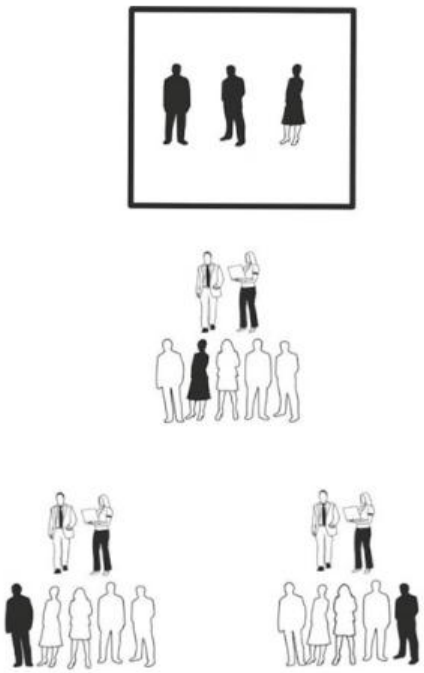
Tim Komponen

Sementara tim fitur mencoba mengiris cerita secara vertikal, tim komponen lebih horizontal melintasi batas arsitektur; misalnya, tim basis data, tim antarmuka pengguna, dan seterusnya. Saya menganggap tim fitur menyediakan irisan vertikal dan tim komponen menyediakan lapisan. Meskipun tampaknya tim fitur adalah yang paling menarik, ada alasan bagus untuk juga memiliki tim komponen. Salah satu alasan terbaik yang dapat saya pikirkan adalah untuk membangun layanan umum yang digunakan oleh banyak tim dan/atau fitur.

4.3 SCRUM-OF-SCRUMS

Saat Anda memiliki beberapa tim Scrum yang mengerjakan sebuah proyek, perlu ada penyelarasan dan transparansi yang lebih baik. Saat menskalakan Scrum, hal ini dicapai dengan rapat Scrum-of-Scrums. Bagi Tim Scrum, semuanya berjalan seperti biasa, kecuali bahwa setiap Tim Scrum memilih satu anggota tim untuk menghadiri rapat Scrum-of-Scrums guna mengoordinasikan pekerjaan. Rapat Scrum-of-Scrums seperti rapat standup harian, kecuali bahwa Anda memiliki anggota dari beberapa Tim Scrum yang berbeda yang melaporkan apa yang sedang dilakukan oleh masing-masing tim.

Biasanya Tim Scrum memilih Scrum Master untuk menghadiri Scrum-of-Scrums, tetapi siapa pun dari tim dapat hadir. Saya bahkan pernah mendengar tim yang bergiliran menghadiri rapat Scrum-of-Scrums. Tujuannya adalah untuk memastikan bahwa informasi mengalir di antara tim dan kembali ke tim, seperti yang terlihat pada Gambar 4.1.



Gambar 4.1 Scrum Of Scrums

Idenya adalah bahwa rapat Scrum-of-Scrums digunakan untuk menyelesaikan ketergantungan antar tim dan menentukan bagaimana tim perlu berkolaborasi untuk memastikan rilis yang sukses. Rapat ini juga dapat digunakan untuk mengidentifikasi dan menghilangkan hambatan. Rapat Scrum-of-Scrums tidak perlu diadakan setiap hari, tetapi harus memiliki irama yang teratur, biasanya setiap beberapa hari. Rapat Scrum-of-Scrums menyerupai rapat standup harian. Perwakilan dari setiap tim melaporkan apa yang telah diselesaikan tim mereka, apa yang akan mereka kerjakan selanjutnya, apakah apa yang mereka lakukan akan memengaruhi tim lain, dan hambatan apa pun di tingkat tim.

Pikirkan hal-hal seperti "Saya butuh Tim A untuk menyelesaikan cerita mereka sebelum tim saya dapat memulai cerita ini." Hambatan-hambatan ini dilacak dalam backlog terpisah. Rapat Scrum-of-Scrums juga berlangsung lebih lama daripada rapat Scrum harian. Rapat Scrum harian bukan untuk memecahkan masalah. Dalam rapat Scrum-of-Scrums, masalah biasanya terjadi di antara tim dan signifikan. Setiap orang yang diminta untuk menyelesaikan masalah sudah ada dalam rapat. Jadi, masuk akal untuk membahasnya di sana. Bayangkan penskalaan Scrum dengan cara ini. Semuanya pada dasarnya tetap sama; hanya saja membesar dalam skala yang lebih besar.

Alih-alih tim lintas fungsi yang dipandu sendiri, Anda memiliki sekelompok perwakilan tim lintas fungsi yang dipandu sendiri yang membentuk tim yang terdiri dari tim-tim yang dipandu sendiri. Seperti yang mungkin dapat Anda bayangkan, model ini dapat diskalakan untuk mengakomodasi ratusan tim. Dalam bab ini, kita melihat berbagai jenis Tim Scrum serta cara menskalakan Scrum menggunakan Scrum-of-Scrums. Tim Scrum harus dapat berkomunikasi dan berkolaborasi bahkan jika para anggotanya berada di lokasi yang terpisah secara geografis. Dalam bab berikutnya, kita akan menjelajahi upacara dan artefak Scrum.

BAB 5
TATA CARA PROYEK SCRUM

Saat itu tahun 1990-an dan entah bagaimana saya dibujuk untuk bekerja di sebuah perusahaan gulat yang liar dan gila. Oke, itu sedikit bohong. Saya suka gulat studio dan selalu begitu. Salah satu kenangan paling awal saya adalah kakek saya (ditunjukkan pada Gambar 5.1) mengumpat di depan televisi dalam bahasa Italia saat menonton gulat.

Gambar 5.1 Lihatlah Scrum Master Kecil Yang Bahagia Bersama Kakek-Neneknya

Tidak, saya tidak bergulat. Saya akan mengatakan bahwa saya dapat menggunakan mikrofon (disebut sebagai "tongkat" oleh mereka yang berkecimpung dalam bisnis ini), tetapi saya tidak tertarik untuk jauh dari keluarga dan bekerja dari koper. Seorang teman saya memiliki tempat kebugaran pada saat itu, dan salah satu pegulat mengangkat beban di sana. Dia meminta beberapa dari kami untuk membantu di pertunjukan gulat dengan bekerja sebagai petugas keamanan. Dia meminta kami datang tiga jam sebelum pertunjukan dengan mengenakan blazer hitam.

Kami datang ke tempat pertunjukan (tepat waktu, perlu saya tambahkan) dan diminta untuk menyiapkan kursi sementara kru jalanan menyiapkan arena. "Kru jalanan" itu, secara mengejutkan, adalah pegulat. Izinkan saya memberi tahu Anda sekarang juga bahwa semua rumor tentang arena gulat profesional yang memiliki bantalan tambahan itu salah. Begitu pertunjukan dimulai, tugas kami adalah memastikan tidak ada yang melempar apa pun ke

dalam arena, dan mengendalikan kerumunan saat aksi menyebar keluar dari lingkaran persegi. Cukup sederhana, kecuali bahwa aksi itu terjadi di antara penonton selama setiap pertandingan.

Sepanjang acara, saya terus memperhatikan dua wanita tua mungil yang duduk di sebelah kanan ring. Saya tidak tahu apa yang mereka lakukan di sana. Kakek saya dulu menertawakan "Rosie di pinggir ring," seorang wanita tua mungil yang duduk di pinggir ring dan berteriak kepada para pegulat. Ini berbeda. Saya khawatir mereka akan terinjak-injak jika aksi itu terjadi terlalu dekat. Untungnya tidak ada yang terlalu dekat dengan mereka sampai acara utama. Acara utama adalah perkelahian sejak awal. Pada satu titik, aksi itu menyebar ke kerumunan, tepat di depan kedua wanita itu. Saya berusaha sekuat tenaga untuk menahan kerumunan, dengan punggung saya menghadap mereka.

Kemudian saya mendengar, "Ini sayang, ambil ini!" Saya berbalik dan melihat kedua wanita itu menawarkan loyang kue kepada para pegulat, yang dengan senang hati mereka ambil dan mulai gunakan sebagai senjata. Saya tidak yakin mengapa saya membagikan cerita itu. Saya kira itu menunjukkan bahwa bahkan dengan niat terbaik, terkadang hal-hal tidak berjalan seperti yang Anda harapkan. Itulah sebabnya artefak Scrum penting untuk diperhatikan secara berkala. Jika Anda menggunakan kerangka Scrum, Anda harus menghasilkan sesuatu. Ya, Anda menghasilkan nilai bagi para pemangku kepentingan, tetapi Anda juga menghasilkan produk sampingan yang mencerminkan kesehatan tim Scrum. Ini disebut artefak Scrum.

5.1 DEFINISI SELESAI DAN KONSENSUS DALAM SCRUM

Bagaimana tim Scrum mengetahui kapan mereka selesai? Saya rasa masuk akal untuk mendefinisikan apa arti "selesai", karena setiap orang mungkin memiliki atau tidak memiliki pendapat yang sama tentang arti sebenarnya dari kata itu. Ini mengingatkan saya pada kutipan terkenal:

Tidak akan berakhir sampai benar-benar berakhir.
—Yogi Berra

Definisi Selesai (DoD) memungkinkan tim secara keseluruhan untuk menyetujui apa artinya... selesai. Menyelaraskan pemahaman setiap orang tentang apa arti sebenarnya dari "selesai" memengaruhi pekerjaan di tingkat cerita (misalnya, "selesai" berarti dikodekan, diuji, dan didokumentasikan). Harapan semua orang terpenuhi, dan tidak ada kejutan, kesenjangan, atau kesalahpahaman saat tim bekerja melalui Sprint. DoD adalah daftar kelengkapan teknis dan profesional. Satu hal yang perlu diingat adalah bahwa DoD harus terpisah dari kriteria penerimaan cerita. Kriteria penerimaan bersifat khusus untuk setiap cerita dan akan bervariasi dari satu cerita ke cerita lainnya.

Kriteria tersebut harus berfokus pada hasil yang diharapkan atau benar dari cerita tersebut. DoD diterapkan pada semua cerita yang dikerjakan oleh tim. Kriteria tersebut tidak berubah; namun, beberapa bagian dari DoD mungkin tidak berlaku untuk semua cerita.

Beberapa tim mencantumkan "kriteria penerimaan terpenuhi" sebagai item pertama dalam DoD. Anda mungkin merasa bingung. Saya akan menjelaskan lebih lanjut, tetapi sebelum saya menjelaskannya, ini adalah waktu yang tepat untuk membicarakan tentang bagaimana sebuah tim mencapai kesepakatan.

Konsensus

Saya pernah menyinggung hal ini sebelumnya. Kebanyakan orang memiliki pemahaman yang keliru tentang apa arti konsensus. Bagi kebanyakan orang, "cara saya di jalan raya" mencakupnya. Hal ini menyebabkan pertengkaran, orang-orang bersikukuh pada pendirian mereka, atau lebih buruk lagi apatis. Seberapa sering Anda melihat orang-orang hanya duduk di sana, mungkin memutar mata mereka saat melihat kejadian yang terjadi di depan mereka. Mereka mungkin setuju dengan apa pun yang diputuskan, tetapi mereka akan keluar dari rapat sambil bergumam "tidak mungkin".

Gambar 5.2 Alter-Ego Saya "Kapten Agile"

Bukan apa yang saya sebut dinamika tim yang sehat. Salah satu hal yang saya pelajari dari para pegulat seperti yang terlihat pada Gambar 5.2 adalah bahasa gaul mereka. Mereka berbicara dengan bahasa mereka sendiri saat berada di sekitar satu sama lain, terutama karena mereka tidak ingin para penggemar (yang mereka sebut "marks") mengetahui apa yang mereka bicarakan. Adu tembak didefinisikan sebagai "realitas brutal" atau kebenaran. Legenda mengatakan bahwa istilah "tembak" berasal dari Jepang. Ketika dua pegulat berhadapan dan

benar-benar bertarung (sungguhan seperti mencoba menyakiti satu sama lain), itu disebut gulat tembak.

Pegulat akan mengatakan hal-hal seperti "Itu tadi adu tembak langsung" atau "Saya sedang bertanding denganmu sekarang" jika mereka berkata jujur. Adu tembak didefinisikan sebagai pura-pura. Kebohongan. Jika Anda sedang dikerjai, itu berarti seseorang bersikap kurang jujur dalam upaya untuk mengalahkan Anda. Dalam hal rapat tim, dalam istilah gulat, saya ingin semua orang ikut serta dalam adu tembak. Membuat rapat menjadi adu tembak tidak membantu siapa pun. Tim harus merasa nyaman dan jujur untuk mencapai konsensus. Percakapan yang terbuka dan jujur mengungkap masalah dan menghasilkan hasil yang lebih baik.

Saya suka mendefinisikan konsensus sebagai "Saya tidak akan menggagalkan keputusan ini." Serius, hasilnya mungkin tidak seperti yang saya harapkan. Saya mungkin tidak mendapatkan apa yang saya inginkan, tetapi jika saya dapat menerima keputusan tersebut, saya akan mendukungnya. Setelah tim mencapai konsensus, semua orang perlu meluruskan diri, membersihkan darah dari dinding, dan berbicara dengan satu suara suara tim.

Gambar 5.3 Lima Jari

Baiklah, saya bercanda tentang darah di dinding (semacam). Rapat bisa jadi buruk. Salah satu teknik yang dapat membantu tim mencapai konsensus disebut "tinju lima". Ya, kedengarannya seperti film seni bela diri yang sangat buruk. Untuk memfasilitasi tinju lima, mintalah tim untuk memberikan suara pada sebuah ide dengan mengangkat jari (Gambar 5.3). Lima jari berarti dukungan total. Yang saya maksud dengan dukungan total adalah dukungan yang menunjukkan "ini adalah ide terbaik yang pernah ada". Anda tidak hanya menyukainya. Anda ingin menikahinya dan punya anak.

Jelas, Anda seharusnya jarang melihat lima jari. Jika ya, Anda mungkin harus menjelaskan prosesnya lagi. Atau tim Anda adalah sekelompok orang sok tahu yang terlalu dramatis. Empat berarti Anda menyukai ide tersebut (Gambar 5.4). Anda merasa bahwa itu

adalah sesuatu yang dapat Anda dukung. Saya pernah mendengar penjelasan bahwa Anda berharap Anda telah memikirkannya. Saya pikir itu menjelaskan keempat jari dengan paling baik.

Gambar 5.4 Empat Jari

Tiga adalah definisi konsensus (Gambar 5.5). Anda dapat menerimanya, dan Anda akan mendukung keputusan tersebut. Itu berarti Anda tidak akan meninggalkan rapat sambil bergumam "tidak mungkin" dalam hati.

Gambar 5.5 Tiga Jari

Dua jari berarti Anda memiliki beberapa masalah dengan ide tersebut dan ingin membahasnya lebih lanjut (Gambar 5.6). Anda tidak sepenuhnya menentangnya, tetapi juga tidak sepenuhnya mendukungnya.

Gambar 5.6 Dua Jari

Satu jari berarti "di atas mayat saya" (Gambar 5.7). Anda memiliki masalah besar dengan ide tersebut dan tidak akan melanjutkannya.

Gambar 5.7 Satu Jari

Seseorang harus selalu diberi isyarat dengan jari telunjuk, tidak peduli seberapa kesalnya Anda jika Anda mengerti maksud saya. Sebagai seorang fasilitator, Anda mencari satu dan dua. Ini adalah kesempatan untuk menyelidiki lebih dalam apa yang sedang dibahas dan lebih jauh mengeksplorasi apakah kesepakatan dapat dicapai. Ini bukan tentang orang-orang yang tidak fleksibel. Ini tentang menemukan konsensus sehingga tim dapat bergerak maju. Kembali ke

Definisi "Selesai" (DoD). Ini mungkin hal terpenting bagi Tim Scrum.

Ingat, tim harus bertanggung jawab atas diri mereka sendiri. Mereka bertanggung jawab atas akuntabilitas mereka sendiri. Jika mereka benar-benar ingin mengarahkan diri sendiri, tim harus mencapai konsensus tentang DoD, lalu mengusahakannya. Dengan kata lain, tim berusaha memenuhi DoD dan kriteria penerimaan dari setiap cerita individu. Tidak ada yang mengawasi tim lagi untuk memastikan bahwa semuanya selesai. Tim mendefinisikan apa yang sudah selesai dan bekerja sesuai definisi itu.

Sangat menggoda bagi Pemilik Produk untuk memasukkan hal-hal dalam kriteria penerimaan yang sebenarnya termasuk dalam DoD. Misalnya, saya ingat situasi di mana Pemilik Produk yang bekerja dengan saya memasukkan frasa "semua pengujian QA ditulis, dijalankan, dan lulus" dalam kriteria penerimaan. Ketika saya bertanya mengapa itu ada di sana, Pemilik Produk mengatakan bahwa pengujian QA sangat penting, dan fokusnya harus ada di sana. Jika tim telah sepakat bahwa pengujian QA (baik manual maupun otomatis) harus ditulis dan lulus sebagai bagian dari DoD, maka tim dan PO dapat yakin bahwa saat cerita telah ditandai sebagai selesai, QA telah selesai.

Tidak perlu meninjau ulang apa pun. Jika cerita telah selesai, semua yang diterapkan di DoD telah selesai. Pemilik Produk perlu percaya bahwa tim akan memenuhi semua item yang berlaku di DoD untuk setiap cerita, dan tim perlu bekerja sesuai dengan DoD dengan itikad baik. DoD perlu dipasang di tempat umum dan mempertimbangkan semua kriteria fungsional dan nonfungsional. Persyaratan fungsional adalah hal-hal yang Anda harapkan. Fungsionalitas khusus yang sedang dibangun oleh tim. Misalnya, helm sepak bola harus mampu menahan beberapa benturan kuat. Persyaratan nonfungsional lebih banyak tentang sistem. Hal-hal seperti skalabilitas dan kinerja. Persyaratan nonfungsional untuk helm sepak bola adalah helm tersebut tersedia dalam berbagai ukuran sehingga cocok untuk berbagai pemain sepak bola. Mengapa memasang DoD di tempat umum, seperti pada Gambar 5.8?

Gambar 5.8 Definisi Selesai Dan Definisi Siap Ditempel Di Dinding Tempat Salah Satu Tim Saya Mengadakan Rapat Standup Harian

Dengan cara ini, tim akan selalu diingatkan tentang hal itu. Mungkin tampak sepele, atau bahkan sesuatu yang biasa Anda lakukan di taman kanak-kanak. Namun, kami mengubah cara kerja tim secara radikal. Ingat, tidak ada yang mengawasi mereka untuk memastikan semuanya selesai. Sekarang terserah kepada tim, dan mereka menggunakan DoD untuk memastikan tidak ada yang terlupakan. Mari kita lihat contoh DoD, yang ditunjukkan pada Tabel 5.1.

Tabel 5.1 Contoh DoD (Definisi Selesai)

	Kriteria	Check-off
1	Desain selesai	Desain telah ditinjau.
2	Kode selesai	Kode telah lolos tinjauan kode tim.
3	Pengujian pengembangan selesai	Cakupan kode 100% dengan pengujian; semua pengujian lulus.
4	Dokumentasi selesai	Penulis dokumen lulus ujian.
5	Proses pergantian QA selesai	Kode telah dipromosikan dalam manajemen sumber.
		Uji regresi QA telah diperbarui.
6	Pengujian QA selesai	Nol cacat.
7	Pengujian regresi selesai	Semua uji regresi telah dijalankan dan lulus.

Ini adalah Definisi Sederhana tentang Selesai. Semua hal di sini mungkin berlaku untuk sebuah cerita. Jika ada yang tidak berlaku, itu tidak berlaku. Beginilah cara DoD dapat berlaku untuk semua cerita. Setiap item dalam DoD mungkin tidak berlaku secara seragam untuk setiap cerita. Misalnya, tidak setiap cerita mungkin memerlukan dokumentasi. Idenya adalah bahwa tim membahas DoD dan menentukan bahwa dokumentasi tidak berlaku di sini. Melakukan percakapan seperti ini sangat penting bagi tim untuk memberikan nilai secara efektif, Sprint demi Sprint.

5.2 INDIVIDU DAN INTERAKSI ATAS PROSES DAN ALAT

Satu orang mungkin berpikir bahwa fungsionalitas dalam cerita perlu didokumentasikan, tetapi yang lain dalam tim mungkin menunjukkan bahwa fungsionalitas tersebut tidak perlu diperlihatkan kepada pengguna akhir, atau telah dimasukkan dalam dokumentasi. Mencapai konsensus sebagai sebuah tim tentang seperti apa "selesai" itu, dan melaksanakannya, memastikan bahwa harapan setiap orang sejalan dengan apa yang akan diproduksi.

Bagaimanapun, membangun perangkat lunak sangatlah rumit. Ini mengharuskan tim untuk bersifat adaptif. Membangun konsensus adalah contoh bagaimana Scrum mengenali kebutuhan akan kemampuan beradaptasi, dan mendorong tim untuk memecahkan masalah secara kolaboratif saat mereka membangun nilai. Karena itu, semua elemen dalam DoD harus didiskusikan dan diperiksa sebelum tim menyatakan sesuatu telah selesai. DoD memudahkan tim untuk membangun hal yang benar dengan benar.

Definisi Selesai juga harus menjadi dokumen hidup yang tumbuh dan berubah seiring dengan perkembangan tim. Tabel 5.2 adalah contoh DoD yang lebih kompleks. Semua tugas perlu dipertimbangkan sebelum sebuah cerita dapat diselesaikan. Oke, jadi tim telah mencapai

konsensus tentang definisi "selesai." Definisi tersebut dicantumkan di tempat yang terlihat. Bagaimana cara penerapannya? Menambahkan kotak centang ke DoD dapat membantu.

Tabel 5.2 Adalah Contoh Dod

Tugas	Keterangan	Pemilik
Pengkodean	Semua kode (baru atau dimodifikasi) yang berkaitan dengan cerita telah ditulis.	Perkembangan
Pengujian Kode	Pemilik cerita telah meninjau dan menguji fungsionalitas yang baru dan/atau berubah.	Perkembangan
Antarmuka Pengguna	Pekerjaan antarmuka pengguna telah selesai.	Perkembangan
Demonstrasi	Fungsionalitas cerita dapat ditunjukkan kepada para pemangku kepentingan.	Tim
Dokumentasi Pembangunan	Semua dokumentasi yang diperlukan telah ditambahkan.	Perkembangan
Terkirim	QA dan/atau Publikasi Teknis telah diberikan satu atau lebih hal berikut: Suatu build atau sekumpulan pustaka yang siap untuk diuji Lingkungan tempat pengujian dapat diselesaikan Pustaka bantuan yang diperbarui Dokumentasi terkini	Perkembangan
Nol Cacat	Semua cacat telah teratasi.	Pengembangan/QA
Pengujian	Fungsionalitas baru atau yang diubah telah diuji.	QA
Pengujian Regresi	Uji regresi telah selesai.	QA
Otomatisasi	Pengujian telah diotomatisasi jika memungkinkan.	QA
Tinjauan Dokumentasi	Penambahan dan/atau perubahan dokumentasi telah ditinjau.	QA
Penyelesaian Dokumentasi	Dokumentasi yang berhadapan dengan pelanggan telah selesai.	Publikasi Teknis
Demonstrasi Tim	Demo telah dilakukan untuk tim.	Tim

Itu mungkin berhasil untuk tim yang lebih matang, tetapi saya lebih suka menekankan DoD—terutama untuk tim yang baru mengenal konsep tersebut. Saya suka mengadakan "rapat umum DoD" singkat saat tim menyatakan sebuah cerita selesai dalam stand-up. Saat ini terjadi, saya memandu tim melalui setiap item di DoD dan menggunakan kepalan tangan lima orang untuk mencapai konsensus. Seperti biasa, saya mencari seseorang untuk mempertanyakan (atau setidaknya ragu-ragu) pada salah satu item yang saya sampaikan. Ini mengarah pada percakapan dan pemahaman yang lebih mendalam tentang status cerita untuk semua yang terlibat. Bonus tambahan di sini adalah DoD membuat perencanaan rilis menjadi lebih mudah. Tim yang bekerja untuk memuaskan DoD sedang menyelesaikan semua hal yang biasanya dikesampingkan untuk melewati gerbang Waterfall (pengujian, ada yang tahu?) sekarang dilakukan setiap Sprint. Hal ini memungkinkan peramalan yang lebih andal sehingga produk yang lebih baik dapat dirilis. Dan itu adalah penembakan langsung.

5.3 DEFINISI SIAP (DOR) DALAM SCRUM

Sebelum lepas landas, setiap pilot akan menjalani daftar periksa prapenerbangan

untuk memastikan bahwa pesawatnya mampu melakukan penerbangan berikutnya. Ini juga merupakan cara yang baik untuk memastikan bahwa tidak ada yang lupa. Anda tidak ingin terbang dan menyadari ada masalah dengan pesawat yang sebenarnya bisa dihindari. Hal yang sama berlaku untuk Definisi Siap (DoR). Bagaimana Anda tahu kapan cerita pengguna siap untuk dimasukkan ke dalam Sprint? Sama seperti DoD yang membantu menentukan kapan cerita selesai, DoR membantu menentukan kapan cerita siap untuk dimasukkan ke dalam Sprint. Cerita harus berukuran tepat.

Sebagian besar tim Scrum mengukur cerita pengguna mereka dengan poin cerita. Ya, poin cerita. Saya hampir bisa mendengar Anda memutar mata sekarang. Kebanyakan orang merasa poin cerita membingungkan, tetapi itu karena mereka memikirkannya dengan cara yang salah. Istri saya tidak suka ketika kami harus pergi ke suatu tempat, dan saya sedang mengerjakan sesuatu. Dalam skenario ini, saya biasanya mengerjakan apa pun yang saya kerjakan hingga siku saya, dan terlambat. Percakapan biasanya seperti ini:

"Menurutmu ini akan memakan waktu berapa lama?" tanya istri saya. "Sepuluh menit atau lebih..." Saya akan menjawab..

Setengah jam kemudian saya masih belum siap, dan istri saya tidak senang. Masalahnya di sini bukanlah saya berbohong. Masalahnya adalah kita manusia tidak pandai memperkirakan berapa lama waktu yang dibutuhkan untuk melakukan sesuatu. Pikirkan seperti ini seberapa besar pekerjaan yang Anda perlukan untuk mengecat ruang tamu? Perlu dicatat bahwa saya tidak bertanya berapa lama waktu yang dibutuhkan, atau apa yang harus Anda lakukan untuk menyelesaikan pekerjaan itu.

Yang ingin saya ketahui adalah seberapa besar pekerjaan itu. Jadi, katakanlah kita memutuskan mengecat ruang tamu adalah pekerjaan besar. Mengetahui hal ini, seberapa besar pekerjaan mengecat kamar tidur utama? Selamat datang di Agile sizing. Jadi mengapa kita melakukan pengukuran dan perencanaan seperti yang kita lakukan di Agile? Maksud saya, bukankah lebih masuk akal untuk merencanakan semuanya dengan cermat seperti yang biasa kita lakukan di Waterfall? Tidak juga. Alasan kita menggunakan apa yang mungkin dianggap sebagai proses yang tidak tepat untuk menentukan seberapa banyak pekerjaan yang dapat dilakukan oleh sebuah tim adalah cara cerita dan/atau tugas saling berhubungan.

Dalam sesuatu yang serumit siklus pengembangan, dapat dipastikan bahwa cerita dan tugas tidak berdiri sendiri. Keduanya saling berhubungan. Dengan kata lain, Anda biasanya tidak langsung mengerjakan cerita 3. Anda harus menyelesaikan cerita 1 dan bagian dari cerita 2 sebelum Anda dapat memulai cerita 3. Mengetahui hal ini, ketika satu cerita membutuhkan waktu lebih lama dari yang diharapkan untuk diselesaikan, tidak masuk akal untuk berpikir bahwa cerita terkait lainnya di Sprint akan membutuhkan waktu lebih sedikit kenyataannya adalah bahwa biasanya semua tugas yang tersisa akan membutuhkan waktu lebih lama.

Selama Anda konsisten dalam cara Anda mengukur cerita, semuanya akan berjalan dengan baik. Lebih lanjut tentang penunjuk cerita nanti. Yang ingin saya sampaikan di sini adalah bahwa cerita harus diukur ukurannya sebelum ditampilkan dalam Sprint. Jika tidak, tim

harus berkumpul dan mengukur cerita sebelum dapat dikerjakan, membuang-buang waktu yang seharusnya digunakan untuk menghasilkan nilai. Kembali ke Definisi Siap (DoR). Yang kami maksud dengan siap adalah siap, bukan siap jenisnya, atau agak siap. DoR adalah kumpulan semua hal yang diperlukan agar cerita pengguna dapat dikembangkan.

Sama seperti DoD, DoR kemungkinan akan berubah dan berkembang selama rilis, tetapi hanya boleh ada satu yang digunakan untuk semua cerita pengguna. Jika DoR didefinisikan dengan benar, semua orang dalam tim harus merasa nyaman dengannya saat dimasukkan ke dalam Sprint mana pun. Sebagai aturan praktis, cerita pengguna harus mengikuti skala INVEST:

I : *Independent* (Independen)
N : *Negoable* (Dapat dinegosiasikan)
V : *Value* (Berharga)
E : *Estimated* (Dapat diperkirakan)
S : *Small* (Kecil)
T : *Tested* (Dapat diuji)

Dengan menggunakan INVEST, tim dapat memastikan bahwa cerita didefinisikan dengan baik. Jangan berhenti di situ saat membangun DoR; pikirkan tentang apa yang dibutuhkan oleh situasi tim Anda. DoR sederhana mungkin terlihat seperti Tabel 5.3.

Tabel 5.3. Contoh FoR

Elemen	Deskripsi
Judul	Bermakna, singkat, dan langsung ke pokok permasalahan. Pelanggan atau pemangku kepentingan harus dapat memahami isi cerita dari judulnya.
Keterangan	Setiap orang di tim Scrum dapat memahami dan mampu menjelaskan tentang apa cerita tersebut. Jika memungkinkan, cerita tersebut harus ditulis dalam format kanonik.
Kriteria Penerimaan	Persyaratan khusus Pemilik Produk yang harus dipenuhi agar cerita dapat diselesaikan. Persyaratan tersebut harus dirinci dalam cerita dan dikomunikasikan kepada tim.
Ketergantungan	Deskripsi cerita mencantumkan semua dependensi dan/atau prasyarat yang diperlukan untuk memulai atau menyelesaikan cerita. Contoh: Instalasi perangkat lunak dan/atau layanan apa pun yang diperlukan untuk memulai cerita, atau pelatihan apa pun yang diperlukan.
Ukuran	Ceritanya sudah diukur. Pekerjaan yang dibutuhkan adalah ukuran relatif dari usaha untuk seluruh tim, bukan dalam satuan waktu aktual. Pilih {1, 3, 5, 10, atau 20} dari skala berikut: 1 Sangat kecil 3 Kecil 5 Ukuran rata-rata 10 Besar 20 Sangat besar (dan mungkin tidak dapat dimuat dalam Sprint)

	Pekerjaan kolektif yang dibutuhkan oleh: QA Pengembangan Dokumentasi
Pemilik	Cerita tersebut telah didukung oleh seseorang di tim Scrum. Orang ini memastikan bahwa cerita tersebut terlaksana.
Dapat dibuktikan	Cerita tersebut memiliki metode di mana pekerjaan yang telah selesai dapat ditunjukkan kepada para pemangku kepentingan di Sprint Review.
Kriteria Pengujian	Cerita tersebut berisi persyaratan untuk pengujian QA.
Kriteria Dokumentasi	Cerita tersebut berisi persyaratan untuk dokumentasi.

DoR yang baik akan memastikan semua orang jelas tentang apa yang diharapkan, tim memiliki apa yang dibutuhkan untuk sukses, dan tidak ada yang terlupakan sebelum ditarik ke Sprint.

Backlog

Backlog adalah tempat semua hal yang bagus berada. Backlog adalah gudang semua persyaratan fungsional dan nonfungsional (ditambah hal-hal lain) yang tercermin dalam cerita pengguna. Persyaratan tersebut muncul sebagai fitur, baik yang baru maupun sebagai perubahan atau perbaikan pada produk yang sudah ada. Backlog dimiliki oleh Pemilik Produk. Itu berarti PO bertanggung jawab untuk memelihara dan memprioritaskan cerita pengguna dalam Backlog. Perhatikan bahwa saya tidak mengatakan bahwa Pemilik Produk menulis semua cerita pengguna dalam Backlog. Siapa pun dapat menulis cerita pengguna; PO harus tahu apa yang masuk dan keluar dari Backlog.

Mari kita lihat lebih dekat cerita pengguna. Cerita pengguna adalah unit kerja yang menciptakan nilai dari sudut pandang pelanggan. Cerita pengguna biasanya berisi pernyataan kanonik sederhana yang terlihat seperti ini: Sebagai pengguna, saya menginginkan sesuatu, sehingga saya dapat memperoleh manfaat. Pengguna adalah orang yang menjadi sasaran cerita tersebut. Saya sarankan Anda tidak menggunakan kata pengguna di sini (sebagai pengguna, saya menginginkan). Anda ingin tim Scrum melihat cerita dari sudut pandang orang yang akan bekerja dengan perangkat lunak tersebut. Istilah yang sering saya tekankan adalah orang yang ada di kaca. Dengan kata lain, orang di pusat data yang menggunakan perangkat lunak ini dengan cara yang tidak pernah kita pikirkan untuk memecahkan masalah yang tidak pernah kita dengar.

Ada frasa itu lagi. Anda akan membacanya beberapa kali lagi dalam buku ini karena menurut saya itu sangat penting. Saya terlibat dalam sebuah proyek di mana selama 6 bulan saya berinteraksi dengan manajemen TI untuk membangun solusi bagi pusat data mereka. Setiap kali kami menunjukkan kepada mereka apa yang sedang kami bangun, mereka menyukainya. Setidaknya sampai kami menunjukkannya kepada operator di pusat data. Reaksi mereka adalah "Kami tidak akan menggunakan itu...". Dan itu saja. Manajemen mungkin menyukainya, tetapi orang-orang yang harus menggunakannya menyimpannya di rak. Pekerjaan selama berbulan-bulan sia-sia karena kami mendengarkan orang yang salah. Bukankah masuk akal untuk mendapatkan umpan balik dari orang yang benar-benar menggunakan produk tersebut? Bagi saya itu tidak perlu dipikirkan lagi, tetapi apakah kita

melakukannya?

Dalam pengalaman saya, saya sering sekali melihat manajemen, atau VP pembelian, atau seseorang yang berada di posisi paling atas dalam rantai komando berinteraksi dengan Tim Scrum atau Pemilik Produk. Coba pikirkan seperti ini: Jika saya membuka stan limun, apakah Anda akan bertanya kepada anak-anak yang minum limun tentang rasanya, atau kepada orang tua yang membeli limun untuk anak-anak mereka?

Saya yakin orang tua akan mengatakan hal-hal seperti:
"Saya tidak suka mereka memberi gula sebanyak itu." "Anda mematok harga terlalu mahal."
"Saya tidak suka tampilan stan Anda."

Perhatikan, tidak ada yang perlu dikhawatirkan tentang rasa minuman yang Anda jual. Kenyataannya, tidak ada perbedaan dengan perangkat lunak. Kami mencoba menjadikan orang-orang yang menggunakan perangkat lunak kami sebagai penggemar kami bukan orang-orang yang mengelola mereka yang menggunakan perangkat lunak kami. Tetapi, bukankah seharusnya tim berfokus pada mereka yang mengendalikan keuangan? Anda tahu, orang yang membuat perjanjian pembelian? Izinkan saya menjawab pertanyaan itu dengan sebuah pertanyaan.

Menurut Anda, apakah orang-orang di pusat data (di atas kaca) senang ketika Anda mengambil perangkat lunak yang mereka suka gunakan? Dari apa yang saya lihat di lokasi pelanggan, ketika pengguna tidak senang, tidak ada yang senang. Jika kita mendapatkan umpan balik dari siapa pun selain pengguna "di atas kaca", apakah itu benar-benar berharga? Sebuah cerita pengguna memaksa tim untuk tidak hanya fokus pada fungsionalitas apa yang ingin mereka berikan atau masalah apa yang ingin mereka selesaikan. Mereka juga harus memikirkan siapa yang menginginkan fungsionalitas tersebut atau yang ingin menyelesaikan masalah tersebut.

Mengetahui hal ini, tim dan Pemilik Produk harus berdiskusi tentang siapa pengguna tersebut dan bagaimana pengguna tersebut melakukan pekerjaan sehari-hari yang diperlukan untuk pekerjaannya. Inilah sebabnya mengapa persona digunakan. Kita akan membahas persona secara terperinci nanti di buku ini. Untuk saat ini, anggap saja kita membuat pengguna imajiner dan memasukkan mereka ke dalam cerita untuk membantu tim membayangkan bagaimana orang-orang nyata di lapangan akan menggunakan perangkat lunak yang dibuat oleh tim. Untuk kembali ke pernyataan kanonik cerita pengguna, sesuatu adalah kebutuhan atau fitur. Itu adalah fungsionalitas yang diinginkan oleh pengguna. Inilah yang akan dibuat dan diberikan oleh tim.

Manfaatnya adalah mengapa cerita tersebut ditulis sejak awal. Itu adalah apa yang akan dicapai pengguna dengan menggunakan fitur tersebut. Itulah nilai yang akan terwujud dengan membangun hal ini. Dan itu penting. Saya telah merusak banyak rapat dengan mengajukan pertanyaan sederhana sementara tim dan Pemilik Produk terus menerus membahas fitur-fitur yang akan dibangun dengan mengatakan satu kalimat singkat: "Mengapa pengguna menginginkan ini?". Jika Anda tidak dapat menjelaskan mengapa pelanggan

menginginkan apa yang ada dalam cerita pengguna, mungkin cerita tersebut tidak diperlukan, tidak peduli seberapa kerennya Pemilik Produk atau tim menganggapnya.

Sebuah cerita pengguna harus dapat berdiri sendiri. Ketika pengembang pertama kali mulai mengatur pekerjaan ke dalam cerita, mereka tergoda untuk membagi pekerjaan menjadi beberapa bagian yang masuk akal bagi seseorang yang membangun perangkat lunak. Misalnya, cerita untuk pekerjaan back-end, cerita untuk pekerjaan basis data, dan cerita untuk antarmuka pengguna. Pada kenyataannya, sebuah cerita membutuhkan semua hal ini untuk berdiri sendiri, jadi mungkin Anda hanya melakukan pekerjaan back-end secukupnya untuk memanfaatkan basis data tiruan yang menggerakkan data sampel melalui antarmuka pengguna sehingga para pemangku kepentingan dapat melihat seperti apa produk tersebut nantinya. Pada Sprint berikutnya, mungkin tim dapat menempatkan basis data nyata sehingga pengguna dapat melihat data nyata.

Ini disebut sebagai pengirisan vertikal. Pengirisan vertikal memungkinkan sebuah cerita menjadi unit kerja yang dapat dibuktikan yang dapat dihargai dan dipahami oleh pelanggan. Lebih lanjut tentang hal itu saat kita berbicara lebih mendalam tentang menulis cerita. Sebuah cerita pengguna juga harus berukuran sedemikian rupa sehingga pekerjaan dapat diselesaikan dalam Sprint. Itu berarti bahwa cerita tersebut dapat diuji, didokumentasikan, dan dapat ditampilkan dalam demo. Cerita tersebut harus memenuhi Definisi Selesai dan kriteria penerimaan. Backlog juga dapat berisi cacat dan utang teknis. Bug adalah kebenaran yang tak terelakkan dalam pengembangan perangkat lunak.

Tidak peduli seberapa keras Anda mencoba, semuanya akan salah dan Anda akan memiliki bug untuk diperbaiki. Dalam dunia Scrum, hal ini disebut cacat. Seperti yang telah Anda lihat, ada cukup banyak rapat dan ritual dalam Scrum. Yang kebanyakan orang lupa adalah bahwa Waterfall juga memiliki ritualnya sendiri. Salah satu favorit saya adalah ketika pengembangan dan dukungan duduk di meja dan berdebat tentang tingkat keparahan bug. Dulu, Anda tidak dapat merilis produk kecuali semua masalah QA tingkat keparahan 1 dan 2 ditutup. Jelas, pengembangan ingin membuat tanggal rilis mereka, dan QA ingin memastikan bahwa produk tersebut berkualitas tinggi.

Hasilnya adalah sesi negosiasi yang akan menyaingi negosiasi kontrak yang paling sulit yang dapat Anda bayangkan. Dalam Agile, jauh lebih mudah standarnya adalah "nol cacat". Yang berarti nol cacat. Seperti tidak ada bug baru: Nol, tidak ada, nol, tidak ada, tidak ada. Idenya adalah bahwa cacat tidak meningkat nilainya seiring waktu, tetapi biayanya meningkat secara eksponensial. Standar yang saya lihat adalah bahwa cacat harus ditutup dalam Sprint tempat cacat itu ditemukan. Cerita awal tidak ditutup sampai cacat tersebut diperbaiki. Idenya adalah bahwa kita tidak, dalam keadaan apa pun, ingin membuat banyak bug.

Oke, itu bagus dan sebagainya, tetapi apa yang kita lakukan dengan semua hal ini yang telah ada di produk kita selama bertahun-tahun? Seperti hal-hal yang "dinegosiasikan" pada masa Waterfall yang tidak pernah kita bersihkan? Itu, teman-teman, adalah utang teknis. Itu dipikirkan persis seperti kedengarannya. Jika Anda tidak punya uang untuk membeli sesuatu yang Anda inginkan saat ini, Anda akan berutang untuk mendapatkannya. Lucunya, setelah hal baru yang mengilap itu memudar. Anda tetap harus membayarnya. Semakin lama waktu yang

dibutuhkan untuk melunasi utang itu, semakin banyak bunga yang harus Anda bayar. Hal yang sama berlaku untuk utang teknis; jauh lebih mahal untuk memperbaiki bug setelah produk dirilis.

Anda perlu membongkar kembali modul-modul, mempelajari kembali cara kerja kode, dan jika pelanggan menemukannya, berusaha sekuat tenaga untuk memperbaikinya dengan cepat dan membuat mereka senang. Setiap produk memiliki sejumlah utang teknis. Sama seperti cacat, utang teknis perlu diidentifikasi dan ditangani. Anda tentu tidak dapat memberikan prioritas setinggi cacat, tetapi Pemilik Produk harus mendorong tim Scrum mereka untuk menanggung sedikit utang teknis setiap Sprint. Angka yang saya pribadi sukai adalah 30 persen dari kapasitas tim.

Saya akan lalai jika tidak mengatakan bahwa hal-hal seperti menulis pengujian otomatis juga merupakan utang teknis. Menurut saya, utang teknis adalah segala sesuatu yang tidak memberikan nilai kepada pelanggan. Secara teknis (maaf atas permainan kata-katanya), cacat juga merupakan utang kami hanya mencoba untuk "membayarnya" segera. Itulah sebabnya cacat dan utang teknis tidak boleh diberi poin cerita. Keduanya harus berdampak negatif pada kecepatan dan memberikan gambaran akurat tentang seberapa cepat tim menyelesaikan fitur.

Anda bertanya, apa itu kecepatan? Saya suka bagaimana teman saya Bob Carpenter mendefinisikannya. Kecepatan adalah seberapa cepat Tim Scrum memberikan nilai kepada pelanggan. Misalnya Anda memutuskan ingin mulai menandai hal-hal dari daftar keinginan Anda. Bagi Anda yang belum tahu, "daftar keinginan" adalah daftar hal-hal yang ingin Anda lakukan sebelum Anda... ehm, meninggal. Jika lari maraton ada di daftar keinginan Anda, seperti yang ditunjukkan pada Gambar 5.9, Anda mungkin ingin menjawab beberapa pertanyaan. Misalnya, berapa lama waktu yang Anda perlukan untuk berlari dalam perlombaan tersebut?

Gambar 5.9 Jim Kokoszynski Di Maraton Pittsburgh. Ya, Dia Berhasil Menyelesaikannya

Itu berarti dia berlari sekitar 25,2 mil lebih jauh dari yang seharusnya saya lakukan. Untuk mengetahui hal ini, Anda perlu tahu seberapa cepat Anda dapat berlari satu mil. Itu tidak akan berhasil. Maraton memiliki jarak 26,2 mil. Anda perlu memperkirakan kecepatan Anda per mil untuk jarak ini. Biasanya, Anda tidak berlari lebih dari 20 mil selama siklus latihan maraton. Jadi, bagaimana Anda memperkirakan kecepatan maraton Anda? Apakah Anda mengada-ada? Yang digunakan kebanyakan orang adalah kalkulator kecepatan. Anda memasukkan waktu Anda dari perlombaan terkini (misalnya, 5k), dan kalkulator tersebut akan memperkirakan berapa lama waktu yang Anda perlukan untuk menyelesaikan perlombaan yang lebih panjang.

Beginilah cara kerja kecepatan dalam Agile. Kecepatan didefinisikan sebagai berapa lama waktu yang dibutuhkan tim untuk menyelesaikan item Backlog. Dengan kata lain, seberapa cepat tim Scrum dapat memberikan nilai kepada pelanggan. Saya sudah membahas tentang ukuran cerita. Tim Agile tidak melakukan latihan ukuran cerita untuk pengalaman; hal itu dilakukan agar kami dapat menentukan kapan proyek akan siap untuk dirilis. Cara kerjanya seperti ini. Sebagai sebuah tim, Anda memiliki Backlog cerita dan epik (yang sebenarnya adalah cerita atau ide yang sangat besar yang perlu dipecah). Selama perencanaan rilis, item dipindahkan ke Backlog Rilis yang lebih terfokus.

Tidak semua hal ini perlu dipersiapkan sepenuhnya sebenarnya tujuannya adalah hanya cerita yang cukup untuk dua atau tiga Sprint yang dipersiapkan sepenuhnya. Ini mungkin belum masuk akal bagi Anda, tetapi ini ide yang bagus. Ingat, kita ingin fleksibel dan bereaksi terhadap masukan. Mengapa menghabiskan waktu untuk mempersiapkan hal-hal yang mungkin atau mungkin tidak Anda ubah (atau bahkan lakukan)? Kuncinya adalah bahwa semua yang ada di Backlog rilis diestimasikan. Semua cerita dan epik memiliki nilai poin. Ini memberi Anda angka. Untuk menyederhanakannya, katakanlah Backlog rilis Anda berisi 500 poin cerita.

Saat tim Anda mengerjakan Sprint, Anda akan mendapatkan gambaran yang cukup bagus tentang berapa banyak poin cerita yang dapat mereka selesaikan per iterasi; itulah kecepatan tim Anda. Jadi, jika kecepatan tim Anda adalah 50, Anda akan membutuhkan 10 Sprint untuk menyelesaikan apa yang ada di Backlog rilis. Kecepatan rata-rata akan tetap konstan selama durasi proyek lebih atau kurang. Kecepatan mungkin meningkat seiring waktu saat tim melakukan perbaikan. Saat tim baru mengenal Scrum, kecepatan mungkin berfluktuasi liar pada awalnya, tetapi rata-rata akan stabil dalam beberapa Sprint.

Hal ini membuat kecepatan berguna untuk memprediksi kapan proyek akan berakhir. Jika pekerjaan dalam Release Backlog berubah (misalnya Pemilik Produk menambahkan cerita dan meningkatkan jumlah poin cerita berdasarkan umpan balik pelanggan), kecepatan akan memberi tahu Anda kapan tim akan selesai. Inilah sebabnya mengapa kecepatan didefinisikan dalam satuan nilai (cerita) dan bukan satuan upaya (tugas). Anda mendapatkan gambaran akurat tentang kapan tim dapat menyelesaikan pekerjaan secara realistis. Lewatlah sudah hari-hari ketika manajer pengembangan akan menetapkan tanggal rilis. Dalam Agile, kemampuan tim untuk memberikan nilai adalah yang akan membuat penentuan itu.

Kembali ke cacat sejenak. Tim Scrum perlu mengambil kebijakan tanpa cacat dengan sangat serius. Seperti yang telah saya katakan sebelumnya, tim Scrum harus setransparan mungkin tentang segala hal, termasuk cacat. Ketika cacat teridentifikasi, cacat itu perlu dilacak

dengan sisa pekerjaan dalam Sprint. Cacat itu harus masuk ke Sprint Backlog dan diberi prioritas setinggi mungkin. Itu adalah penilaian, tetapi saya akan mengatakan bahwa jika cacat yang ditemukan saat ini dalam Sprint cukup serius, pekerjaan Sprint harus dihentikan jika tim merasa tidak dapat mengatasinya secara efektif. Sekarang setelah Anda mengetahui apa saja yang ada dalam Backlog, seperti apakah Backlog yang baik? Backlog yang sehat disempurnakan dan diprioritaskan.

Saat cerita naik dalam prioritas, cerita tersebut menjadi lebih jelas (atau disempurnakan). Pemilik Produk harus berhati-hati agar Backlog mereka tidak menjadi apa yang saya sebut sebagai "Backlog yang gemuk." Backlog yang gemuk adalah Backlog yang besar, membengkak, dan sulit diatur. Backlog dimulai dengan niat yang baik. Semua persyaratan pemangku kepentingan ini ada di sana karena kami ingin menerapkannya suatu hari nanti. Anggap saja seperti lemari saya. Berat badan saya cenderung sedikit berfluktuasi. Lemari saya berisi pakaian "gemuk" yang saat ini saya pakai. Saya juga punya pakaian yang terlalu kecil untuk saya: untuk berjaga-jaga jika saya memutuskan untuk melakukan diet dan menurunkan berat badan. Jadi lemari saya penuh dengan pakaian yang sangat banyak sebagian besar tidak akan pernah saya pakai seperti yang terlihat pada Gambar 5.10.

Gambar 5.10 Ya, Ini Lemari Saya. Penuh Dengan Barang-Barang Yang Jarang Saya Pakai

Tentu, saya menyimpan pakaian-pakaian ini karena saya tidak suka membuangnya bagaimanapun juga, saya menghabiskan banyak uang untuk pakaian-pakaian ini. Sejujurnya, bahkan jika saya sampai pada titik di mana saya bisa mengenakan pakaian-pakaian itu, pakaian-pakaian itu akan ketinggalan zaman. Jadi, mengapa pakaian-pakaian itu memenuhi lemari saya? Hal yang sama terjadi pada Backlog. Mengapa kita memiliki semua barang yang mungkin tidak pernah kita lakukan sehingga semuanya menjadi berantakan?

Saya ingat pernah berbicara dengan seorang Product Owner tentang ukuran Backlog-nya. Saat itu, Backlog-nya berisi lebih dari 400 cerita. Backlog-nya benar-benar wilayah yang sangat besar. Saya berbicara kepadanya tentang membuang beberapa barang di Backlog-nya,

dan hasilnya tidak memuaskan. “Itu semua barang yang pelanggan sempat katakan kepada kami bahwa mereka menginginkannya,” katanya. "Baiklah," balasku, "tetapi kamu tidak akan punya waktu untuk melakukan semua hal itu dalam empat rilis, apalagi yang sedang kita kerjakan saat ini”.

Di dunia yang sempurna, kita dapat memenuhi setiap hal kecil yang diminta pelanggan. Saya juga akan mengendarai unicorn ke kantor, tetapi itu tidak penting. Kenyataannya adalah bahwa ada jumlah terbatas hal yang dapat dihasilkan oleh tim Scrum. Backlog perlu mencerminkan fakta ini sehingga tim dapat mengetahui prioritasnya, apa yang mungkin akan datang di masa mendatang, dan pekerjaan yang tidak terduga. Backlog yang besar dan tebal akan menyulitkan hal ini. Saya tidak menentang Pemilik Produk membuat Backlog “jaga-jaga” yang berisi cerita dengan prioritas sangat rendah yang mungkin akan menarik perhatian suatu hari nanti.

Meskipun demikian, saya masih mempertanyakan apakah hal-hal itu perlu disimpan. Ingat, Anda setidaknya memiliki Backlog Rilis, yang berisi apa yang Anda rencanakan untuk dikerjakan dalam rilis ini, dan Backlog Sprint, yang berisi hal-hal yang Anda rencanakan untuk diselesaikan dalam Sprint saat ini. Sebagian besar tim juga memiliki Backlog umum yang berisi hal-hal yang penting, tetapi tidak ada dalam rilis yang sedang dikerjakan.

5.4 BAGAN BURN-DOWN DAN BURN-UP

Bagan burn-down dan burn-up digunakan untuk menunjukkan kemajuan Sprint atau rilis. Bagan burn-down menunjukkan upaya yang tersisa dalam Sprint. Bagan burn-up menunjukkan apa yang telah disampaikan hingga titik waktu ini dalam proyek. Mari kita kembali dan berbicara tentang cerita lagi. Ketika sebuah cerita ditarik ke dalam Sprint, itu harus memenuhi Definisi Siap. Untuk tujuan kita, kita akan berasumsi bahwa cerita tersebut berukuran tepat. Selama rapat Perencanaan Sprint, tim melihat cerita dan membaginya menjadi tugas.

Tugas adalah segmen pekerjaan dan dihitung dalam jam. Namun tidak seperti yang Anda pikirkan. Tim Agile bekerja dalam "hari yang ideal”. Saya telah menjalani sebagian besar hidup saya di kota pabrik di Pennsylvania Barat bernama Aliquippa. Pabrik baja telah lama tutup, tetapi etos kerja masih ada di sini. Pekerjaan sehari yang jujur untuk upah sehari yang jujur adalah mantra yang saya dengar sepanjang hidup saya. Hari yang ideal untuk pekerja perangkat lunak adalah sekitar lima jam. Percayalah, orang-orang sulit menerima hal itu. Saya tahu bahwa saya telah dikondisikan untuk percaya bahwa jika Anda bekerja kurang dari delapan atau sembilan jam sehari, Anda curang.

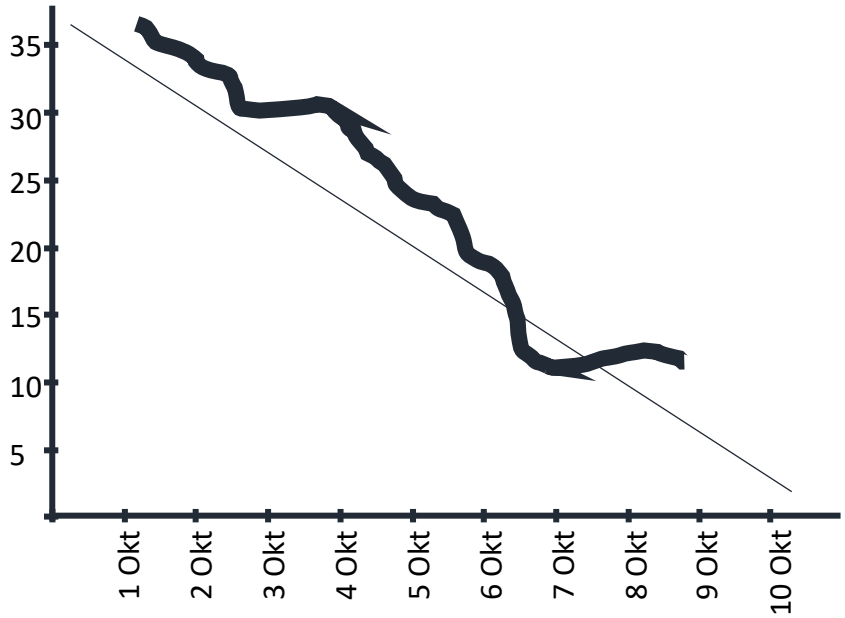
Hari yang ideal bukanlah saat Anda bekerja lima jam dan pulang. Untuk lebih jelasnya, saya tidak menyarankan Anda menghabiskan lima jam sehari untuk bekerja dan sisanya memperbarui profil Facebook Anda. Apa yang kami coba lakukan adalah bersikap "jujur" tentang hari kerja kami yang jujur. Sejujurnya, kami tidak pernah duduk dan menghabiskan delapan jam sehari untuk mengetik kode atau melakukan pengujian. Kami menghadiri rapat, menangani email kami, berbicara dengan rekan satu tim kami tentang desain atau masalah yang kami hadapi dengan apa yang sedang kami kerjakan, menghadiri rapat balai kota, minum

kopi, mengunjungi kamar kecil, dan banyak hal lain yang tidak akan saya jelaskan di sini.

Lima jam tersebut seharusnya mencerminkan berapa banyak waktu yang kami habiskan untuk benar-benar mengerjakan cerita pengguna dalam sehari. Ambil hari-hari ideal Anda dan kalikan dengan jumlah hari kerja yang ada di Sprint mendatang. Sebagian besar Sprint dua minggu akan memiliki sepuluh hari kerja. Sekarang kurangi satu untuk semua upacara Agile (yang belum kita bicarakan), dan Anda akan mendapatkan sekitar 45 jam kapasitas yang tersedia untuk Sprint.

Jelas, jika Anda berencana untuk mengambil hari libur selama Sprint, itu akan mengurangi kapasitas Anda. Jumlahkan kapasitas semua orang dan sekarang Anda memiliki gambaran tentang berapa banyak pekerjaan yang dapat dilakukan tim selama Sprint. Selama rapat Perencanaan Sprint (atau upacara), tim, PO, dan Scrum Master akan mengambil cerita yang ditarik ke dalam Sprint dan menguraikannya menjadi tugas. Saya akan membahas lebih dalam tentang bagaimana ini dilakukan nanti di bab ini, tetapi untuk saat ini pahamilah bahwa sebuah cerita akan dipecah menjadi beberapa tugas, dan tugas-tugas ini akan diberikan kepada anggota tim dan diperkirakan.

Estimasi tugas dilakukan dalam hitungan jam. Kembali ke contoh "Mengecat Ruang Tamu", saya mungkin memiliki tugas berjudul "Mengecat langit-langit" yang saya perkirakan akan memakan waktu dua jam. Tugas lainnya mungkin "Menutupi semuanya dengan kain penutup," yang dikerjakan orang lain dan diperkirakan memakan waktu satu jam.



Gambar 5.11 Contoh Bagan Burn-Down Sprint

Semua jam ini akan bertambah untuk mencerminkan berapa banyak pekerjaan yang dibutuhkan dalam Sprint. Tim berhati-hati untuk tidak melampaui kapasitas tim, dan anggota tim berhati-hati untuk tidak berkomitmen melebihi kapasitas pribadi mereka. Saat pekerjaan selesai, jam-jam ini "dibakar" atau dikurangi dari jumlah yang mencerminkan berapa banyak pekerjaan yang tersisa untuk dilakukan dalam Sprint. Scrum Master mengambil informasi ini

dan menerbitkan bagan burn-down setiap hari, seperti yang terlihat pada Gambar 5.11. Bagan burn-down menunjukkan berapa banyak pekerjaan yang tersisa dalam iterasi saat ini. Bagan burn-down harus diperlihatkan kepada tim setiap hari. Ini memberikan indikasi akurat tentang berapa banyak pekerjaan yang diselesaikan, dan mampu memprediksi dengan baik apakah tim akan menyelesaikan semua pekerjaan yang telah dijanjikan untuk iterasi saat ini.

Burn-up rilis bekerja dengan prinsip yang sama, kecuali dengan story point. Saat tim menyelesaikan story, story point yang terkait dengan story tersebut "dibakar habis." Setelah tiga atau lebih Sprint, tim harus memiliki kecepatan yang andal. Seperti yang telah kita bahas sebelumnya, kecepatan adalah jumlah rata-rata story point yang dapat diselesaikan tim dalam satu Sprint. Scrum Master menerbitkan bagan burn-up rilis yang menunjukkan story point apa yang telah dibakar tim, serta berapa banyak story point yang tersisa dalam Backlog rilis.

Stand-up Harian atau Rapat Scrum Harian

Scrum harian, atau rapat stand-up, mungkin merupakan satu upacara Agile yang menyebabkan jumlah kecemasan terbesar. Tampaknya orang-orang tidak dapat memahami konsep rapat harian yang singkat. Bahkan, mereka panik karenanya. Saya dapat memahami mengapa ini terjadi. Menurut pendapat saya, salah satu pilar budaya perintah dan kendali adalah rapat status. Dalam rapat status Waterfall, manajer pengembangan akan meminta setiap orang dalam tim untuk memberikan pembaruan status. Anda tahu, apa yang telah mereka lakukan sejak rapat status terakhir. Kedengarannya masuk akal; namun, bukan itu yang terjadi. Masalah dengan perintah dan kendali adalah perintah tersebut memberikan apa yang Anda harapkan, tetapi belum tentu apa yang Anda inginkan.

Karena orang yang secara langsung mengendalikan tinjauan Anda (dan gaji Anda) yang menjalankan rapat, orang-orang dalam rapat tersebut mencoba memberi tahu manajer pengembangan apa yang ingin didengarnya. Dengan kata lain, mereka mencoba menutupi kesalahan kolektif mereka. Berikut ini adalah istilah gulat profesional lainnya untuk Anda. Kata sell berarti bertindak. Membuat penonton percaya bahwa apa yang terjadi di atas ring adalah sesuatu yang sebenarnya tidak terjadi. Misalnya, lawan saya mengayunkan tinjunya dengan liar dan mendaratkan pukulan cinta di rahang saya, tetapi saya bertindak seolah-olah dia baru saja merontokkan empat gigi saya. Jika Anda mengamati pertandingan gulat profesional dengan saksama, Anda akan melihat bahwa hampir semuanya dilebih-lebihkan.

Dari ekspresi wajah hingga cara menahan pukulan hingga cara pegulat bergerak di sekitar ring dan memberi isyarat kepada penonton semuanya tentang menjual apa yang Anda lakukan kepada penonton. Jika seorang pegulat tidak bertindak seolah-olah gerakan lawannya tidak terlalu memengaruhinya, ia akan dituduh tidak menjual. Lebih buruk lagi jika Anda dituduh menjual secara berlebihan. Di sinilah Anda bertindak terlalu berlebihan dan membuat segalanya tampak tidak realistis. Semua orang dalam rapat status menjual kepada bos mereka, dan ini menciptakan beberapa masalah.

Sebagian besar rapat status mengalami kurangnya transparansi secara keseluruhan. Tidak seorang pun ingin memberi manajer pengembangan kabar buruk, jadi mereka menyembunyikannya selama mungkin. Mereka berharap dapat memperbaikinya pada rapat status berikutnya. Manajer pengembangan dan pimpinan teknologi membuat keputusan

tentang apa yang perlu dilakukan pada rapat ini. Pada dasarnya, kebanyakan orang tidak suka diberi tahu apa yang harus dilakukan, jadi ini biasanya tidak menyenangkan bagi tim. Rapat status juga merupakan urusan yang panjang dan sulit karena semua orang perlu duduk dan mendengarkan setiap pembaruan dari setiap orang dalam tim. "Dengarkan" mungkin kata yang terlalu kuat.

Sejujurnya, kebanyakan orang dalam rapat status tidak benar-benar mendengarkan. Mereka memikirkan apa yang akan mereka katakan saat giliran mereka tiba atau berpura-pura peduli dengan apa yang sedang terjadi. Rapat harian bukanlah rapat status. Ini adalah rapat komitmen. Alih-alih memberi tahu manajer apa yang mereka lakukan, tim membuat komitmen satu sama lain. Setiap 24 jam, tim berkumpul dan menjawab tiga pertanyaan:

- Apa yang saya lakukan kemarin?
- Apa yang saya rencanakan untuk dilakukan hari ini?
- Apa yang menghalangi saya?

Menjawab ketiga pertanyaan ini merupakan cara yang luar biasa bagi tim untuk memahami dengan tepat pekerjaan Sprint apa yang telah dilakukan dan apa yang tersisa. Ketika seorang anggota tim mengatakan apa yang akan ia lakukan hari ini, ia membuat komitmen kepada timnya. Jika orang lain dalam tim dapat membantu, mereka menawarkan jasa mereka membuat komitmen. Ketika orang berkomitmen pada suatu tugas, ada rasa bangga dan kepemilikan. Ini merupakan dinamika yang sama sekali berbeda dari seorang manajer yang membagi-bagi pekerjaan.

Kemajuan diukur setiap hari saat anggota tim merinci apa yang telah dicapai pada hari sebelumnya. Apa pun yang menghalangi kemajuan anggota tim ditangani oleh Scrum Master. Dengan cara ini, anggota tim tidak perlu khawatir tentang penanganan masalah tersebut. Sekarang, hal itu menjadi tanggung jawab Scrum Master. Anggota tim sekarang dapat fokus pada pekerjaan Sprint. Misalnya, jika seorang anggota tim mengalami masalah dengan komputer laptopnya, hal itu akan disampaikan sebagai hambatan pada rapat harian, dan Scrum Master akan mengurus perbaikan laptop tersebut.

Rapat harian seharusnya tidak lebih dari lima belas menit. Ini bukan tempat untuk memecahkan masalah, merencanakan, atau mendesain. Selain itu, hanya anggota tim yang boleh berbicara. Untuk lebih jelasnya, siapa pun dapat menghadiri stand-up harian. Ini adalah cara yang sangat baik untuk melihat perkembangan proyek, tetapi rapat tersebut adalah milik tim. Jika ada sesuatu yang muncul yang melampaui semangat dan tujuan rapat stand-up harian, rapat tersebut harus dipindahkan ke tempat parkir. Tempat parkir adalah tempat tim "memarkir" subjek untuk dibahas nanti.

Saya biasanya menggantung selembar kertas besar di dinding tempat tim mengadakan rapat stand-up harian dan menempelkan catatan tempel di atasnya saat ada masalah di tempat parker. Dan tempat parkir selalu kosong saat semua orang pulang. Saat catatan tempel ditempel di kertas tempat parkir, saya berharap hal itu akan diurus hari itu juga.

5.5 PENYEMPURNAAN BACKLOG

Penyempurnaan backlog (atau grooming, seperti yang sebelumnya dikenal) adalah saat tim berkumpul dengan Product Owner dan Scrum Master dan mengerjakan cerita dalam Backlog. Backlog harus terlihat, terorganisasi, dan terkini. Persyaratan berubah dengan cepat, dan Backlog (atau Backlog-backlog) harus mencerminkan hal itu. Product Owner harus terus mengevaluasi Backlog. Saat persyaratan berubah, cerita mungkin perlu berubah prioritasnya. Saat cerita diprioritaskan lebih tinggi dan semakin dekat untuk dimasukkan ke dalam Sprint, cerita tersebut harus disempurnakan. Dengan kata lain, cerita tersebut harus didefinisikan dengan lebih jelas sehingga tim memahami nilai apa yang akan dihasilkan.

Product Owner menerbitkan daftar cerita yang ingin dibicarakannya sebelum rapat sehingga tim memiliki kesempatan untuk memeriksanya. Saya ingin tim memiliki waktu setidaknya 24 jam, tetapi itu bukan aturan yang pasti. Secara teknis, tim harus memiliki ide yang cukup bagus tentang apa yang akan datang jika mereka memperhatikan bagaimana cerita diprioritaskan dalam Release Backlog. Pemilik Produk memulai rapat dengan memperkenalkan cerita dengan prioritas tertinggi dan merinci apa yang menjadi "permintaan" mengapa pelanggan menginginkan fungsi ini, dan bagaimana cara penggunaannya.

Tim mencari tahu apa yang akan dikembangkan sebagai hasil dari pengerjaan cerita ini. Tim meminta Pemilik Produk untuk lebih jauh mendefinisikan "permintaan." Tim perlu melihat cerita dari sudut pandang pelanggan, dan ini adalah kesempatan Pemilik Produk untuk memandu persepsi tim terhadap cerita tersebut. Tim mencoba mengonseptualisasikan cerita, jadi Pemilik Produk perlu "berperan sebagai pelanggan" di sini dan menjelaskan mengapa pelanggan menginginkan fungsi tersebut, bagaimana cara penggunaannya, dan persyaratan apa yang penting untuk diingat. Kuncinya adalah tidak terlalu teknis. Perlu ada diskusi yang cukup untuk membuat cerita dipahami dan diukur dengan benar, bukan untuk merancang fungsi di awal.

Tim harus berusaha untuk menyempurnakan cerita sebanyak dua atau tiga Sprint. Ingat, Tim Scrum harus mengharapkan perubahan. Tidak ada alasan untuk menyempurnakan tiga ratus cerita kecuali tim dapat menyelesaikannya dalam tiga Sprint. Jika tim mengharapkan perubahan, mereka mengharapkan prioritas berubah secara drastis dari Sprint ke Sprint.

Perencanaan Sprint

Rapat Perencanaan Sprint adalah tempat Pemilik Produk menjelaskan Sasaran Sprint dan fitur dengan prioritas tertinggi kepada tim. Tim mengubah cerita pengguna menjadi daftar tugas kerja yang terperinci. Di sinilah Pemilik Produk berperan sebagai pelanggan. Tim mengajukan pertanyaan tentang cerita yang ingin dilihat PO dalam Sprint, dan PO menjawabnya dari sudut pandang pelanggan. Cobalah untuk tidak hanya membuat tugas untuk pengembangan, tugas untuk QA, dan tugas untuk dokumentasi. Misalnya, katakanlah kita sedang mendiskusikan sebuah cerita tentang pembuatan sabuk angkat beban (Gambar 5.12). Ya, saya mengerti bahwa sabuk angkat beban tidak ada hubungannya dengan perangkat lunak. Bekerjasamalah dengan saya di sini.

Gambar 5.12 Sabuk Angkat Beban Saya. Beban Itu Akan Melipat Saya Menjadi Dua Jika Saya Tidak Memakainya

Pernyataan kanonik cerita tersebut akan terlihat seperti ini: Dave, seorang atlet angkat beban kelas super berat, menginginkan sabuk berat untuk melindungi punggungnya sehingga ia dapat berkompetisi dalam kompetisi angkat beban tanpa mengalami cedera. Kriteria Penerimaan: Sabuk harus terbuat dari kulit atau suede. Sabuk harus cukup tahan lama untuk menahan kerasnya latihan berat, tetapi cukup nyaman untuk dikenakan untuk ketiga acara angkat beban (squat, bench press, dan deadlift).

Selama penyempurnaan, tim meminta Pemilik Produk untuk mendefinisikan kebutuhan pelanggan dengan lebih baik. PO (berbicara dari sudut pandang pelanggan) mengatakan bahwa atlet angkat beban mengandalkan sabuk tebal dan kokoh untuk melindungi punggung mereka saat mengangkat beban. Sebelum mengangkat beban, mereka mengencangkan sabuk sekencang mungkin dan menahan otot perut mereka pada sabuk. Tekanan antar otot dari tindakan ini membantu menstabilkan inti tubuh mereka dan menjaga mereka tetap tegak selama mengangkat beban.

Setelah membahas cerita tersebut lebih lanjut dengan PO, tim membagi cerita tersebut menjadi 10 poin cerita. Sekarang tim tersebut ingin menarik cerita tersebut ke dalam Sprint. Saat tim mulai berpikir untuk menguraikan cerita tersebut menjadi tugas-tugas, beberapa informasi akan muncul, dan tim mungkin perlu memikirkan tugas-tugas yang perlu dibuat. Misalnya, tim mungkin bertanya di organisasi angkat beban mana Dave akan berlatih. Ada beberapa organisasi angkat beban di Amerika Serikat. Akan lebih baik jika sabuk kami dapat memenuhi spesifikasi dan legal di semua organisasi, tetapi hal itu telah ditetapkan sebagai sesuatu yang tidak mungkin.

Tim perlu memastikan bahwa sabuk tersebut legal di organisasi yang diinginkan

pelanggan. PO memberi tahu tim organisasi mana yang diinginkan pelanggan, dan tim membuat tugas untuk memastikan bahwa sabuk tersebut memenuhi spesifikasi. Seorang pengembang mencantumkan nama mereka pada tugas tersebut dan memperkirakan akan memakan waktu tujuh jam. Pengembang memperkirakan bahwa mereka perlu mendapatkan salinan aturan organisasi angkat beban, mendokumentasikannya di halaman wiki tim, dan memberi tahu anggota tim lainnya bagaimana sabuk tersebut harus dibuat. Berikutnya, tugas QA dibuat untuk memastikan bahwa setelah sabuk dibuat, sabuk tersebut memenuhi spesifikasi yang ditentukan dalam tugas sebelumnya.

Tugas juga dibuat untuk mendokumentasikan spesifikasi sehingga tidak ada pertanyaan bahwa sabuk tersebut legal di organisasi yang dipilih. Saat tim mendiskusikan tugas, beberapa elemen desain akan muncul. Misalnya, PO menyebutkan bahwa meskipun bukan kebutuhan mutlak, pelanggan menginginkan sabuk dengan satu cabang. Biasanya, seorang pengangkat akan membutuhkan satu atau dua orang untuk menarik sabuk agar sekencang mungkin. Jauh lebih mudah untuk mengencangkan sabuk dengan satu cabang daripada dua. Tugas ditambahkan untuk membuat gesper satu cabang, serta tugas QA yang sesuai untuk memastikannya berfungsi dengan benar.

Setiap tugas ditugaskan kepada seseorang di tim dan diperkirakan dalam hitungan jam. Setelah tim merasa senang bahwa cerita tersebut dibagi dengan benar menjadi beberapa tugas, rutinitas yang sama dilakukan pada cerita berikutnya. Untuk lebih jelasnya, itulah cerita tertinggi berikutnya pada Backlog yang diprioritaskan oleh Pemilik Produk. Mungkin ada alasan mengapa tim tidak dapat menarik cerita berdasarkan urutan prioritas. Misalnya, mungkin ada cerita di bagian bawah daftar yang perlu diselesaikan untuk menyelesaikan cerita yang baru saja disetujui untuk dikerjakan oleh tim.

Tim mungkin tidak memiliki cukup kapasitas yang tersisa di Sprint untuk menerima cerita berikutnya; namun, mereka mungkin memiliki ruang untuk cerita yang lebih kecil yang berada di bagian bawah daftar prioritas. Jika PO setuju, mereka dapat menerima cerita tersebut berikutnya. Jika memungkinkan, tim perlu menghormati prioritas yang ditetapkan oleh PO. Tim perlu fokus untuk memenuhi Sasaran Sprint sebagaimana yang ditetapkan oleh PO. Saat mempersiapkan rapat Perencanaan Sprint, Scrum Master harus mencari tahu ketersediaan tim untuk Sprint mendatang. Sprint dua minggu yang normal memiliki sembilan hari kerja atau 45 jam yang tersedia per orang (sepuluh hari ideal berdurasi lima jam dikurangi satu hari untuk demo akhir Sprint dan rapat Perencanaan Sprint).

Kami bersikap setransparan mungkin, jadi perhitungkan hari libur, liburan yang direncanakan, atau apa pun yang akan menghalangi pekerjaan untuk diselesaikan. Misalnya, tim beranggotakan enam orang tanpa liburan atau hari libur akan memiliki "bucket" 270 jam untuk Sprint dua minggu mendatang. Kemudian kami "mengisi bucket" sebagai satu tim. Saat anggota tim setuju untuk mengerjakan tugas, perkiraan jam mengurangi jam yang tersedia di bucket tim dan individu. Penting untuk mengawasi kapasitas individu setiap anggota tim serta kapasitas tim. Baik tim maupun individu mana pun tidak boleh kelebihan beban untuk Sprint. Misalnya, jika penguji hampir mencapai kapasitas, tetapi masih ada ruang bagi orang lain untuk berkomitmen pada lebih banyak pekerjaan, orang lain di tim harus setuju untuk mengambil

alih tugas terkait pengujian yang tersisa.

Tentu saja, tim lintas fungsi yang sebenarnya tidak memiliki peran yang ditentukan sehingga bucket individu tidak perlu dipertahankan. Jika tim masih memiliki peran yang ditentukan, sebaiknya lacak kapasitas individu. Itu membawa saya ke poin penting. Tim berkomitmen pada apa yang menurutnya dapat diselesaikan dalam Sprint. Estimasi tidak dilakukan sebagai sistem pelacakan jam untuk manajemen. Hal ini dilakukan untuk memastikan bahwa tim memiliki peluang realistis untuk menyelesaikan semua cerita yang ditarik ke dalam Sprint.

5.6 SPRINT RETROSPECTIVE

Terkadang, saat tim sedang kesulitan, mereka akan mengadakan rapat tertutup. Tidak masalah olahraga apa. Bisa sepak bola, basket, hoki, atau sepak bola Amerika. Olahraga tim apa pun bisa melakukannya. Yang terjadi adalah tim mengadakan rapat khusus untuk anggota tim, biasanya di ruang ganti. Tidak ada pelatih, manajemen, atau pers yang diundang. Tim menggunakan rapat tertutup untuk mengidentifikasi apa yang perlu ditangani dan bagaimana tim berencana untuk mengatasinya. Saya memandang rapat Sprint Retrospective sebagai rapat ruang ganti tertutup untuk tim Scrum. Di sanalah "inspeksi dan adaptasi" terjadi. Ada banyak cara untuk menjalankan Retrospective.

Saya akan merinci beberapa favorit saya nanti di buku ini. Untuk saat ini, saya akan merinci dasar-dasarnya. Tim pengembangan dan Scrum Master menghadiri rapat tersebut. Beberapa orang berpendapat bahwa Pemilik Produk adalah bagian dari tim dan juga harus hadir. Itu keputusan tim. Jika tim merasa nyaman dengan kehadiran PO, itu tidak masalah. Jika kehadiran PO menyebabkan anggota tim tidak berpartisipasi penuh dalam retrospeksi, PO tidak boleh ikut serta. Pertemuan tidak boleh menjadi sesi pengaduan.

Scrum Master harus memastikan bahwa ada hal positif dalam pertemuan tersebut. Membiarkan tim bersikap negatif tidak menyelesaikan apa pun. Ya, mereka mungkin butuh waktu untuk melampiaskan kekesalan; namun, tujuan Sprint Retrospective adalah untuk memeriksa dan beradaptasi. Batasi jumlah hal buruk yang didiskusikan, soroti hal positif, dan putuskan apa yang harus diubah atau dicoba untuk menjadi lebih baik. Teknik favorit saya untuk digunakan dalam rapat retrospektif adalah "perahu layar" (Gambar 5.13). Scrum Master menggambar perahu layar, dan memberikan catatan tempel kepada tim.

Tim diminta untuk menuliskan pemikiran mereka pada catatan tempel dan menempelkannya pada gambar di salah satu dari tiga tempat berikut:

- Hal-hal baik diletakkan di layar (angin di layar kita).
- Hal-hal buruk diletakkan di jangkar (jangkar menahan kita).
- Hal-hal yang ingin ditingkatkan oleh tim diletakkan di depan perahu. (Ke mana kita akan pergi?)

Gambar 5.13 Perahu Layar

Scrum Master mendorong setiap anggota tim untuk meletakkan sesuatu di perahu, sambil memastikan tidak ada yang mendominasi latihan. Biasanya sarannya adalah setiap orang harus meletakkan setidaknya satu catatan di perahu, tetapi tidak lebih dari tiga. Setelah catatan berada di atas perahu, tim berdiskusi tentang catatan tersebut. Dengan menggunakan teknik "fist-of-five", tim harus memutuskan tindakan yang akan dilakukan. Tindakan tersebut sebagian besar berfokus pada hal-hal yang ada di bagian depan perahu, tetapi terkadang hal-hal akan muncul saat tim mendiskusikan catatan di layar dan di jangkar.

Tujuan dari Pertemuan Retrospektif adalah untuk menghasilkan satu atau dua hal yang akan diubah atau dikerjakan oleh tim pada iterasi berikutnya. Ini adalah tindakan yang akan dilakukan. Jangan mencoba untuk "merebus lautan" atau memperbaiki semuanya sekaligus. Perubahan kecil terakumulasi dan berubah menjadi peningkatan besar dalam kelincahan dan kecepatan. Pada akhir rapat, hasilnya harus diunggah di tempat umum sehingga tim terus diingatkan tentang tindakan yang disepakati.

Jika ada orang dalam tim yang merasa tidak nyaman dengan apa pun yang diunggah, jangan tampilkan tindakan tersebut. Pada awal retrospektif berikutnya, tim meninjau tindakan dan membahas apa yang berhasil, apa yang tidak, dan bagaimana cara untuk melangkah maju. Bab ini membahas tentang artefak Agile dan mengapa artefak tersebut merupakan indikator penting bagi kesehatan keseluruhan proyek Agile.

BAB 6
TANGGUNG JAWAB DAN FUNGSI INTI SCRUM MASTER

Pada titik ini, siapa pun yang membaca buku ini seharusnya sudah memiliki gambaran yang baik tentang apa itu Scrum. Sekarang saya ingin melakukan sesuatu yang sedikit berbeda. Saya akan menunjukkan kepada Anda seperti apa Scrum dengan menggunakan Scrum Master bernama Cassie. Tidak, Cassie bukanlah orang sungguhan. Saya akan menggunakan Scrum Master yang mistis ini dan beberapa skenario yang pernah saya alami atau dengar dari beberapa rekan saya untuk menunjukkan kepada Anda perspektif Scrum Master tentang Scrum.

6.1 MEMBANGUN TIM SCRUM YANG EFEKTIF

Duane duduk di mejanya, dikelilingi buku-buku tentang kerangka kerja Agile dan Japanese armor. Sejujurnya, dia tidak terlalu peduli dengan Japanese armor. Dia telah menjabat sebagai Agile coach untuk sebuah perusahaan pengembangan perangkat lunak menengah selama beberapa tahun. Dia meminta Cassie untuk menjabat sebagai Scrum Master untuk tim yang baru dibentuk. “Jadi, bagaimana tim ini dibentuk?” tanya Cassie. Duane tersenyum dan bersandar di kursinya. “Seperti yang saya yakin Anda tahu, Anda tidak pernah bisa memilih tim sendiri. Saya sangat senang dengan hasil yang dicapai tim ini.

Anda ingin tim tersebut multidisiplin dengan semua kompetensi yang dibutuhkan untuk menyelesaikan setiap Sprint dengan baik”. Cassie terkekeh, “Mengapa begitu banyak kata-kata yang tidak penting, Duane? Bicaralah dalam bahasa Inggris.” “Cassie, pernahkah Anda mendengar pepatah lama tentang membangun tim Scrum?” “Apa itu, Duane?”. Duane tersenyum dan meletakkan tangannya di mejanya. “Tim Scrum yang baik harus memiliki semua keterampilan yang diperlukan untuk menghasilkan perangkat lunak yang berfungsi setiap tahap, dan cukup kecil untuk diberi makan dua pizza besar”. “Dua pizza, Duane. Benarkah?”.

“Yup… tim Scrum terdiri dari tujuh orang. Kevin adalah pimpinan teknis. Dia adalah bintang utama dalam kelompok tersebut. Jack dan Steven adalah pengembang berpengalaman lainnya dalam proyek tersebut. Mike dan Kelly adalah orang baru di perusahaan tersebut dan relatif tidak berpengalaman. Dari apa yang saya pahami, mereka memiliki kemampuan. Mereka hanya butuh waktu di lapangan. Sue dan Ray adalah penguji. Ganesh menangani dokumentasi”.

“Wow, tim Scrum yang hidup dan bernapas. Dan mereka semua milikku,” kata Cassie dengan nada sarkastis. Duane mengabaikannya dan melanjutkan. “Annie adalah Pemilik Produk. Dia telah mengerjakan Backlog selama beberapa minggu ini. Saya akan memperingatkan Anda, dia agak kasar. Terkadang saya pikir dia merasa bahwa dia adalah manajer pengembangan, bukan PO”. “Jadi kapan saya mulai bekerja dengan mereka? Cassie bertanya. “Sekarang,” jawab Duane. “Saya tahu Anda akan bekerja dengan baik bersama orang-orang ini. Tidak ada keraguan dalam benak saya bahwa Anda akan mengubah kelompok

individu ini menjadi tim yang berkinerja tinggi."

Kenyataannya adalah bahwa sebuah tim perlu memiliki keterampilan untuk dapat menghasilkan perangkat lunak yang berfungsi di setiap Sprint. Sebuah tim Scrum perlu bersifat lintas fungsi dan mandiri. Namun, jangan terjebak dalam mitos bahwa setiap orang dalam tim Scrum adalah setara. Sebuah tim Scrum adalah sekelompok individu yang bekerja bersama untuk menghasilkan apa yang telah disepakati. Tim perlu memanfaatkan keterampilan yang mereka miliki atau memperoleh keterampilan yang diperlukan untuk menghasilkan apa yang dibutuhkan. Kuncinya adalah Anda ingin mencoba untuk menjaga tim tetap bersama sebanyak mungkin sehingga mereka dapat belajar dan tumbuh bersama. Keharmonisan tercipta ketika setiap orang dalam tim mengetahui tugasnya dan bagaimana tugas tersebut sesuai dengan dinamika tim.

6.2 KARAKTERISTIK TIM YANG BERKINERJA TINGGI

Menyiapkan Panggung

"Mengapa kita melakukan ini?"

Cassie telah menjadwalkan latihan membangun tim. Salah satu peserta melontarkan pertanyaan itu dan menyeringai malu saat ia melihat dari mana pertanyaan itu berasal.

"*Bekerjalah denganku di sini,"* balasnya. *"Ini akan masuk akal dalam satu atau dua menit."*

Saat memulai dengan tim baru, Cassie suka mengumpulkan mereka dan meminta mereka untuk menceritakan tentang tim yang mereka kagumi. Itu bisa berupa tim yang memenangkan banyak kejuaraan atau harus mengatasi kesulitan besar untuk memenangkan pertandingan besar. Itu bahkan tidak harus berupa tim olahraga. Ia mencari tim yang berkinerja tinggi.

"Oke, apakah semua orang punya tim dalam pikiran?" tanya Cassie.

"Tentu saja!" kata Mike. *"Tim sepak bola sekolah menengahku memenangkan kejuaraan negara bagian."*

Cassie tersenyum *"Apa yang membuat mereka begitu hebat? Apa karakteristik tim itu?"* Ia berjalan ke flip chart dan mengambil spidol.

"Matt, quarterback kami, adalah pemimpin yang hebat." Mike menjelaskan. "Ia tidak pernah ingin kalah dalam hal apa pun. Orang itu adalah manusia paling ulet yang pernah saya temui dalam hidup saya. Sikapnya yang pantang menyerah menular. Saya akan menerobos tembok demi orang itu."

"Jadi, memiliki pemimpin yang hebat itu penting?" Cassie tahu bahwa memiliki pemimpin itu

penting bagi dinamika tim, tetapi dia ingin tim menyatakan fakta ini.

Steve menimpali, *"Tidak harus quarterback yang mengendalikan siapa yang mendapatkan bola, tetapi pemimpin yang menginspirasi tim. Tidak ada tim yang akan mencapai sesuatu yang hebat tanpa seseorang yang memimpin jalan."*

"Hebat!" kata Cassie. Dia berbalik dan menulis *"Pemimpin yang baik"* di flip chart. Dia kembali ke tim dan berkata, *"Apa lagi?"*

Kelly menyela dan berkata, *"Bagaimana dengan akuntabilitas?"*

Cassie menyeringai dan membalas, *"Bagaimana dengan akuntabilitas? Apa maksudmu?"*

"Maksudku, kurasa semua orang dalam tim harus saling bertanggung jawab. Kita harus saling bertanggung jawab."

Di akhir latihan, Cassie berdiri di depan daftar yang kira-kira seperti ini:

- Pemimpin yang baik
- Saling bertanggung jawab
- Kebanggaan
- Eksekusi yang hebat
- Komunikasi yang baik
- Saling percaya
- Benar-benar transparan

"Selamat!" kata Cassie, *"Kamu baru saja menciptakan serangkaian karakteristik yang harus kita, sebagai sebuah tim, tiru."*

Aturan Tim

"Mari kita bicarakan tentang bagaimana tim ini akan bekerja sama." Cassie senang karena tim tampak terlibat dalam rapat dan ingin memastikan bahwa dia memanfaatkannya.

"Bukankah ini seperti kegiatan taman kanak-kanak?" Kevin tampak sedikit terganggu oleh Cassie yang mengangkat topik tersebut. *"Bukankah kita semua orang dewasa di sini?"*

"Saya harap begitu," kata Cassie, *"tetapi bahkan orang dewasa pun butuh pedoman."*

Ganesh angkat bicara. *"Saya setuju bahwa kita harus lebih tahu, tetapi saya jamin bahwa tim ini membutuhkannya."*

"Apa maksudnya itu?" Kevin membalas.

"Katakan saja kamu selalu terlambat untuk rapat," kata Ganesh dengan nada bercanda. *"Maksudku kamu tidak pernah tepat waktu."*

"Oke," kata Kevin sambil memutar matanya.

"Jika tidak ada kesepakatan tim, sulit untuk membuatmu datang ke rapat tepat waktu." kata Ganesh.

"Saya mengerti," kata Kevin. Anda tidak dapat menegakkan aturan yang tidak berlaku. "

"Benar," kata Ganesh. Dia tersenyum dan berkata, *"Tim harus menyepakati peraturan dan hukuman tim."*

"Hukuman?" tanya Kevin yang kebingungan. Cassie merasa lebih baik untuk menyela pada saat ini. *"Ya, hukuman".*

Tim harus memberikan semacam hukuman karena melanggar aturan tim. Tidak ada yang tidak pantas... membeli bagel untuk tim atau semacamnya. Cassie berjalan ke flip chart di depan ruangan dan menulis kata *"tanggung jawab"* dengan huruf besar. *"Memiliki kesepakatan tim menciptakan tanggung jawab bersama,"* kata Cassie.

"Itu adalah aturan yang harus diikuti oleh tim agar kalian lebih efisien dan sukses". Tim mendiskusikan seperti apa daftar aturan tim yang menurut mereka dapat diterima. Setelah beberapa kali berdiskusi dan beberapa kali pemungutan suara, flip chart tersebut memiliki daftar yang terlihat seperti ini:

Aturan Tim

- Berkomunikasi/berdiskusi/berusaha untuk mendapatkan sudut pandang semua orang.
- Menyelesaikan cerita tepat waktu.
- Mendengarkan setiap orang berhak untuk didengarkan. Jangan berbicara di sela-sela pembicaraan orang lain.
- Pemungutan suara tim.
- Tunjukkan penghargaan.
- Berkolaborasi.
- Menghargai bakat satu sama lain secara setara.
- Debat yang membangun.
- Saling menyukai.
- Kepercayaan dan ketergantungan.

Seperti Cassie, saya suka menjalankan latihan seperti ini sebagai semacam lokakarya membangun tim. Dengan membuat anggota tim fokus pada tim yang mereka kagumi, saya

sebenarnya membiarkan mereka memberi tahu saya seperti apa tim yang berkinerja tinggi. seperti. Sekarang tujuannya adalah sesuatu yang mereka tetapkan bersama, bukan sesuatu yang dipaksakan oleh manajer atau Scrum Master kepada mereka. Saat tim bekerja melalui Sprint, saya akan mengingatkan mereka seperti apa visi kolektif mereka tentang tim berkinerja tinggi, dan menanyakan apa yang perlu dilakukan untuk membuat tim saat ini terlihat seperti tim berkinerja tinggi.

Aturan tim, yang juga dikenal sebagai perjanjian kerja, persis seperti namanya. Anda menetapkan aturan, menyetujuinya, dan mengharapkan semua orang untuk menghormatinya. Di sinilah hukuman yang menyenangkan menjadi kuat. Percayalah, tidak ada yang lebih baik daripada mengingatkan pimpinan teknologi bahwa ia perlu membeli donat karena ia pikir ia berada di atas aturan. Aturan tim menciptakan rasa lapangan yang setara. Kita semua bersama-sama dalam hal ini.

6.3 PERENCANAAN SPRINT

Tim sedang mengerjakan rapat Perencanaan Sprint dan Cassie senang dengan cara segala sesuatunya berjalan. Annie telah mengartikulasikan tujuan Sprint dengan jelas. Sekarang saatnya bagi Cassie untuk berbicara.

"Baiklah, Tim, mari kita cari tahu kapasitas kita untuk Sprint."

"Apa yang Anda maksud dengan kapasitas?" tanya Kelly. *"Bukankah kita sudah mengukur cerita-cerita ini dalam rapat penyempurnaan kita?"*

"Ya," kata Cassie, "*tetapi sekarang kita berbicara tentang estimasi, bukan pengukuran. Izinkan saya meluangkan waktu sebentar dan berbicara tentang apa yang kita lakukan dalam rapat ini."* Cassie berjalan ke depan ruangan. Ia merasa bahwa agar rapat ini berhasil, ia harus memastikan bahwa setiap orang memiliki pemahaman yang jelas tentang apa yang tercakup dalam Perencanaan Sprint. Tujuan Perencanaan Sprint adalah untuk menyetujui cerita apa yang akan diselesaikan dalam Sprint ini. Untuk melakukan ini, kita perlu mengetahui kapasitas tim kita dan menguncinya."

"Baiklah, tetapi apa sebenarnya kapasitas itu?" tanya Kelly.

"Sebagai tim, kami melihat berapa banyak hari kerja yang tersedia untuk Sprint ini. Untuk Sprint dua minggu, biasanya sepuluh hari."

"Jadi, delapan puluh jam per anggota tim per Sprint," kata Kevin.

"Tidak." Cassie menyilangkan lengannya, lalu cepat-cepat membukanya agar tidak terlihat defensif. *"Saya berbicara tentang waktu kerja yang sebenarnya, seperti waktu mengetik di papan ketik. Menyisihkan waktu untuk email, rehat kopi, rapat, dan gangguan, Anda secara*

realistis harus merencanakan empat hingga lima jam sehari."

"Dan hilangkan hari-hari yang kami rencanakan untuk tidak berada di sini," tambah Ganesh. *"Saya berencana mengambil cuti beberapa hari minggu depan untuk pergi memancing."*

"Benar," kata Cassie. *"Kami perlu gambaran realistis tentang berapa jam ketersediaan tim yang kami miliki untuk dua minggu ke depan."*

"Jadi, kami mencari tahu berapa jam yang tersedia. Lalu apa?" tanya Kelly.

Cassie tersenyum dan berkata, *"Kemudian kami menguraikan cerita menjadi tugas-tugas, memperkirakan berapa lama tugas-tugas itu akan berlangsung, dan mengikat anggota tim pada tugas-tugas itu." "Jadi, sebaiknya kita memiliki pemahaman yang sangat jelas tentang apa yang perlu kita lakukan,"* kata Kevin.

"Benar sekali! Tim perlu memiliki pemahaman yang baik tentang kriteria penerimaan dan cakupan cerita," kata Cassie. Tim akan mengambil sebuah cerita; ingatlah bahwa kita sedang mencoba untuk memenuhi tujuan Sprint Annie di sini.

Tim kemudian akan membahas cerita tersebut dan menguraikannya menjadi beberapa tugas. Kalian pilih detail yang kalian butuhkan di sini. Lakukan apa yang masuk akal di sini. Jangan terlalu mendetail sehingga kalian menghabiskan lebih banyak waktu untuk memperbarui tugas daripada menulis kode. Kalian memang memerlukan tingkat detail yang mencerminkan bagaimana pekerjaan akan mengalir".

"Bisakah Anda memberi kami contoh?" tanya Ganesh.

"Daripada satu tugas yang bertuliskan Pengembangan, cobalah untuk membuat tugas untuk setiap bagian logis dari proses pengembangan. Para penguji kemudian dapat meniru tugas pengembangan dengan tugas QA sehingga mereka dapat mulai menulis dan menyelesaikan pengujian untuk setiap bagian pengembangan yang sedang dilakukan."

"Tidak benar-benar pengembangan yang digerakkan oleh pengujian, tetapi memungkinkan para penguji untuk terlibat di awal Sprint," kata Ganesh.

Cassie menyela: *"Dan memungkinkan transparansi yang lebih besar tentang apa yang sedang dikerjakan dalam Sprint."* Cassie duduk dan berkata, *"Saya juga ingin berhati-hati terhadap optimisme di sini. Bersikaplah realistis dalam perkiraan Anda. Apa pun yang Anda lakukan, jangan terlalu berkomitmen."*

"Cassie, saya tidak yakin tentang apa yang harus dilakukan untuk menugaskan sebuah cerita. Bisakah Anda memberi kami gambaran umum?" tanya Ray.

"Tentu saja!" kata Cassie. Dia senang bahwa seseorang meminta gambaran umum tentang cara membuat tugas. Ini akan berjalan lebih baik daripada jika dia yang membicarakannya. "Semua orang di tim berpartisipasi. Tentu saja, Kevin, Jack, dan Steven memiliki keahlian terbanyak, tetapi seluruh tim bertanggung jawab untuk mengidentifikasi tugas.

Saya sarankan Anda memperhitungkan pengujian terutama rencana pengujian otomatis. Otomatisasi dan integrasi berkelanjutan sangat penting bagi Scrum. Perangkat lunak yang berfungsi tidak ada gunanya jika tidak dapat dikirimkan dengan cepat. Untuk memangkas waktu yang dibutuhkan untuk mengirimkan perangkat lunak yang berfungsi, kita perlu dapat mengujinya secara menyeluruh tetapi cepat. Keindahan tersembunyi dari otomatisasi pengujian adalah bahwa rencana pengujian otomatis dapat ditulis saat kode sedang ditulis."

"Wah, saya tidak pernah memikirkan itu," kata Sue.

"Benar," jawab Cassie. *"Saya benci ketika penguji dibanjiri pekerjaan di akhir Sprint. Ini adalah cara untuk membantu mengatasinya. Hal lain yang perlu dipikirkan adalah demo akhir Sprint. Selalu pikirkan itu dan berikan sedikit kapasitas."*

"Jadi, kami berusaha setransparan mungkin tentang pekerjaan yang harus dilakukan," kata Steve.

"Ya, dan hanya untukmu, Steve jangan terlalu teknis. Kamu harus berbicara tentang kode pada tingkat tinggi. Jangan sampai Anda terjebak dalam hal-hal yang tidak penting sampai pekerjaan yang sebenarnya dimulai." Cassie melihat ke seluruh ruangan dan berkata, "Pastikan tugas Anda memenuhi Definisi Selesai. Ingat, jika cerita tidak memenuhi persyaratan DoD, maka cerita itu belum selesai.

Perencanaan Sprint adalah tempat orang-orang nyata berkomitmen pada pekerjaan nyata. Setelah semua pembicaraan tentang hal-hal umum saat menentukan ukuran cerita, Perencanaan Sprint adalah tentang menjadi sangat spesifik tentang apa yang Anda rencanakan untuk dilakukan dan berapa lama waktu yang dibutuhkan. Butuh waktu cukup lama untuk menyelesaikan rapat Perencanaan Sprint mulai dari empat hingga enam jam, tergantung pada kematangan tim.

Saya tidak akan berbohong; pada awalnya, rapat-rapat ini akan terasa seperti siksaan. Anggota tim tidak akan datang dengan persiapan, dan akan butuh waktu lebih lama untuk menugaskan cerita daripada yang Anda kira. Sama seperti yang saya lakukan dengan tim lintas negara saya, saya tidak menggurui dan memberi tahu anggota tim bahwa mereka harus datang dalam keadaan siap untuk menguraikan cerita-cerita ini dan memperkirakan tugas. Setelah beberapa Sprint, mereka akan mengerti.

6.4 MEMBANGUN DEFINISI SELESAI DALAM TIM SCRUM

Cassie masuk ke ruangan sambil membawa setumpuk catatan tempel dan pulpen, sambil menunjukkan ekspresi percaya diri di wajahnya yang menutupi fakta bahwa ia takut

dengan apa yang akan terjadi. Tim berkumpul di sekitar meja besar. Ekspresi wajah mereka bervariasi dari kegembiraan hingga kebosanan.

"Jadi, apa artinya selesai?" tanya Cassie. *"Itu artinya selesai."* jawab seorang anggota tim.

"Oke, apa artinya... selesai?"

Mike, salah satu pengembang junior, angkat bicara: *"Permainan yang aneh, tetapi saya akan bermain...Bagaimana dengan kode lengkap?"*. Ganesh menyeringai dan berkata, *"Apakah kode lengkap berarti kode tersebut diberi komentar dan dibersihkan, atau hanya dikompilasi?"* Ia telah berkecimpung dalam bisnis perangkat lunak selama dua puluh tahun dan memahami pola pikir pengembangan meskipun ia terutama menulis dokumentasi untuk tim ini.

"Jika kalian berhasil, kalian akan menghasilkan begitu banyak kode sehingga kita tidak akan pernah mengujinya dan mendokumentasikannya." Mike membalas, *"Ganesh, kita semua tahu bahwa pelanggan tidak membaca dokumen itu. Mereka membeli kode yang kita tulis". "Jangan sampai aku menghukum kalian berdua."* Cassie bercanda. *"Aku ingin mengingatkan semua orang di ruangan ini bahwa mereka berada di tim yang sama. Aku ingin kalian semua mengesampingkan prasangka apa pun yang kalian miliki tentang peran dan tanggung jawab untuk saat ini dan memikirkan apa arti selesai bagi tim ini".*

Dia menyipitkan matanya dan mencondongkan tubuhnya ke seberang meja untuk menyampaikan maksudnya. *"Sebagai catatan, selesai berarti kalian sebagai tim telah memberikan fungsionalitas yang berfungsi. Bukan kode yang dikompilasi atau terlihat cantik, tetapi perangkat lunak yang berfungsi, teruji, terdokumentasi, dan fungsional. Pelanggan tidak membeli kode mereka membeli perangkat lunak yang membantu mereka memecahkan masalah rumit yang muncul dari pekerjaan sehari-hari mereka."* Dia menunduk dan menyadari bahwa lengannya gemetar. Dia dengan cepat menyilangkan lengan di depannya. Dia ingin menunjukkan otoritas terutama karena dia sangat takut. Ini adalah pertama kalinya dia menantang tim, dan dia tidak yakin apa hasilnya nanti.

Setelah keheningan yang terasa sangat lama, Kevin (pengembang paling berpengalaman dalam tim) angkat bicara dan berkata, "Baiklah, Miss Scrum Master, apa yang Anda ingin kami lakukan?". "Kalian semua punya catatan tempel dan pulpen," kata Cassie dengan lega. *"Saya ingin semua orang menulis sesuatu yang ingin mereka lihat sebagai item daftar periksa pada Definisi Selesai dan menempelkannya pada flip chart yang terletak di bagian depan ruangan".* Tim mulai menuliskan ide-ide mereka dan sesuatu yang ajaib terjadi. Mereka mulai berbicara satu sama lain. Setelah setiap anggota tim menempelkan catatan tempel pada flip chart, Cassie memimpin diskusi dan pemungutan suara lima suara untuk setiap ide. Tim kemudian melanjutkan putaran lain dengan menuliskan ide-ide pada catatan tempel dan menambahkannya ke flip chart.

Di akhir pertemuan, mereka memiliki sesuatu yang dapat mereka kerjakan bersama: Definisi Selesai yang sederhana. Keponakan saya Max tidak pernah menjadi tipe orang yang saya anggap "*tergila-gila pada makanan*." Dia sama sekali tidak tertarik pada makanan, yang

merupakan sesuatu yang tidak dapat saya pahami. Saya bangun di pagi hari sambil memikirkan apa yang akan saya makan hari itu. Begitu saya makan, saya sudah memikirkan makanan saya berikutnya. Max memiliki pola pikir yang berbeda. Ketika dia masih sangat kecil, yang dapat saya ingat hanyalah dia berdiri di kursi tingginya dan berteriak, *"Selesai!"* Tentu saja, dia belum selesai.

Ibunya akan mengatakan sesuatu yang manis seperti "Sayang, makanlah sedikit lagi." *"Selesai!"* "Tapi kamu belum menghabiskan" *"Selesai!"* Max berteriak bahwa dia sudah selesai ketika dia pikir dia sudah selesai lebih banyak berhubungan dengan bagaimana tim pengembangan bertindak daripada yang benar-benar ingin saya akui. Jika dibiarkan beroperasi tanpa kendali, semua jenis hal akan "dikesampingkan." Anda tahu, disingkirkan untuk diselesaikan nanti sehingga tim dapat menyelesaikan hal-hal yang besar dan penting. Itu dilakukan dengan niat terbaik. Tim benar-benar berencana untuk mengerjakan semua pekerjaan yang tertunda itu, tetapi kenyataannya adalah selalu ada lebih banyak pekerjaan daripada waktu yang dapat diselesaikan oleh tim.

Pekerjaan yang tertunda itu sebenarnya tidak bisa dianggap remeh. Pekerjaan itu mungkin bukan membuat fitur, tetapi mencakup hal-hal seperti pengujian menyeluruh dan mungkin infrastruktur. Pekerjaan ini biasanya menjadi utang teknis. Utang yang harus dibayar di masa mendatang. Definisi Selesai sangat penting bagi keberhasilan tim Scrum karena memaksa tim untuk memastikan tidak ada yang terlupakan atau ditunda ke lain waktu. Ketika tim bekerja untuk memenuhi DoD, semua orang mulai dari Pemilik Produk hingga pemangku kepentingan dapat yakin bahwa tim tersebut menghasilkan fungsionalitas yang berpotensi dapat dikirimkan.

Beberapa tim menggunakan DoD untuk membuat tugas untuk cerita. Tidak ada yang dapat dilakukan sampai hal itu dibangun, diuji, didokumentasikan, dan dapat dibuktikan. Jika tim bekerja untuk memenuhi Definisi Selesai, semua orang memahami bahwa ini masalahnya. Tidak perlu dipertanyakan lagi bahwa cerita itu telah selesai. Jika belum selesai, maka cerita itu belum selesai. Tindak lanjut. Selesai!

Pentingnya Definisi Siap dalam Tim Scrum

"Baiklah, kurasa aku paham konsep Definisi Selesai, tapi apa Definisi Siap?" tanya Steve. Cassie terkekeh pelan. Definisi Selesai dan Definisi Siap secara konseptual serupa, tetapi sangat berbeda dalam praktik. *"Baiklah, Steve,"* katanya, *"Jika kamu akan terjun payung, apakah kamu ingin memastikan bahwa kamu sudah memeriksa ulang bahwa kamu memiliki parasut utama dan cadangan?"*. Steve menggaruk kepalanya dan berkata, *"Baiklah, ya. Aku akan memastikan bahwa aku memiliki parasut, tentu saja".* Jack ikut campur dalam percakapan: *"Kamu tidak hanya ingin memastikan bahwa kamu memiliki kedua parasut, kamu juga ingin memastikan bahwa mereka dikemas dengan benar."* Dia memutar matanya dan berkata, *"Hal terakhir yang kamu butuhkan adalah dua parasut yang tidak berfungsi".*

Kelly menambahkan, *"Kamu juga membutuhkan salah satu dari benda yang mereka kenakan di pergelangan tangan mereka. Apa namanya, altimeter atau semacamnya?". "Jadi kamu tahu kapan harus menarik tali parasut,"* kata Cassie*. "Apa yang kalian gambarkan di sini adalah Definisi Siap. Sama seperti Anda tidak ingin melompat dari pesawat tanpa memastikan*

semuanya siap untuk membawa Anda kembali ke daratan dengan selamat, Anda ingin memastikan bahwa sebuah cerita siap untuk dibawa ke Sprint sebelum melakukannya."

Saya dan istri saya menjalankan program remaja di gereja kami. Kami terutama bekerja dengan anak-anak yang lebih muda, terutama karena mereka menganggap lelucon saya lucu. Biasanya, di suatu waktu di malam hari, kami mengadakan semacam permainan lari estafet. Permainan ini memberi anak-anak jalan keluar untuk energi mereka yang tak terbatas dan sangat menyenangkan bagi anak-anak dan orang dewasa. Saya biasanya memulai perlombaan seperti ini:

"Siap?"

"Siap?"

"Jangan mulai dulu!"

Semua orang berlari cepat sejauh tiga atau empat langkah. Begitu mereka menyadari apa yang telah saya lakukan, biasanya akan ada banyak tawa dan protes pura-pura tentang start saya yang salah. Kami menempatkan semua orang kembali pada tempatnya dan saya mencoba memulai kembali perlombaan…

"Siap?"

"Siap?"

"Jangan mulai dulu!"

Sekali lagi, anak-anak begitu bersemangat untuk memulai lomba sehingga mereka tidak mendengarkan, dan mulai berlari. Saya biasanya melakukan proses ini tiga atau empat kali sebelum anak-anak menyadari kebodohan saya.

Saya melihat hal yang sama dengan tim Scrum. Bukan masalah kebodohan saya, tetapi fakta bahwa mereka begitu peduli dengan memulai sehingga lupa memastikan bahwa cerita siap untuk dimulai. Bukan hal yang menyenangkan ketika cerita ditarik ke Sprint dan seminggu kemudian, Anda menyadari bahwa tugas prasyarat belum selesai. Memiliki Definisi Siap mencegah cerita dikerjakan sebelum siap. Definisi Siap memastikan Anda dapat menyiapkan, mengatur, dan menjalankan cerita pengguna.

Cerita

"Hai Annie, apa kabarmu hari ini? Apakah kamu siap untuk melihat beberapa cerita?" Cassie tampak bersemangat dan tak sabar untuk bekerja dengan Product Owner. *"Hai Cassie,"* kata Annie. *"Saya ingin memberi tahu Anda bahwa saya siap mengerjakan cerita, tetapi saat ini saya sedang frustrasi." "Uh oh… tidak ada yang suka PO yang frustrasi."* Cassie menggeser

kacamatanya kembali ke tempatnya di hidungnya. *"Apa saja kendala Anda?" "Saya rasa tim tidak mengerti apa yang saya minta mereka berikan. Maksud saya, mereka orang-orang pintar. Saya suka menganggap diri saya orang pintar, dan pelanggan telah menjelaskan dengan jelas apa yang mereka inginkan kepada saya. Ketika kami semua masuk ke ruangan dan saya mencoba menjelaskannya kepada tim… rasanya kami semua kehilangan sesuatu."* Annie menundukkan kepalanya.

"Kami mulai berbicara tentang sebuah cerita dan butuh waktu berjam-jam untuk membuat mereka mengerti apa yang ingin saya buat. Kalau saja saya bisa membuat mereka mengerti." Cassie memiringkan kepalanya dan tersenyum. *"Hanya itu yang membuatmu frustrasi?" "Ya, tentu saja,"* kata Annie dengan nada sedikit terkejut. *"Kebetulan saja aku sangat ahli dalam menulis cerita, jadi mari kita mulai." "Aku tidak tahu ilmu hitam macam apa yang kau kuasai di sana, tetapi kau menarik perhatianku."* Cassie terkekeh, *"Itu bukan ilmu hitam, hanya praktik menulis cerita yang baik."* Cassie duduk dan melipat tangannya di depan dada. *"Siapa yang menulis cerita?" "Ya, aku yang menulis,"* kata Annie. "*Annie, maksudmu kau yang menulis semua cerita?" "Ya, aku yang menulis. "Kenapa?" "Karena aku tidak ingin tumpukan sampah memenuhi Backlog." "Hmmm."*

Cassie berusaha keras untuk memasang ekspresi serius di wajahnya. *"Kamu benar… kurang lebih begitu. Pemilik Produk bertanggung jawab penuh atas konten Backlog, tetapi siapa pun seharusnya bisa menulis cerita. Kamu tidak perlu terlalu khawatir tentang siapa yang menulisnya. Kita perlu bekerja sebagai tim. Untuk saling membantu. Kamu sama sekali tidak ingin ada sampah di Backlog-mu. Itu tidak berarti kamu harus menulis apa pun. Itu berarti kamu memiliki kendali atas apa yang diterima di Backlog.""Aku benar-benar tidak melihat apa masalahnya di sini."* kata Annie. *"Siapa yang peduli jika aku menulis semua cerita?" "Baiklah, biar kujelaskan begini,"* kata Cassie. *"Bagaimana jika salah satu anggota tim mengatakan kepadamu bahwa menulis cerita bukanlah tugas mereka." "Maksudmu mereka benar-benar akan memberiku kalimat 'itu bukan tugasku'?"* Annie mulai membuat tanda kutip dengan jarinya. *"Ya." "Ya ampun, kedengarannya mengerikan."*

Annie menutup mulutnya dengan tangan dan memasang ekspresi terkejut di wajahnya. Cassie mencondongkan tubuh ke depan dan berkata, *"Sejujurnya, itu tidak sopan. Tidak sopan terhadap tim ANDA. Menurut saya, cerita akan lebih baik jika ditulis secara kolaboratif." "Saya tidak pernah berpikir seperti itu. Saya rasa saya perlu memberi tahu tim bahwa saya butuh bantuan untuk menulis cerita." "Silakan, Nak!"* Cassie mengangkat tangannya untuk tos, tetapi menariknya kembali ketika Annie menatapnya dengan tidak nyaman. *"Baiklah, mari kita bicarakan tentang bagaimana beberapa cerita Anda ditulis." "Saya tidak sabar,"* jawab Annie. *"Mari kita lihat cerita dengan prioritas tertinggi di Backlog Anda. Judulnya adalah 'Buat Koneksi SSL ke Server Tertentu.'" Cassie menurunkan kembali kacamatanya dan menatap Annie dari balik kacamatanya. "Apakah Anda tahu apa maksudnya?" "Apa?" "Tidak ada…" "Oh, ayolah!"* seru Annie. *"Saya akan memberi tahu Anda apa yang sebenarnya Anda dapatkan." "Tidak juga,"* kata Annie. Judul cerita harus ringkas tetapi pada saat yang sama mengomunikasikan nilai pelanggan.

Sebagai pelanggan, saya benar-benar tidak peduli apakah Anda menggunakan SSL, atau

teknologi apa pun dalam hal ini. *"Yang Anda pedulikan hanyalah apakah kami menyediakan koneksi aman ke server?"* tanya Annie. *"Bingo! Itu judul cerita Anda. Dengan cara ini, siapa pun pelanggan, pemangku kepentingan internal, anggota tim, atau bahkan CEO perusahaan dapat memahami nilai apa yang akan dihasilkan cerita ini. Anda juga ingin judulnya mudah dicari." "Agar saya dapat menemukannya dengan mudah?"* tanya Annie. *"Benar. Tidak ada salahnya memikirkan hal-hal semacam itu di awal."* Cassie membahas cerita itu lebih dalam dengan mengatakan, *"Berikutnya kita memiliki pernyataan kanonik." "Saya butuh sedikit bantuan untuk yang satu itu,"* kata Annie. *"Saya tidak mengerti mengapa perlu menulis deskripsi dalam format ini."* Cassie duduk kembali di kursinya.

"Anda menjelaskan untuk siapa cerita itu ditujukan, apa yang dijelaskan cerita itu, dan yang terpenting manfaat apa yang diperoleh pengguna dengan menambahkan nilai yang baru saja Anda jelaskan. Ingat, Anda mencoba membuat tim melihat ini melalui mata pelanggan. Menggunakan pernyataan kanonik memudahkan untuk melakukan ini. Pernyataan itu mengomunikasikan untuk siapa perubahan itu ditujukan, perubahan apa, dan mengapa perubahan itu merupakan hal yang baik. Apakah itu masuk akal?" "Kurasa begitu," kata Annie. Cassie melipat tangannya di depannya dan berkata, *"Tantanganku kepadamu adalah kamu tidak mengatakan apa yang kamu inginkan. Kamu mengatakan apa yang kamu inginkan."* Mata Annie berbinar. *"Karena tim memutuskan bagaimana mereka memberikan nilai. Saya hanya harus menjelaskan dengan jelas tentang apa yang saya minta."* "Itu benar! Anda mulai memahaminya di sini."

Sekarang mari kita bahas tentang persona." "*Persona? Apa itu?"* tanya Annie. *"Semua yang kami lakukan melibatkan seseorang yang mengonsumsi hasilnya. Bayangkan karakter yang merupakan perwujudan 'pengguna' Anda dan beri mereka nama. Itulah persona."* Annie menggaruk kepalanya dan berkata, *"Membuat orang palsu?" "Benar sekali!"* jawab Cassie. *"Kami membutuhkan tim untuk melihat cerita tidak hanya dari sudut pandang pelanggan, tetapi juga dari sudut pandang pelanggan yang benar. Anda tahu pelanggan yang benar-benar akan menggunakan produk kami."* Annie memikirkan apa yang baru saja dikatakan Cassie sejenak dan berkata, *"Jadi, dengan berpura-pura, kami benar-benar membuat tim berpikir seperti pelanggan dan membangun sesuatu yang ingin mereka gunakan."*

"Saya suka mengatakan membangun sesuatu yang tidak sabar untuk dimiliki pelanggan," Cassie terkekeh. Dia senang bahwa Annie mulai melihat nilai persona. "Dapatkah Anda melihat bagaimana persona dapat menjadi alat yang ampuh*?" "Oh ya,"* seru Annie. "*Saya ingin duduk bersama tim dan mengembangkan beberapa persona sekarang juga."* Menulis cerita pengguna bisa menjadi keterampilan yang sulit dikuasai. Cerita harus cukup kecil agar sesuai dengan Sprint tetapi mengandung cukup nilai untuk berdiri sendiri. Idenya adalah untuk membuat tim Scrum berbicara tentang persyaratan (baik fungsional maupun nonfungsional) yang diperlukan untuk menciptakan sepotong nilai yang dapat diinstal. Kebanyakan orang yang saya ajak bicara menggambarkan cerita pengguna sebagai potongan kecil fungsionalitas. Saya lebih suka menyebutnya unit fungsionalitas yang dapat dikirimkan.

Kebanyakan tim yang pernah bekerja dengan saya dapat mencapai titik di mana mereka dapat menghasilkan sesuatu yang dapat mereka demo setiap dua minggu. Tujuan sebenarnya

adalah memiliki fungsionalitas yang siap untuk didemokan dan diinstal di akhir setiap Sprint. Ini dapat menjadi pengubah permainan bagi tim, karena mereka perlu memikirkan kembali cara mereka mengembangkan perangkat lunak. Tim sering kali perlu meninjau kembali cara mereka menulis, mencari sumber, mengontrol, dan membangun kode agar dapat mengirimkannya dengan irama ini. Juga menjadi jelas bagi para penguji bahwa otomatisasi pengujian sangat penting. Pengujian perlu dilakukan sedini mungkin dalam Sprint, dan kasus pengujian harus diotomatisasi sebanyak mungkin. Semua itu perlu tercermin dalam cerita pengguna.

Cerita harus selalu ditulis dari sudut pandang pelanggan. Saya suka pengembang. Saya tidak ingin ada yang berpikir bahwa saya punya masalah dengan perilaku tim Scrum, tetapi saya tahu ini fakta. Begitu saya membiarkan tim Scrum kehilangan fokus pada pelanggan, mereka mulai berbicara tentang detail pengkodean. Persona adalah alat yang ampuh yang dapat digunakan oleh Scrum Master dan Pemilik Produk untuk menjaga tim tetap fokus pada perspektif pelanggan. Saya akui bahwa seluruh konsep ini tampak konyol pada awalnya. Persona adalah pengguna buatan dari tipe tertentu yang akan membantu tim mengembangkan perangkat lunak yang tidak... generik. Dengan risiko dianggap terlalu sederhana, Anda membuat sekelompok pengguna imajiner dan menulis perangkat lunak untuk mereka.

Persona ini mewakili karakteristik pengguna sebenarnya, jadi mereka tidak sepenuhnya dibuat-buat. Ya, saya serius. Perangkat lunak yang diproduksi oleh tim Scrum biasanya digunakan oleh lebih dari satu kategori orang. Menggunakan persona memungkinkan kita untuk melihat apa yang kita bangun dari perspektif pengguna. Mereka memaksa kita untuk menggunakan pendekatan yang lebih berpusat pada pengguna dalam cara kita melakukan sesuatu untuk mengutamakan pengguna. Dengan memahami bagaimana fitur tertentu memengaruhi Bob sang operator, Jerry sang teknisi otomasi, dan Fred sang sysprog MVS, kita sebenarnya dapat lebih memahami konsekuensi dari pilihan desain. Persona akan membantu tim Scrum untuk lebih memahami nilai cerita yang mereka kerjakan.

Penyempurnaan Tunda

Annie dan tim berkumpul di ruang konferensi untuk rapat Penyempurnaan Tunda. Cassie, bersemangat seperti biasa, bertanya kepada seluruh hadirin, “Apakah semua orang familier dengan rapat Penyempurnaan? Dengan kata lain, apakah kalian tahu mengapa kalian semua ada di sini?”Ganesh angkat bicara, “Kami sedang menyiapkan cerita… benar?” “Benar, Ganesh,” kata Cassie. Salah satu hal yang ingin kami lakukan di sini adalah memenuhi Definisi Siap sehingga cerita dapat ditarik ke Sprint. Apa lagi?” ,“Saya tidak tahu,” kata Ganesh. “Baiklah, apakah ada yang punya ide?” tanya Cassie. “Anda ingin menyiksa kami dengan lebih banyak rapat?” tanya Ray. Cassie mencatat dalam benaknya bahwa Ray yang biasanya pendiam mulai berbicara.

Dia sedikit khawatir karena Ray tidak melihat nilai dari rapat Refinement. “Ray, saya bisa mengerti mengapa Anda mungkin berpikir ini hanya rapat biasa, tetapi Anda dan tim perlu memahami bahwa rapat Refinement itu penting dan bahkan berharga. Di sinilah Product Owner memperkenalkan cerita yang dapat dikerjakan dalam dua atau tiga Sprint berikutnya.

Seperti yang dikatakan Ganesh, Definisi Siap harus terpenuhi; namun, hal-hal lain terjadi dalam rapat Refinement. Kalian perlu menyempurnakan kriteria penerimaan cerita sehingga semua orang jelas tentang apa yang harus disampaikan. Cerita itu harus diestimasikan." "Itu berarti kita menetapkan poin cerita," kata Kelly. "Itu benar. Tim perlu mengukur cerita dengan estimasi awal. Sebelum itu terjadi, kalian perlu mencari tahu cakupan ceritanya." "Siapa sebenarnya 'kalian'?" tanya Jack, membuat tanda kutip udara dengan jari telunjuk dan jari tengah kedua tangannya.

Cassie membalas, "Tim dan Pemilik Produk. Memahami cakupan cerita sebelum memperkirakannya adalah hal yang masuk akal. Cerita tersebut mungkin perlu dipecah. Cerita tersebut mungkin perlu direfaktor. Annie mungkin memutuskan bahwa cerita khusus ini terlalu mahal dan menurunkan prioritasnya." "Wah, apa maksudnya?" tanya Sue. "Itu berarti saya mungkin memprioritaskan cerita dengan cakupan yang lebih kecil daripada yang ini." Kata Annie. "Kami ingin memberikan nilai yang sebesar-besarnya. Terkadang masuk akal bagi bisnis untuk menyelesaikan sejumlah cerita yang lebih kecil." "Daripada sesuatu yang besar, membengkak, dan mahal," kata Steve. Rapat penyempurnaan sangat berharga.

Di sanalah inti permasalahan. Di mana tim dan Pemilik Produk berkumpul dan menyusun cerita. Biasanya, Penyempurnaan Backlog harus dilakukan setiap minggu dan memakan waktu satu hingga dua jam. Tujuan rapat adalah untuk memperkenalkan cerita yang menjadi kandidat untuk beberapa Sprint berikutnya. Ketika semua orang berkumpul dan mulai membicarakan cerita tertentu, beberapa hal akan menjadi jelas. Pertama, apakah ceritanya terlalu besar? Tidak ada cerita yang boleh mengambil lebih dari seperempat kecepatan Sprint tim. Itulah aturan yang saya ajarkan kepada tim untuk dipatuhi. Tim Scrum harus berusaha membuat cerita sekecil mungkin. Cerita kecil memiliki manfaat besar:

- Cerita menghilangkan variabilitas. Oke, akan selalu ada variabel yang muncul selama Sprint. Menjaga cerita tetap kecil membantu mengendalikan dan menangani variabilitas.
- Cerita lebih mudah diperkirakan.
- Cerita kecil lebih mudah dipahami.
- Lebih mudah menulis kriteria penerimaan untuk cerita yang lebih kecil.

Cobalah untuk membuat cerita pengguna sekecil mungkin, tetapi tidak terlalu kecil sehingga menjadi tugas. Menemukan keseimbangan akan terasa tidak nyaman pada awalnya. Mungkin diperlukan kerja sama seluruh tim untuk mencapai tujuan. Tim mungkin perlu menjadi kreatif, tetapi jika mereka termotivasi, hal itu dapat dilakukan. Tugas apa pun dapat dipecah menjadi langkah-langkah yang lebih kecil yang lebih mudah dikelola. Saya tidak peduli apakah Anda berbicara tentang bisnis atau penurunan berat badan atau belajar cara bermain gitar.

Jika Anda ingin menurunkan berat badan 100 pon, lakukan setiap kali makan, setiap kali latihan. Fokus pada tugas di depan Anda dan lakukan yang terbaik. Pada akhirnya, itu bukanlah kesalahan tim. Waterfall mendorong tim untuk menyembunyikan hal-hal yang buruk. Scrum akan mengungkap hal-hal yang buruk... dengan cepat. Keindahannya adalah bahwa setelah hal-hal yang buruk terungkap, tim dapat dengan mudah menyingkirkannya karena mengikuti metodologi Agile memerlukan upaya terus-menerus untuk menjadi lebih baik.

Pelajari disiplin untuk memecah cerita dan menyelesaikannya dalam Sprint.

6.5 KRITERIA PENERIMAAN

"Anda perlu menganggap kriteria penerimaan sebagai kontrak antara Anda, PO, dan tim tentang pekerjaan yang harus dilakukan." "Sebuah kontrak? Bukankah itu agak kaku?" tanya Annie. "Bukankah itu agak berlebihan?" Cassie menjawab, "Tidak. Kriteria penerimaan adalah cara terbaik bagi Anda untuk memastikan bahwa tim memahami apa yang Anda harapkan dari mereka." Wajah Annie berseri-seri dan dia berkata, "Jadi kita bisa menghilangkan respons 'Oh Anda juga menginginkan itu' dari tim." "Benar sekali!" kata Cassie. "Kriteria penerimaan yang baik memberi Anda dan tim pemahaman yang lengkap tentang seperti apa kesuksesan itu. Setelah membaca kriteria penerimaan, tim harus memiliki pemahaman yang kuat tentang tujuan cerita.

Kriteria tersebut harus merinci beberapa kasus penggunaan yang berada dalam cakupan cerita, serta yang tidak." "Bagaimana saya harus melakukannya?" Cassie dapat merasakan lampu pijar menyala pada titik ini. "Anda perlu fokus pada hasil yang benar atau yang diharapkan. Bisa sesederhana perintah dan output yang diinginkan atau kasus penggunaan kompleks yang menunjukkan fungsionalitas yang diinginkan. Intinya adalah Anda perlu berasumsi bahwa jika tidak ada, Anda tidak akan mendapatkannya." Annie menimpali. "Jadi ini kesempatan saya untuk merinci apa yang saya inginkan." "Dan memastikan bahwa semua orang benar-benar memahami apa yang ingin kita lakukan," kata Cassie.

Ia melanjutkan, "Jika sebuah cerita memiliki kriteria penerimaan yang baik, siapa pun dalam tim harus dapat melihat cerita tersebut enam bulan kemudian dan memahami fungsionalitas dan nilai apa yang akan diperoleh saat cerita tersebut diimplementasikan." Kriteria penerimaan yang baik menghilangkan segala ketidakjelasan atau ketidakpastian dari cerita. Saya melatih Pemilik Produk dan tim untuk membangun kriteria penerimaan menggunakan kasus penggunaan dan hasil. Hasilnya tidak boleh berupa daftar kondisi yang dibuat-buat yang harus dipenuhi tanpa tujuan tertentu. Kriteria tersebut harus berfokus pada hasil yang diinginkan yang akan dihasilkan oleh cerita pengguna. Jika kriteria tersebut tidak termasuk dalam kriteria penerimaan, maka kriteria tersebut tidak diperlukan untuk penyelesaian cerita. Sesederhana itu.

Persyaratan

"Persyaratan masuk ke dalam cerita pengguna. Itulah cara kami melakukannya di Scrum." Cassie menanggapi beberapa pemangku kepentingan dan eksekutif yang memiliki beberapa pertanyaan tentang cara kerja Scrum. "Annie, Pemilik Produk kami, bertanggung jawab untuk mengumpulkan persyaratan. Itulah sebabnya dia berbicara kepada Anda semua dan pelanggan untuk mencari tahu apa yang harus kami bangun. Apa yang dia kumpulkan diubah menjadi cerita pengguna potongan kecil nilai yang dikerjakan tim selama iterasi. Cerita pengguna menggambarkan persyaratan pada tingkat yang sangat tinggi, dengan mengatakan, 'Sebagai X (pengguna), saya memerlukan Y (fungsionalitas) sehingga saya dapat Z (memperoleh beberapa jenis nilai).' Tim dan Pemilik Produk menyempurnakan cerita dan, akhirnya, lebih banyak detail dan kriteria penerimaan akan muncul.

Kriteria penerimaan merinci persyaratan yang sebenarnya apa yang perlu disampaikan tim sehingga Annie dapat menerima cerita tersebut dan mendemonstrasikannya pada rapat Tinjauan Sprint." Seberapa sering Anda berkata kepada diri sendiri "hal ini akan jauh lebih baik jika melakukan ini?" Persyaratan adalah apa yang diinginkan pelanggan dan pemangku kepentingan untuk menjadi kenyataan. Terserah kepada Pemilik Produk dan manajemen produk untuk mengumpulkan persyaratan ini dan memprioritaskannya. Memang, ini bisa jadi sulit karena beberapa alasan.

Sering kali, pelanggan dan pemangku kepentingan tidak mengerti apa yang mereka minta. Persyaratan berubah sangat cepat. Tidak, saya tidak mengatakan bahwa pelanggan dan pemangku kepentingan itu bodoh. Tidak ada yang lebih jauh dari kebenaran. Yang saya katakan adalah bahwa persyaratan bisa sangat rumit dan, seperti halnya arsitektur, muncul seiring waktu. Menambah kerumitan ini, persyaratan dapat berubah dengan cepat. Itu berarti bahwa apa yang dibutuhkan pelanggan saat ini dapat berubah secara radikal minggu depan, atau akhir iterasi tim saat ini. Scrum tidak mengharuskan semua persyaratan didefinisikan di muka. Persyaratan diharapkan berubah saat tim beriterasi. Itulah salah satu alasan mengapa setiap guru Agile yang baik akan memberi tahu tim Scrum untuk membuat iterasi sesingkat mungkin.

Estimasi

"Baiklah, tim," kata Cassie, "Kita berada dalam ritme yang baik di sini." Ia mengadakan sesi Penyempurnaan Backlog dan merasa senang dengan cara kerja tim. Mereka baru saja selesai mendiskusikan seluk-beluk cerita tertentu. Sekarang saatnya untuk menentukan ukurannya. Ia membagikan satu set kartu kepada setiap anggota tim; setiap kartu memiliki nomor yang tertulis di atasnya. "Kita memiliki beberapa masalah dengan penentuan ukuran hingga saat ini, jadi saya ingin mencoba sesuatu yang baru. Saya telah memberikan kalian semua tujuh kartu. Setiap kartu memiliki nomor di atasnya. Angka-angka ini disebut deret Fibonacci satu, dua, tiga, lima, delapan, tiga belas, dan dua puluh. Saya suka menggunakan deret Fibonacci karena angka-angka tersebut benar-benar tidak saling berhubungan, tetapi mereka berkorelasi dengan baik dengan ukuran kaus.

Ekstra kecil, kecil, sedang, besar, ekstra besar, ekstra-ekstra-besar, dan terlalu besar untuk Sprint." "Apakah kita benar-benar membutuhkan semua ini, Cassie?" kata Kevin. "Bukankah sebuah cerita hanya pekerjaan sehari-hari?" "Kevin, itu akan membuatmu mendapat masalah," kata Cassie. "Manusia, spesies yang kita semua miliki, sangat buruk dalam memperkirakan berapa lama waktu yang dibutuhkan untuk melakukan sesuatu." "Bukankah itu benar," kata Mike. "Setiap kali saya melakukan sesuatu di rumah, saya memberi tahu istri saya bahwa itu akan memakan waktu lima belas menit dan akhirnya memakan waktu dua jam." Cassie mengangkat alisnya dan bertanya, "Mengapa begitu, Mike?" Mike menjawab, "Yah, saya mengerjakan sesuatu dan menemukan bahwa ada lebih banyak hal yang harus dilakukan daripada yang saya pikirkan."

"Tepat sekali," seru Cassie. "Manusia sangat buruk dalam memperkirakan berapa lama waktu yang dibutuhkan untuk melakukan sesuatu, tetapi sangat pandai membandingkan berbagai hal." "Bagaimana?" Mike jelas bingung. "Baiklah, Mike, dengan menggunakan kartu-kartu di depanmu, berapa angka kapal induk?" "Eh…. Tiga belas?" kata Mike. "Bukankah kapal

induk lebih seperti angka dua puluh?" kata Kelly. "Kapal itu sangat besar. Lebih besar dari apa pun yang dapat saya pikirkan." Mike menjawab, "Baiklah, dua puluh memang." Cassie menyeringai dan berkata, "Jika kapal induk adalah dua puluh, apa itu rumah perahu?" Mike tertawa dan berkata, "Bukan dua puluh." Dia mengusap dagunya, berpikir sejenak, dan menjawab, "Saya kira rumah perahu adalah tiga." "Itu dia," kata Cassie.

Poin cerita adalah apa yang Anda harapkan dari usaha Anda untuk menyelesaikan sebuah cerita dalam kaitannya dengan cerita-cerita lainnya. Nah, usaha bukanlah kata yang tepat untuk digunakan. Anda harus mengukur seberapa banyak usaha yang terlibat bersama dengan kompleksitas cerita dan keraguan yang mungkin Anda miliki." Kevin menimpali: "Sekarang saya paham mengapa poin tidak selalu sama dengan waktu." "Bagaimana bisa?" tanya Cassie. "Karena setiap orang dalam tim memiliki keterampilan dan pengalaman yang berbeda. Apa yang mungkin memerlukan waktu sehari bagi saya mungkin memerlukan waktu lebih lama bagi orang lain, dan sebaliknya." "Tepat sekali," kata Cassie. "Perhatikan bahwa saya tidak menjelaskan secara spesifik jenis kapal induk atau rumah perahu. Masing-masing memiliki ukuran yang berbeda. Estimasi adalah perbandingan relatif tentang seberapa banyak pekerjaan yang dibutuhkan untuk menyelesaikan sebuah cerita.

Estimasi tidak ada hubungannya dengan berapa lama waktu yang dibutuhkan seorang anggota tim untuk menyelesaikan cerita tersebut. Tidak peduli siapa yang mengerjakan cerita tersebut, poin ceritanya sama saja." "Cassie, saya rasa tim ini tidak dapat membangun kapal induk dalam satu Sprint," kata Kevin. "Itulah mengapa saya memilih kapal induk." Cassie sangat bersemangat. Dia harus menahan diri untuk tidak melompat-lompat. "Kapal induk berukuran dua puluh inci jelas terlalu besar untuk dimasukkan dalam satu Sprint. Cerita itu perlu dipecah atau disempurnakan lebih lanjut." Estimasi bukanlah ilmu pasti. Saya merasa sulit meyakinkan sekelompok orang dengan gelar teknik bahwa sesuatu yang tidak begitu tepat seperti story point bisa jadi berharga.

Memang begitu. Jika dilakukan dengan benar, story point memberikan gambaran yang sangat akurat tentang pekerjaan yang tersisa untuk diselesaikan dan berapa lama waktu yang dibutuhkan tim untuk menyelesaikannya. Tim hanya perlu jujur dengan diri mereka sendiri tentang volume pekerjaan yang harus diselesaikan. Ini bukan hanya tentang menulis kode. Ini tentang semua yang dibutuhkan untuk menyampaikan apa yang dibutuhkan cerita (pengembangan, dokumentasi, QA, dan sebagainya). Jangan hanya mengikuti pimpinan teknis atau arsitek. Ingat Agile Manifesto di sini. Semakin banyak tim mendiskusikan cerita, semakin tim akan memahami cerita tersebut. Guru Agile selalu menyarankan deret Fibonacci digunakan saat mengukur cerita. Karena empat dari enam angka pertama adalah bilangan prima, maka mustahil untuk memecah sesuatu menjadi bagian-bagian yang sama dan mengerjakannya secara paralel. Sepatah kata tentang membagi cerita.

Jika cerita terlalu besar, Anda mungkin tergoda untuk membagi cerita berdasarkan tugas. Misalnya, bagi cerita tersebut sehingga pekerjaan pengembangan masuk ke Sprint pertama dan pengujian serta dokumentasi terjadi di Sprint kedua. Hal ini tidak dapat dilakukan dalam Scrum, karena Anda tidak memiliki apa pun untuk didemonstrasikan di akhir Sprint pertama. Ingat, kita ingin membangun nilai dalam peningkatan yang dapat didemonstrasikan.

Ketika sebuah cerita dibagi, setengah dari pengembangan, pengujian, dokumentasi, dan apa pun yang diperlukan perlu dilakukan di Sprint pertama. Tim sekarang memiliki fungsionalitas yang teruji dan dapat didemonstrasikan untuk ditunjukkan kepada pelanggan dan pemangku kepentingan. Dan jangan pernah lupa untuk melihat cerita dari sudut pandang pelanggan.

Velocity

"Apa yang sedang kamu lakukan, Cassie?" Sue datang beberapa menit lebih awal untuk stand-up harian. Ketika dia masuk ke ruangan, dia mendapati Cassie dengan setumpuk kertas di depannya. "Oh, hai Sue." Cassie bernyanyi, "Aku sedang menghitung kecepatan rata-rata tim." Sue mengernyitkan wajahnya seolah sedang berpikir keras. "Velocity… Aku pernah mendengar kata itu digunakan dalam rapat sebelumnya. Apa artinya?" "Maksudmu aku tidak pernah menjelaskan kecepatan kepada tim?" Cassie sedikit kesal dengan dirinya sendiri karena lupa menjelaskan sesuatu yang sangat penting. "Oke, apakah kamu siap mempelajari rahasia besar kecepatan Agile?" "Tentu," Sue terkekeh. "Kecepatan adalah seberapa cepat tim memberikan nilai kepada para pemangku kepentingannya." "Jadi seberapa cepat kita memberikan kode yang teruji?" Cassie menurunkan kacamatanya. "Tidak, seberapa cepat kamu memberikan nilai.

Apakah kamu pernah bertanya-tanya mengapa kita tidak memperkirakan cacat?" "Maksud Anda adalah hal-hal yang kami temukan salah saat kami mengujinya?" tanya Sue. "Ya, itu adalah cacat. Kami memberi mereka prioritas yang sangat tinggi karena kami ingin memberikan pelanggan kami barang yang bebas cacat, tetapi kami tidak memperkirakan seberapa besar pekerjaan yang diperlukan untuk memperbaikinya. Pernahkah Anda bertanya-tanya mengapa?" "Sekarang setelah Anda menyebutkannya, kami tidak memperkirakan cacat. Mengapa demikian?" tanya Sue. "Karena pelanggan mengharapkan produk kami bebas cacat. Kami tidak benar-benar memberikan nilai baru saat kami memperbaiki cacat. Kami melakukan sesuatu yang diharapkan pelanggan. Sama seperti saat Anda membeli mobil, Anda mengharapkannya berfungsi dengan sempurna... benar?" kata Cassie. "Tentu saja!" jawab Sue. Cassie tersenyum dan menjawab, "Itulah sebabnya kami tidak menyoroti hal-hal seperti cacat dan utang teknis.

Hal-hal itu tidak benar-benar memberikan nilai kepada pelanggan dan pemangku kepentingan kami, karena mereka mengharapkan hal semacam itu dilakukan. Dengan kata lain, mereka mengharapkan produk kami berfungsi dengan baik dan bebas cacat. Jika kita menghasilkan sesuatu yang penuh bug, itu memperlambat kecepatan kita. Sama halnya dengan utang teknis. Itulah hal-hal yang kita biarkan keluar ke lapangan karena kita mencoba membuat tanggal. Saya menggolongkannya sebagai dosa masa lalu. Kita tahu itu ada di lapangan, tinggal menunggu pelanggan menemukannya." "Dan kita harus membersihkan hal-hal itu. Sebagai penguji, saya benar-benar tidak suka merilis apa pun yang menurut saya tidak benar. Anda berkhotbah kepada orang yang sudah seiman, saudari!" seru Sue. "Jadi, apa yang Anda lakukan dengan semua kertas ini?" "Tim telah melalui tiga Sprint, jadi sekarang saya memiliki cukup informasi untuk menghitung kecepatan rata-rata kami.

Sederhana saja. Hitung berapa banyak poin cerita yang telah diselesaikan tim setiap Sprint dan bagi dengan jumlah Sprint. Ini akan menunjukkan bagaimana kinerja kami sebagai

tim, dan perkirakan berapa lama waktu yang dibutuhkan untuk menyelesaikan poin cerita yang tersisa di Backlog rilis." Sue tersenyum dan berkata, "Jadi, tidak ada lagi perdebatan tentang tanggal?" Cassie menjawab, "Kita masih bisa berdebat, hanya saja tidak tentang kapan kita bisa merilis. Kecepatan rata-rata tim kami menunjukkan seberapa cepat kami memberikan nilai." Bug adalah kenyataan yang tak terelakkan dalam pengembangan perangkat lunak. Seberapa keras pun Anda mencoba, semuanya akan salah dan Anda akan memiliki bug yang harus diperbaiki. Selain itu, cacat harus ditangani dengan cepat dan tegas dalam Agile. Yang berarti nol cacat… Seperti tidak ada bug baru Nol, tidak ada, nol, tidak ada, tidak ada.

Idenya adalah bahwa cacat tidak akan meningkat nilainya seiring waktu, tetapi biayanya akan meningkat secara eksponensial. Standar yang saya lihat adalah bahwa cacat harus ditutup dalam Sprint tempat cacat itu ditemukan. Idenya adalah bahwa kita tidak ingin, dalam keadaan apa pun, membuat banyak bug. Oke, itu bagus dan sebagainya, tetapi apa yang kita lakukan dengan semua hal yang telah ada dalam produk kita selama bertahun-tahun? Seperti hal-hal yang "dinegosiasikan" pada masa Waterfall yang tidak pernah kita bersihkan? Itu, teman-teman, adalah utang teknis, seperti yang saya uraikan dalam Bab 5. Sekali lagi, jauh lebih mahal untuk memperbaiki bug setelah produk dirilis. Anda perlu membongkar kembali modul-modul, mempelajari kembali cara kerja kode, dan jika pelanggan menemukannya, berusaha sekuat tenaga untuk memperbaikinya dengan cepat dan membuat mereka senang.

Setiap produk memiliki utang teknis dalam jumlah tertentu. Mulai dari dependensi yang sulit dipahami dalam modul yang stabil dan jarang dimodifikasi di bagian tersembunyi produk Anda hingga kode yang ceroboh yang sering diabaikan dalam menangani masalah dengan prioritas lebih tinggi dan tidak pernah ditinjau ulang utang teknis itu ada di sana... mengintai. Sama seperti cacat, utang teknis perlu diidentifikasi dan ditangani. Anda tentu tidak dapat memberikan prioritas setinggi cacat, tetapi Pemilik Produk harus mendorong tim Scrum mereka untuk menanggung sedikit utang teknis setiap Sprint. Angka yang saya dengar adalah 30% dari kapasitas tim. Saya akan lalai jika tidak mengatakan bahwa hal-hal seperti menulis pengujian otomatis juga merupakan utang teknis. Menurut saya, utang teknis adalah segala sesuatu yang tidak memberikan nilai kepada pelanggan.

Dapat dikatakan bahwa ini adalah utang yang baik. Saya setuju menulis otomatisasi pengujian memungkinkan nilai masa depan dengan memungkinkan pengujian regresi yang lebih cepat. Saat ini, hal itu masih menunda nilai pelanggan. Secara teknis (maaf atas permainan kata-katanya), cacat juga merupakan utang kami hanya mencoba untuk "membayarnya" segera. Inilah sebabnya mengapa baik cacat maupun utang teknis tidak boleh ditetapkan sebagai poin cerita. Keduanya seharusnya berdampak negatif pada kecepatan dan memberikan gambaran akurat tentang seberapa cepat tim menyelesaikan fitur. Saya suka bagaimana Bob Carpenter mendefinisikan kecepatan. Kecepatan adalah seberapa cepat tim Scrum memberikan nilai kepada pelanggan. Pelanggan sudah mengharapkan produk yang kami jual kepada mereka untuk diuji, dan bebas bug. Nilainya ada pada fitur yang kami berikan. Kecepatan adalah jumlah poin cerita yang diselesaikan tim Scrum Anda selama Sprint.

Sprint demi Sprint, Anda mengembangkan kecepatan rata-rata yang memungkinkan Anda mengukur dengan lebih baik kapan Anda akan menyelesaikan proyek. Bagaimana

sesuatu yang saya sebut "perkiraan" dapat digunakan untuk melacak proyek? Dengan kata lain, bagaimana sesuatu yang diukur menggunakan metodologi yang tidak tepat dapat digunakan untuk memberi Anda tanggal rilis yang tepat, atau bahkan tanggal rilis yang bahkan berada di kisaran yang sama? Sederhana. Selama Anda menetapkan poin cerita secara seragam, poin cerita akan memberi Anda gambaran yang lebih tepat tentang berapa lama sesuatu akan berlangsung daripada menggunakan jam dan/atau hari. Poin cerita tidak memberikan rasa presisi seperti halnya jam atau hari kerja. Jika estimasi poin cerita Anda konsisten, kecepatan dengan cepat menunjukkan tidak hanya seberapa cepat tim bergerak, tetapi juga bagaimana hal-hal yang bersifat eksternal bagi tim memengaruhi pekerjaan yang dapat diselesaikan tim dalam Sprint.

Yang penting bukanlah berapa jam yang Anda habiskan untuk bekerja dalam Sprint melainkan berapa banyak poin cerita yang Anda selesaikan per Sprint dan berapa banyak yang tersisa untuk dilakukan. Inilah sebabnya Anda ingin menyelesaikan cerita Anda dalam Sprint yang sama saat Anda memulainya. Ada banyak alasan untuk ini. Anda ingin membuat cerita Anda cukup kecil untuk diselesaikan dalam Sprint sehingga Anda mendapatkan umpan balik dan memberikan nilai kepada pelanggan secara bertahap. Namun, alasan terbesar menurut saya adalah bahwa poin cerita untuk cerita tertentu dikenali dalam Sprint saat cerita ditutup. Jika Anda secara konsisten membiarkan cerita melewati Sprint, Anda memengaruhi kecepatan tim Anda secara negatif.

Hambatan

“Hai Richard,” Cassie bernyanyi saat ia berjalan memasuki kantor manajer fungsional. “Untuk apa saya berutang hak istimewa ini?” tanya Richard. “Kunjungan dari orang paling positif yang saya kenal.” “Oh, hentikan,” Cassie berpura-pura protes, tetapi senang karena seseorang memperhatikan kepositifannya yang konstan. “Tim mengemukakan beberapa hal pada rapat stand-up hari ini, dan saya ingin bantuan Anda untuk menyelesaikannya.” “Oke,” kata Richard. “Apa yang bisa saya bantu?” Cassie duduk dan melipat tangannya. “Masalah pertama adalah kita memerlukan kontrol sumber untuk rencana pengujian otomatis kita. Saya berbicara dengan Sue dan Ray tentang hal itu setelah rapat…” “Salah satu rapat tempat parkir Anda?” tanya Richard. “Ya, tentu saja.” Cassie tersenyum lebar saat ini, “Mereka mengatakan bahwa mereka akan memerlukan server dan perangkat lunak kontrol versi. Mereka memang menyebutkan satu yang mereka sukai, tetapi saya tidak dapat mengingatnya, dan saya meninggalkan catatan saya di meja saya.” “Jangan khawatir, Cassie. Ini adalah sesuatu yang ada di bidang saya.

Saya akan mengumpulkan para penguji dan mengurus apa yang mereka butuhkan. Mereka mungkin perlu mengalokasikan anggaran, jadi saya orang yang tepat untuk pekerjaan itu!” Dia duduk bersandar di kursinya dan meletakkan tangannya di belakang kepala. “Apa masalah lainnya?” “Seorang tenaga penjualan telah mencoba menghubungi tim secara langsung. Saya cukup yakin dia mencoba untuk menutup kesepakatan atau melakukan sesuatu dengan bukti konsep,” kata Cassie. Mata Richard menyipit, “Menurutmu apakah dia harus berbicara dengan Pemilik Produk?” “Benar,” kata Cassie. “Kita perlu meminimalkan gangguan. Tim perlu fokus pada sasaran Sprint. Peluang penjualan ini mungkin besar, tetapi harus

diperiksa dengan Pemilik Produk dan diprioritaskan." Richard mengusap dagunya dan berkata, "Saya ingin memastikan bahwa saya sangat jelas tentang ini.

Mengapa Pemilik Produk perlu dilibatkan?" "Karena Pemilik Produk bertanggung jawab atas proyek tersebut," tegas Cassie. "PO adalah satu-satunya orang yang bertanggung jawab atas apa yang terjadi. Jika penjualan ini benar-benar cukup penting untuk mendapatkan perhatian tim, PO perlu menentukan apa yang harus dikeluarkan dari Sprint dan dikembalikan ke Backlog. Ia harus mampu mengartikulasikan kriteria pekerjaan dan penerimaan dengan jelas kepada tim sehingga mereka dapat fokus pada peluang penjualan." "Begitu ya," kata Richard. "Tidak ada yang gratis." "Benar, Richard. Anda tidak dapat membuat tim bekerja lembur dan di akhir pekan kecuali mereka setuju untuk melakukannya."

Salah satu tugas utama Scrum Master adalah menyingkirkan hambatan. Hambatan didefinisikan sebagai apa pun yang dapat memperlambat tim atau mencegah mereka melakukan pekerjaan mereka. Berikut adalah alasan lain mengapa rapat stand-up harian sangat penting. Salah satu dari tiga pertanyaan yang perlu dijawab adalah, "Apa yang menghalangi Anda?" Scrum Master perlu mengingat bahwa meskipun merupakan tanggung jawab mereka untuk menyingkirkan hambatan, mereka tidak harus melakukannya sendiri. Manajer fungsional harus bekerja sama dalam topik ini. Ingat, Scrum Master tidak memiliki wewenang apa pun. Anda tahu, hal yang sama berlaku pada servant-leader. Manajer fungsional memiliki wewenang, dan dapat membantu menyelesaikan berbagai hal. Kendala bisa berupa laptop anggota tim yang perlu diperbaiki.

Dalam kasus ini, Scrum Master akan menyediakan sistem pinjaman dan memastikan bahwa anggota tim memiliki semua yang diperlukan di laptop pinjaman tersebut. Mereka juga akan memastikan bahwa perangkat keras yang rusak telah diperbaiki. Dengan cara ini, alih-alih anggota tim berurusan dengan meja bantuan, mereka mengerjakan sasaran Sprint. Tim juga membutuhkan perlindungan dari gangguan luar. Seperti yang dikatakan Cassie dalam cerita kami semuanya harus melalui Pemilik Produk. Apa pun yang diminta dari tim harus diprioritaskan dan disesuaikan. Jika tim diharapkan untuk meninggalkan semuanya, setiap orang perlu memahami bahwa ada akibat dari tindakan ini. Anda tidak bisa begitu saja menumpuk sesuatu dan mengharapkan tim untuk mengatasinya. Tim Scrum harus dapat fokus untuk mencapai sasaran Sprint setiap iterasi.

6.6 PELAKSANAAN SPRINT

Cassie menyiapkan rapat harian seperti biasa. Papan Scrum tampak menonjol di tengah ruangan. Papan tersebut dengan jelas menunjukkan status dan kemajuan setiap cerita dalam Sprint Backlog. Ia memperbarui bagan burn-down Sprint, dan menunggu tim Scrum tiba. Semakin ia melihat papan Scrum, semakin ia tidak menyukai apa yang dilihatnya. Tim tersebut berbaris, menunjukkan berbagai macam emosi. Dari kegembiraan hingga kekesalan, tim berkumpul di sekitar papan Scrum dan menatap Cassie. "Haruskah kita mulai?" tanya Sue. Salah satu hal yang Cassie jelaskan kepada tim sejak awal adalah bahwa ini adalah rapat mereka. Ia tidak menjalankannya, jadi ia tidak tertarik untuk mengajukan pertanyaan apa pun. Bahkan, setelah beberapa rapat stand-up, Cassie berdiri dan tidak mengatakan apa pun di awal

rapat. Mereka semua berdiri dalam keheningan yang tidak nyaman sampai Steve akhirnya mulai berbicara tentang apa yang telah ia lakukan kemarin, apa yang akan ia lakukan hari ini, dan hambatan apa pun yang dialaminya. “Sebelum kalian mulai, saya ingin menyampaikan sesuatu.”

Cassie tidak suka mengganggu rapat seperti ini, tetapi dia sangat merasa perlu untuk mengatasi situasi yang ada. “Apakah ada di antara kalian yang melihat potensi masalah dengan papan Scrum kita pagi ini?” Tim tampak bingung. Mike angkat bicara, “Cassie, menurutku ini terlihat bagus. Kami menyelesaikan berbagai macam pekerjaan pada cerita-cerita ini.” “Mungkin,” kata Cassie. “Tetapi inilah kekhawatiran saya. Setiap cerita dalam Sprint Backlog saat ini sedang dalam proses.” “Bagaimana itu bisa menjadi masalah?” tanya Jack. “Tujuan dari Sprint adalah menyelesaikan cerita.” Cassie berjalan mendekat dan menyentuh papan Scrum untuk menarik perhatiannya. “Melihat Sprint burn-down dan papan Scrum, apakah menurutmu kita akan menyelesaikan semua cerita ini pada akhir Sprint?” “Tentu saja,” kata Jack dengan seringai di wajahnya. Ganesh sedikit lebih skeptis: “Saya tidak tahu tentang itu, Jack. Masih banyak pekerjaan yang tersisa di sana, dan kita sudah setengah jalan melalui Sprint.” “Kita akan menyelesaikannya, Ganesh.

Apa pun yang terjadi. Kita akan memberikan apa yang kita janjikan.” Cassie melihat peluangnya dan memanfaatkannya. “Jack, itu mungkin berhasil dalam jangka pendek, tetapi tidak berkelanjutan dalam jangka panjang.” Jack memotong Cassie dan berkata, “Saya tahu, manifesto mengatakan tim harus dapat bekerja dengan kecepatan yang dapat mereka pertahankan tanpa batas waktu, atau semacamnya.” “Itu benar, Jack, dan cara tim Scrum dapat mencapai kecepatan itu adalah dengan membatasi WIP.” Cassie tidak yakin, seberapa terbuka tim terhadap konsep ini, tetapi merasa sudah waktunya untuk menantang mereka dengan konsep ini. “Apa itu WIP?” tanya Kelly. “Apa yang akan Cassie lakukan kepada kita agar kita bekerja lebih cepat,” canda Steve. Jack tidak mengatakan apa pun. Dia hanya menatap Cassie dan menunggu penjelasan. “WIP adalah singkatan dari work in progress.

Izinkan saya mengajukan pertanyaan sederhana.” Cassie menoleh ke Mike, yang hingga saat ini diam saja. “Apa yang lebih baik di akhir Sprint, menyelesaikan seratus persen cerita hingga lima puluh persen atau menyelesaikan lima puluh persen cerita hingga seratus persen?” Mike menggigit bibirnya dan memikirkannya. “Kurasa kita ingin menyelesaikan lima puluh persen cerita.” “Woop woop! Mike menang!” seru Cassie. “Kita ingin, sebagai tim, menyelesaikan cerita dengan prioritas tertinggi. Misalnya, alih-alih satu orang mengerjakan satu cerita, kalian hanya mengerjakan dua cerita dengan prioritas tertinggi hingga selesai.” “Aku tidak melihat nilainya dalam hal ini,” kata Jack. “Kedengarannya seperti…” Cassie menimpali. “Giliranku untuk memotong pembicaraanmu.

Jack Bagaimana jika kamu berada di tengah danau di atas perahu, dan tiba-tiba, ada lubang di perahu dan air masuk dengan deras. Apa yang akan kamu lakukan?” “Wah, aku akan mulai membuang air secepat yang kubisa…” Cassie tersenyum dan bertanya, “Bukankah sebaiknya kau tutup lubang di perahu itu terlebih dahulu?” Jack menyeringai malu dan berkata, “Kurasa begitu.” “Ketika tim tidak membatasi WIP mereka, sangat mudah untuk melupakan prioritas. Yang kau lihat hanyalah segunung pekerjaan. Namun, jika kau

mengerumuni cerita yang paling penting terlebih dahulu, cerita itu akan selesai. Segunung pekerjaan akan mengecil, dan tim akan beralih ke prioritas tertinggi berikutnya." "Tetapi mengapa itu penting?" tanya Mike. "Kami mungkin akan menyelesaikan semuanya." Cassie tersenyum dan berkata, "Mungkin. Namun, jika Anda membatasi WIP sebagai sebuah tim, setidaknya akan ada peningkatan kecepatan sepuluh persen.

Saya pikir itu ada hubungannya dengan efek psikologis saat melihat sesuatu selesai. Itu menciptakan semacam kegilaan makan atau efek longsor." "Jadi, saya kira Anda ingin kami membatasi jumlah cerita yang sedang dikerjakan secara agresif," kata Mike dengan sedikit sarkasme. "Begini," kata Cassie, dengan cukup berani untuk melawan usaha Mike yang buruk untuk menjadi jenaka. "Mengapa kalian tidak mencobanya saja? Kumpulkan beberapa cerita dan lihat bagaimana hasilnya. Jika kita sepakat bahwa pembatasan WIP berhasil, kita dapat memperluas praktiknya." "Maksud Anda, Anda ingin kami melakukan semacam eksperimen?" tanya Sue. "Itulah yang saya maksud," jawab Cassie. Mengelola pekerjaan yang sedang dikerjakan (WIP) adalah tanda kedewasaan. Untuk beberapa alasan, kebanyakan orang ingin mulai mengerjakan semuanya sekaligus. Saya menyebutnya kutukan mengerjakan banyak tugas sekaligus.

Ya, kita hidup di masa yang tampaknya menuntut kita untuk mengerjakan banyak tugas sekaligus, tetapi menurut saya, menyelesaikan sesuatu itu lebih penting. Menyelesaikan sesuatu mengharuskan kita untuk fokus. Selalu ada lebih banyak pekerjaan yang harus dilakukan daripada yang dapat kita selesaikan. Tanpa fokus, kita akan kewalahan dan hal-hal penting seperti kualitas akan menurun. Salah satu pepatah favorit saya adalah "Lebih baik menyelesaikan lima puluh persen cerita Anda seratus persen daripada menyelesaikan seratus persen cerita Anda lima puluh persen." Membatasi WIP memaksa tim keluar dari pola pikir Waterfall. Kami tidak menghargai kesibukan dalam Scrum; kami menghargai penyelesaian cerita dan penyampaian nilai. Membatasi WIP memungkinkan tim Scrum untuk fokus pada penyampaian cerita yang lengkap dan teruji. Mungkin tampak berlawanan dengan intuisi untuk membiarkan beberapa cerita terbengkalai sementara tim fokus pada penyelesaian yang lebih berprioritas. Namun, jika dipikir-pikir, tim tersebut menyelesaikan hal-hal yang berprioritas tinggi.

Rahasia besarnya adalah bahwa cerita-cerita berprioritas rendah itu akan tetap diselesaikan, karena berfokus pada ukuran batch yang lebih kecil memungkinkan tim menyelesaikan sesuatu dengan lebih cepat. Ada kekuatan dalam melakukan apa yang Anda katakan akan Anda lakukan. Batasan WIP menjamin bahwa tim akan menyampaikan cerita berprioritas tinggi setiap Sprint. Itulah sebabnya saya sangat menganjurkan tim melakukan eksperimen. Dalam kasus WIP, saya akan melatih tim untuk mencoba mengumpulkan beberapa cerita untuk melihat apakah mereka dapat menyelesaikannya lebih cepat. Jika tim menyadari manfaat dari eksperimen tersebut, mereka akan terbuka untuk mengadopsi praktik tersebut. Dalam kasus ini, tetapkan batasan WIP dalam arti yang lebih luas. Menurut pengalaman saya, sebagian besar tim menolak ketika Anda mencoba mengubah sesuatu. Perubahan itu menakutkan dan tidak diketahui. Meskipun demikian, tim biasanya terbuka untuk melakukan eksperimen selama beberapa minggu. Jika semuanya berhasil, tim akan

membuang ide tersebut. Jika nilainya terwujud, tim dapat mengadopsi praktik tersebut.

Menyerahkan Pekerjaan Tambahan Ke Dalam Iterasi

"Cassie, bolehkah kami bertanya?" kata Mike. Ia berdiri di dekat bilik Cassie dengan Kelly di belakangnya. "Tentu, apa yang ada dalam pikiranmu?" kata Cassie. "Kelly dan aku sedang berselisih pendapat, dan kami tidak yakin siapa yang benar," kata Mike. "Dan kami ingin kau memberi tahu kami." Kata Kelly. "Oke, lanjutkan," kata Cassie dengan senyum di wajahnya. "Pada dasarnya, aku sudah menyelesaikan semua pekerjaan Sprint-ku," kata Mike. "Bolehkah aku mengambil cerita baru dari Backlog? Aku ingin memulai pekerjaan untuk Sprint berikutnya lebih awal," kata Mike. "Kurasa itu bukan yang seharusnya kita lakukan." kata Kelley. "Bukankah itu melanggar aturan?" "Tidak ada aturan," Cassie terkekeh. Scrum adalah sebuah kerangka kerja. Idenya adalah kalian, tim, mencari tahu cara bekerja dalam pedoman kerangka kerja." "Jika tidak ada aturan, mengapa kita harus menghadiri semua rapat ini?" tanya Kelly. "Anda harus bekerja dalam kerangka kerja. Itu berarti Anda harus mengulanginya setiap dua hingga empat minggu dan menyertakan semua upacara Scrum."

Cassie berhenti sejenak dan menggigit bibirnya. "Baiklah, mungkin hal-hal itu bisa dianggap sebagai aturan, tetapi kami mencoba membuatnya seminimal mungkin. Saya tidak ingin melihat siapa pun mendikte tim tentang cara melakukan pekerjaan. Tetapkan saja apa yang dibutuhkan untuk menjaga semua orang selaras dengan tujuan Sprint dan visi proyek." Cassie merasakan bahwa percakapan itu menyimpang dari pertanyaan awal, jadi dia menatap Mike dan bertanya, "Jadi, Anda sudah menyelesaikan semua pekerjaan yang Anda janjikan untuk Sprint ini?" "Ya," kata Mike. "Apakah anggota tim lainnya juga sudah selesai?" tanya Cassie. "Yah… tidak," jawab Mike. "Apakah ada yang bisa Anda lakukan untuk membantu menyelesaikan pekerjaan Sprint yang tersisa?" Cassie merasa dia hampir bisa melihat roda berputar di kepala Mike. "Aku tidak tahu," Mike tergagap. "Tenang saja, Mike," Cassie terkekeh. "Aku tidak mencoba membuatmu terpojok.

Sebagai anggota tim, kamu harus selalu mencari cara untuk membantu tim mencapai tujuan Sprint. Itu mungkin berarti kamu turun tangan dan melakukan beberapa pengujian atau membantu menulis beberapa kode untuk cerita yang belum selesai." "Jadi, daripada mengerjakan cerita dengan prioritas rendah, aku membantu menyelesaikan cerita dengan prioritas tinggi yang sudah ada di Sprint," kata Mike. Cassie tersenyum dan berkata, "Saya sendiri tidak bisa mengatakannya dengan lebih baik." "Jadi Anda tidak akan pernah bisa menarik cerita ke Sprint setelah perencanaan sprint selesai?" tanya Kelly. "Saya lebih suka tidak pernah mengatakan tidak pernah." Cassie menjawab. "Selalu lakukan apa yang masuk akal. Mungkin ada situasi di mana hal terbaik yang dapat dilakukan adalah menarik cerita ke Sprint saat ini, tetapi menurut saya itu bukan ide yang bagus saat ini." Di permukaan, itu tampak seperti ide yang bagus. Anda telah menyelesaikan pekerjaan Sprint Anda dan ingin menarik cerita dari iterasi berikutnya ke iterasi ini.

Anda tidak akan dapat menyelesaikan cerita tersebut, tetapi poin cerita dihitung dalam Sprint tempat cerita tersebut diterima. Ini seperti memulai lebih awal pada Sprint berikutnya. Ide yang bagus, bukan? Biasanya, tidak. Apakah ada yang dapat dilakukan untuk membantu tim mencapai tujuan Sprint? Dengan kata lain, dapatkah Anda membantu mengumpulkan

cerita dengan prioritas tertinggi yang tersisa? Dapatkah Anda mengatasi beberapa utang teknis, atau mungkin melakukan refaktor beberapa kode? Suatu cerita dapat dimasukkan ke dalam iterasi saat ini jika PO dan tim setuju bahwa itu merupakan ide yang bagus.

PO mungkin dapat memasukkan sesuatu yang kecil yang dapat diselesaikan dalam sisa waktu Sprint, atau bahkan menyiapkan cerita lonjakan untuk melakukan sedikit riset tentang sesuatu yang akan datang. Tak perlu dikatakan lagi bahwa cerita yang dipertimbangkan perlu disempurnakan dan diberi tugas. Menarik pekerjaan tambahan ke dalam suatu iterasi harus berada di bagian bawah daftar hal-hal yang perlu dipertimbangkan jika seorang anggota tim menyelesaikan pekerjaan Sprint-nya lebih awal.

Stand-Up Harian Atau Rapat Scrum

Tim berdiri di sekitar papan Scrum, dan tampak lesu. Cassie bertanya, “Apa yang terjadi hari ini, tim? Kalian tampak seperti sedang menghadiri pemakaman.” Kelly berkata, “Saya akan langsung mengatakannya… Saya tidak melihat manfaat mengadakan rapat ini setiap hari.” Cassie memutar matanya dan berkata, “Kalian sadar bahwa tidak ada yang mengawasi kalian lagi. Sungguh, saya serius di sini… Kalian harus bertanggung jawab. Manajer kalian bukan lagi orang yang menyelesaikan semua masalah kalian dan memberi tahu apa yang harus dikerjakan.” “Maksudmu kami bertanggung jawab atas segalanya?” tanya Jack. “Richard adalah manajer fungsional di sini. Tugasnya adalah menyediakan lingkungan kerja bagi tim. Dia mempertahankan apa yang saya sebut sebagai pabrik.” "Itukah sebabnya saya mendengar Anda berbicara dengannya tentang sesuatu yang disebut masalah pabrik tempo hari?" tanya Kevin. "Ya, Kevin. Masalah pabrik adalah hal-hal yang memengaruhi kemampuan tim untuk bekerjaseperti meja atau pelatihan.

Saya berbicara dengannya tentang memastikan bahwa tim memiliki cukup lisensi produk sehingga setiap orang dalam tim dapat mengompilasi kode." Ini sebenarnya bukan yang ingin dibicarakannya, tetapi Cassie dengan senang hati menjelaskan peran manajer fungsional dan definisi masalah pabrik. Dia melanjutkan, "Tim ini perlu menyadari satu fakta. Anda yang bertanggung jawab. Ambil kepemilikan atas pekerjaan dan kelola diri Anda sendiri." Ganesh menyela, "Tunggu sebentar, Cassie. Apakah Anda benar-benar bermaksud tidak ada yang akan memberi tahu kita apa yang harus dilakukan?" Cassie menarik napas dalam-dalam, berhenti sejenak untuk mengumpulkan pikirannya, dan menjawab. "Struktur Waterfall lama melibatkan peran manajerial yang mengawasi tim pengembangan dan membuat keputusan tentang cara kerja tim. Manajer menugaskan pekerjaan dan mengkhawatirkan cara produk dibangun. Sekarang, anggota tim Scrum berkomitmen, atau mendaftar untuk bekerja. Tim bertanggung jawab untuk menyampaikan apa yang mereka janjikan untuk disampaikan. Tim memutuskan bagaimana menyampaikan fungsionalitas.

Tim Scrum harus mengarahkan dan mengatur diri sendiri. Tim membuat pilihannya sendiri dan memiliki hasil dari pilihan ini.” “Jadi, terserah kita,” kata Jack. Cassie tersenyum. “Ya. PO akan memberi tahu Anda apa yang dia inginkan, tetapi terserah Anda TIM untuk memutuskan bagaimana mengubah ide PO menjadi produk yang berfungsi. Setiap keputusan adalah milik tim. Dari persyaratan pelatihan hingga bahasa pengodean apa yang akan digunakan. Tim Scrum memiliki apa yang dikerjakannya.” “Itu adalah tanggung jawab yang

sangat besar," kata Kelly. "Bagaimana kita membuatnya berhasil?" Cassie menjawab, "Dengan berbicara satu sama lain. Itulah sebabnya kami mengadakan rapat stand-up setiap hari. Ini bukan dimaksudkan sebagai rapat di mana Anda datang dan berbicara dengan saya atau Pemilik Produk atau manajer Anda. Anda seharusnya berbicara satu sama lain.

Melakukan percakapan, bahkan yang sulit. Mencapai resolusi. Kalian bukan sekelompok orang yang disatukan…" "Kita adalah sebuah tim," kata Kevin. 'Ya, kalian adalah sebuah tim. Saya rasa kalian belum menyadari fakta itu. Beberapa dari kalian perlu menemukan suara kalian dan berbicara. Apa pun yang kalian katakan, terlibat akan menambah perspektif dan informasi dalam percakapan. Tidak adil mengharapkan Kevin atau Jack untuk membahas semuanya. Ya, mereka adalah anggota tim yang paling berpengalaman; namun, mereka bukan satu-satunya dua orang dalam tim. "Cassie bersandar ke dinding dan berkata, "Itulah mengapa rapat harian, atau rapat Scrum harian sangat penting. Kalian perlu berkumpul sebagai sebuah tim, membicarakan apa yang telah dicapai sejauh ini, dan membuat komitmen satu sama lain." "Apa artinya itu?" tanya Sue. "Artinya, Anda tidak hanya mengatakan apa yang akan Anda lakukan hari ini. Anda berkomitmen kepada tim bahwa inilah yang akan Anda lakukan hari ini. Anda mungkin berhasil atau tidak dalam apa yang Anda katakan, tetapi tim sekarang tahu apa yang sedang terjadi." "Jadi, kami dapat merencanakan pekerjaan kami dengan baik," kata Jack. "Dan menawarkan bantuan jika kami bisa," imbuh Ganesh. Mereka mulai mengerti, pikir Cassie dalam hati.

Ia berkata, "Itulah sebabnya rapat tatap muka tidak boleh berlangsung selama satu jam. Ini adalah kesempatan cepat bagi tim untuk saling bertukar pikiran, memahami di mana posisi kami, dan apa yang direncanakan untuk hari ini. Seperti yang saya katakan, tidak seorang pun akan memberi tahu Anda apa yang harus dilakukan. Kalian harus mencari tahu sendiri." Satu hal yang selalu dilakukan oleh setiap tim yang pernah bekerja dengan saya adalah mengeluh tentang rapat. Khususnya, rapat tatap muka harian. Di masa lalu, saya pernah membiarkan tim tidak mengadakan rapat tatap muka setiap hari. Kami mencoba setiap dua hari sekali, dua kali seminggu, bahkan tiga kali seminggu. Setiap kali, tim gagal dengan cara yang sama. Semuanya tidak selesai. Banyak hal yang terlewatkan. Tim tidak berkomunikasi, apalagi berkolaborasi. Jelaslah bahwa kami perlu bertemu setiap hari. Masalahnya adalah tim tidak melihat nilai dari rapat stand-up harian. Saya tidak bisa menyalahkan mereka. Lagi pula, mereka terbiasa dengan rapat status Waterfall.

Anggota tim akan memberikan informasi yang terlalu terperinci. Seperti yang saya katakana "di balik layar." Karena itu, rapat stand-up akan berlangsung lebih dari lima belas menit. Jauh lebih lama. Saya mulai memasang pengatur waktu besar tepat di sebelah papan Scrum. Saya bermaksud memberi tahu tim bahwa saya hanya akan mengizinkan rapat berlangsung selama lima belas menit, tetapi saya tidak pernah perlu mengatakan sepatah kata pun tentang masalah tersebut. Tim mulai mengawasi dirinya sendiri hanya karena saya memasang jam besar yang menghitung mundur di depan ruangan. Rapat stand-up sangat berharga. Jika tim tidak melihat nilai dalam rapat "sentuhan" harian yang singkat, maka Scrum Master perlu mencari tahu alasannya. Jaga agar rapat stand-up tetap dalam batas waktu yang tepat dan fasilitasi rapat sedemikian rupa sehingga diskusi mengalir di antara anggota tim.

Rapat stand-up harian bukanlah rapat status. Rapat status adalah tempat Anda duduk di sekitar meja bersama manajer dan memainkan permainan yang disebut "lindungi pantat Anda" dengan semua orang di tim Anda. Rapat stand-up adalah rapat singkat tempat tim membuat komitmen satu sama lain bukan kepada manajemen.

Ulasan Dan Demo Sprint

Annie menjalankan rapat tinjauan dengan sangat baik. Beberapa pelanggan dan pemangku kepentingan hadir dan tampak terlibat dalam proses tersebut. Saat Annie mendemonstrasikan fungsionalitas yang disediakan tim, para pemangku kepentingan mengajukan pertanyaan dan memberikan umpan balik. Seluruh tim juga hadir dalam rapat tersebut, meskipun beberapa dari mereka tampak tidak senang dengan apa yang sedang terjadi. "Saya tidak menyukainya," bisik Jack. "Apa yang tidak Anda sukai?" tanya Cassie. "Pertama, kita menopang benda ini dengan data palsu. Itu sudah cukup buruk. Sekarang kita harus duduk di sini dan mendengarkan pelanggan mengobrak-abriknya." Jack mencoba berbisik, tetapi mengalami kesulitan. Cassie menjawab, "Jack, kita perlu mendapatkan umpan balik dari para pemangku kepentingan untuk memastikan bahwa kita membangun hal yang benar." "Saya mengerti," kata Jack. "Tetapi sulit untuk mendengarkan ini. Anda tahu betapa kerasnya saya bekerja pada fitur ini."

Cassie menggigit bibirnya dan berkata, "Jack, tidak ada yang mempertanyakan seberapa keras seseorang bekerja pada fitur apa pun. Namun, tim harus menyambut umpan balik yang mengubah apa yang sedang dikerjakan. Produk terbaik di dunia tidak ada nilainya jika tidak melakukan apa yang diinginkan pelanggan kami.” Pemilik Produk panik tentang demo Sprint… Karena pelanggan menghadiri rapat khusus ini, ada anggapan bahwa semuanya harus apik dan profesional. Meskipun saya menghargai itu, saya tidak merasa perlu slide deck dan demo yang bagus. Saya pernah mendengar pepatah bahwa jika Anda suka sosis, Anda tidak boleh menonton proses pembuatan sosis. Saya tidak setuju; saya ingin tahu apa yang masuk ke dalam makanan yang saya makan. Sama halnya dengan demo Sprint. Kita harus mencari umpan balik dari pelanggan dan pemangku kepentingan kita. Untuk mendapatkan umpan balik yang jujur, tim Scrum perlu secara jujur menyajikan apa yang dikerjakan selama Sprint. Terkadang itu mengharuskan proses pembuatan sosis terlihat.

Daripada membuang-buang waktu membuat slide deck yang besar, tunjukkan apa yang telah dibuat. Saya tidak mengatakan untuk menyingkirkan slide sepenuhnya. Yang saya katakan adalah bahwa manifesto memberi nilai maksimum pada perangkat lunak yang berfungsi. Itulah yang harus menjadi fokus Pemilik Produk selama rapat tinjauan Sprint tidak peduli seberapa "kasarnya" rapat tersebut. Setelah pelanggan dan pemangku kepentingan memberikan umpan balik, tim perlu melakukan sesuatu terhadap umpan balik tersebut. Tim harus cukup fleksibel untuk mengubah apa yang mereka lakukan secara radikal dan bahkan bersedia membuang pekerjaan jika perlu.

Retrospektif Sprint

Cassie baru saja membawa tim mengikuti rapat Sprint Retrospective. Di akhir batas waktu tiga puluh menit, semua orang sepakat mengenai tiga hal untuk dikerjakan. Idenya adalah mereka akan mengambil tindakan ini dan mengerjakannya selama Sprint berikutnya.

“Rapat yang hebat, tim,” kata Cassie. “Mari kita ambil ketiga hal ini dan masukkan ke dalam Backlog.” Saya berharap dapat memutar suara jarum yang ditarik melintasi piringan hitam karena akan muat di sini. Raut wajah tim mencerminkan keterkejutan dan kebingungan. “Backlog?” tanya Kevin. “Saya pikir Anda ingin kami menjaga Backlog, seperti yang Anda katakan, tetap rapi dan teratur.” “Saya tidak ingin banyak hal yang tidak ingin kita lakukan di Backlog, Kevin,” kata Cassie. “Namun, saya ingin semua hal yang kita rencanakan untuk dilakukan di sana. Menempatkan tindakan retrospektif tim ke dalam Sprint Backlog memastikan bahwa kita akan mengatasinya sebelum rapat Retrospective berikutnya.” Teman saya sekaligus sesama pelatih Agile, Jay Cohen, suka berkata, "Jika tidak ada dalam Backlog, maka tidak akan selesai."

Itu konsep yang sederhana. Mencantumkan item tindakan ke dalam Sprint Backlog memberikan gambaran tentang bagaimana tim berencana untuk memeriksa dan beradaptasi. Dengan cara ini, tidak ada yang terlupakan atau terabaikan. Memeriksa dan beradaptasi adalah sesuatu yang tanpa saya sadari telah saya lakukan sepanjang hidup saya. Sejak pertama kali menyentuh barbel, saya telah berusaha untuk menjadi lebih baik. Jika Anda kebetulan masuk ke pusat kebugaran angkat beban yang sangat berat, Anda akan melihat bahwa ketika seseorang mengangkat beban, semua orang menghentikan apa yang sedang mereka lakukan. Bukan untuk mengawasi pengangkat beban, tetapi untuk memberikan perintah lisan dan melatih. "Angkat kepala!" "Duduklah!" "Bergerak dengan pinggul!" Semua orang berkumpul untuk mendukung orang yang mengangkat beban dengan mencari kekurangan dalam teknik dan titik lemah mereka yang perlu ditangani.

Tubuh manusia bekerja sebagai satu kesatuan. Itu berarti semua jenis kelompok otot perlu bekerja sama untuk, misalnya, melakukan squat berat. Saya telah berkompetisi dalam semua jenis olahraga kekuatan yang gila selama bertahun-tahun di planet Bumi. Jika Anda mengadakan semacam kontes di mana Anda harus menggerakkan sesuatu yang berat saya akan muncul dan biasanya menang… Agar dapat bersaing di level ini, saya harus memahami apa saja kekuatan dan kelemahan saya. Itu semua baik dan bagus, tetapi intinya adalah mencoba mengubah kelemahan tersebut menjadi kekuatan sambil memastikan bahwa Anda menguasai hal-hal yang sudah Anda kuasai. Ini masalahnya: Untuk melakukan itu, Anda harus melakukan hal-hal yang tidak ingin Anda lakukan. Seperti yang saya katakan, "kuasai hal-hal yang tidak Anda kuasai." Misalnya, selama bertahun-tahun, pegangan saya lemah.

Itu benar-benar menghambat deadlift saya. Setiap kali saya mengangkat beban berat, barbel akan terlepas dari tangan saya. Setelah memikirkan sekitar sejuta alasan di kepala saya, saya menghadapi situasi itu dan berfokus pada kelemahan saya. Itu tidak menyenangkan, tetapi sekarang saya bisa menekuk wajan penggorengan dengan tangan itu. Saya tidak ingat kapan terakhir kali saya menjatuhkan deadlift. Pahami apa yang Anda kuasai, dan jangan takut untuk meninggalkan zona nyaman Anda untuk memperbaiki kelemahan Anda. Luangkan waktu untuk memeriksa dan beradaptasi. Saat Anda mengatasi kelemahan tersebut, buatlah kelemahan itu terlihat. Beri tahu semua orang bahwa hal ini sedang ditangani dan catat kemajuan Anda. Seperti yang saya katakan, ubah kelemahan Anda menjadi kekuatan.

Ketrampilan Teknis Master Scrum

"Duane, ini aneh." Cassie sedang mendiskusikan penemuan yang ia buat bersama Duane selama pertemuan mingguan mereka. "Sepertinya semakin sedikit yang saya ketahui tentang pekerjaan yang sedang berlangsung, semakin baik saya menjadi Master Scrum." "Lucu sekali cara kerjanya," jawab Duane. "Saya tahu Anda ahli dalam pengembangan perangkat lunak." "Benar sekali," kata Cassie. "Saya bekerja sebagai pengembang selama lima tahun sebelum saya mengejar posisi Master Scrum." "Tetapi sekarang," jawab Duane, "Anda bekerja dengan tim yang menggunakan bahasa pemrograman yang tidak Anda kenal." Cassie berkata, "Benar sekali. Saya tidak tahu bagaimana mereka menulis hal-hal ini." Duane tersenyum dan berkata, "Jadi Anda dapat fokus pada kelincahan tim. Perhatian utama Anda adalah membuat mereka lebih baik dalam Scrum."

Tim pertama yang saya layani sebagai Master Scrum mengerjakan produk yang sangat saya kenal.Saya telah mengerjakan produk tersebut, mengetahui kodenya, dan sangat ahli dalam cara menginstal, mengonfigurasi, dan menggunakannya. Anda akan berpikir bahwa semua pengetahuan mendalam yang saya miliki tentang produk ini akan memungkinkan saya menjadi Scrum Master yang efektif bagi tim yang mengerjakannya. Bukan itu masalahnya. Malah, kalau dipikir-pikir lagi, saya akan mengatakan bahwa pengetahuan saya tentang produk tersebut menghambat kemampuan saya untuk menjadi pemimpin yang melayani tim saya. Saya hanya "terseret" ke dalam seluk-beluk pengembangan perangkat lunak dan aktivitas tim sehari-hari. Alih-alih berfokus untuk membantu tim menjadi lebih Agile, saya justru berfokus pada pengembangan.

Saya mulai bertindak seperti manajer pengembangan yang ditakuti. Lebih buruk lagi, saya hampir saja terjun dan membantu tim. Itu bukanlah yang seharusnya dilakukan oleh seorang Scrum Master. Fakta ini menjadi sangat jelas bagi saya ketika saya mulai melayani tim yang mengerjakan produk yang tidak saya kenal. Saya merasa lebih mudah untuk berfokus pada kelincahan tim secara keseluruhan ketika tidak ada godaan untuk terlibat dalam pekerjaan teknis. Seorang Scrum Master harus memahami nuansa tentang bagaimana perangkat lunak bersumber dan dibangun, tetapi peran utama mereka adalah menjadi pemimpin yang melayani tim.

Membangun Teknik Ke Dalam Backlog Produk

"Jadi, bagaimana Anda mengharapkan kami membangun sesuatu yang berhasil jika Anda tidak mengizinkan kami merencanakan semuanya di awal?" Kevin agak bingung. Ia kesulitan merancang apa yang sedang dikerjakan tim. Ia mendiskusikannya dengan Cassie, dan Cassie merasa sebaiknya mereka berdua membicarakan hal tersebut dengan Duane. "Kevin, dalam Agile tidak ada desain besar di awal. Arsitektur disempurnakan seiring dengan evolusi produk," kata Duane. Kevin menatap Duane seolah-olah ia memiliki tiga kepala. "Kembali?" Duane berkata, "Mari kita mulai dengan mendefinisikan apa itu arsitektur. Arsitektur dapat berupa desain tingkat tinggi dan pembuktian konsep, atau panduan tentang teknologi terbaik. Arsitektur juga dapat dilihat sebagai pemetaan nilai bisnis ke dalam persyaratan teknis atau pengembangan kasus penggunaan. Bagi saya, arsitektur adalah fondasi yang membangun organisasi. Anda tidak dapat mengharapkan hasil yang berkualitas jika kode Anda tidak

dibangun dan diatur untuk kualitas.

Apakah Anda setuju dengan itu?" "Oh ya," kata Kevin. "Desain yang buruk dapat mengacaukan Anda untuk waktu yang lama. Jika Anda memilih teknologi yang salah, kode Anda sulit diintegrasikan. Anda hampir perlu melakukan refaktor pada semuanya." "Itulah sebabnya Scrum mengubah cara kita memandang rekayasa," kata Duane. "Kami tidak memulai dengan melakukan banyak penelitian dan membuat dokumen arsitektur, yang sangat besar. Seperti yang saya yakin Anda ketahui, kami sekarang bekerja dengan kisah epik dan cerita pengguna yang kami upayakan untuk tetap sekecil mungkin. Tidak ada desain besar di awal. Seperti yang saya katakan, tim Scrum menyempurnakan arsitektur seiring dengan perkembangan produk." Cassie menimpali: "Itulah sebabnya kami memiliki tim arsitektur di sini. Tugas mereka adalah bekerja dengan tim Scrum dan memengaruhi bagaimana arsitektur berkembang.

Setiap tim Scrum memiliki pimpinan teknologi, atau arsitek yang harus bekerja sama dengan mereka, atau lebih disukai bagian dari tim arsitektur untuk mendapatkan arsitektur yang ditetapkan dalam Backlog." "Jadi, kisah-kisah ini diprioritaskan?" tanya Kevin. "Ya," jawab Duane. "Mereka diperlakukan seperti item Backlog lainnya." "Baiklah, tetapi bagaimana saya memulainya?" tanya Kevin. Duane tersenyum dan bersandar di kursinya. "Kevin, apakah kamu pernah mendengar tentang spike?" "Tidak bisa bilang kalau aku pernah," jawab Kevin. "Spike adalah cerita kecil yang kamu gunakan untuk melakukan riset. Saya biasanya bersikeras bahwa spike tidak menghabiskan seluruh Sprint. Hasil spike seharusnya cukup untuk menulis cerita." "Maksudmu salah satu cerita arsitektur yang kamu bicarakan?" tanya Kevin. "Mungkin," jawab Duane. "Atau cerita pengguna biasa yang membangun nilai." Hanya karena kita melakukan Scrum bukan berarti tidak ada arsitektur.

Scrum bergantung pada apa yang disebut arsitektur emergen. Arsitektur emergen muncul (karena itu namanya) saat pekerjaan dilakukan, dan berasal dari dalam tim Scrum. Biasanya ditemukan selama penyempurnaan cerita. Memiliki tim teknik yang bekerja dengan tim Scrum untuk menciptakan visi memastikan bahwa tim Scrum melihat gambaran besarnya. Kerja sama antara pengembang, penguji, arsitek, dan yang lainnya membantu menciptakan dan menangkap ide-ide baru. Saya pernah mendengar bahwa arsitektur adalah olahraga tim. Saya sangat setuju. Melihat bagaimana perkembangan di Sprint saat ini memengaruhi keseluruhan arsitektur. Ini adalah hal yang baik. Arsitek harus membantu tim mengidentifikasi arsitektur apa yang diperlukan dan membantu mereka menerapkannya. Ini adalah pendekatan bottom-up yang memungkinkan tim menentukan spesifikasi desain dan arsitektur. Bab ini merinci tanggung jawab dan fungsi inti Scrum Master sebagai orang ketiga melalui sudut pandang Scrum Master fiktif yang bekerja dengan tim Scrum baru. Di bab berikutnya, kita akan mengeksplorasi interaksi Scrum Master dengan peran lainnya

BAB 7
INTERAKSI SCRUM MASTER DENGAN PERAN LAIN

Bagi saya, Scrum Master lebih mementingkan soft skills daripada kemahiran teknis atau menjadi sangat terorganisasi. Berikut ini pendapat saya tentang bagaimana Scrum Master harus berinteraksi dengan berbagai peran.

7.1 PRODUCT OWNER

"Yang perlu Anda lakukan adalah menghilang selama beberapa hari." Saya bekerja dengan Product Owner baru yang mencoba menyusun rencana rilis, tetapi terseret ke berbagai arah. Setiap kali ia meluangkan waktu untuk mengerjakan rencana rilis, ada sesuatu atau orang lain yang membutuhkan waktunya. Solusi saya untuk dilema ini adalah dengan "berdiam diri". Kami masuk ke ruang konferensi dan menutup pintu. Dia tidak melihat emailnya, mematikan perangkat lunak pengiriman pesannya, dan mengerjakan rencana rilis bersama saya. Sebenarnya saya menyuruhnya untuk memberi tahu semua orang bahwa dia sakit, tetapi dia tidak melakukannya. Kalau dipikir-pikir lagi, itu bukan sikap jujur dan transparan, jadi saya senang dia tidak melakukannya. Kami fokus untuk menyelesaikan rencana rilis dan berhasil. Itu contoh yang agak ekstrem.

Saya biasanya melatih Pemilik Produk agar sebisa mungkin siap sedia untuk tim pengembangan, tetapi ini kasus khusus. PO memiliki banyak tanggung jawab yang membutuhkan banyak dukungan dari Scrum Master. PO memiliki pekerjaan yang berat. Saya menyebut mereka orang yang bisa diandalkan. Satu-satunya orang yang bertanggung jawab kepada pemangku kepentingan dan tim. Scrum Master perlu mendukung PO dan membantu mereka menciptakan produk hebat dengan fitur yang tepat. Mendukung PO adalah salah satu cara terbaik bagi Scrum Master untuk memimpin tim. Saya suka mengatakan bahwa Scrum Master adalah asisten Product Owner.

Peran Scrum Master dalam Membimbing Product Owner untuk Meningkatkan Kolaborasi dan Fokus pada Nilai Pelanggan

Scrum Master harus melatih Product Owner agar selalu siap sedia bagi tim. PO harus melakukan apa pun yang mungkin untuk melibatkan pelanggan dengan tim, tetapi tidak merugikan tim. Jika PO hanya ada untuk perencanaan Sprint dan demo Sprint, mereka merugikan tim. Tim harus dapat mengajukan pertanyaan kepada PO saat mereka membangun apa yang diminta. Saat saya mengajar Scrum, saya suka menjalankan simulasi. Simulasi memberi siswa kesempatan untuk menggunakan apa yang telah mereka pelajari, dan membuat semua orang bersemangat dan bergerak. Saya suka meminta siswa membangun sesuatu dengan balok plastik kecil yang dapat disatukan.

Saya merasa bahwa meminta kelas melakukan sesuatu yang tidak memerlukan pengodean atau teknologi memungkinkan siapa pun untuk ikut serta tanpa memandang latar belakangnya. Sulit juga untuk membuat orang yang bekerja di bidang teknologi tidak memikirkan teknologi. Misalnya, saya biasa menjalankan simulasi yang meminta kelas untuk

membangun situs web hipotetis untuk toko pizza. Tidak ada pengodean yang terlibat, tetapi para siswa akan berdebat tentang jenis server web apa yang akan digunakan atau cara terbaik untuk mengode JavaScript. Itu tidak terjadi dengan blok plastik. Yang terjadi adalah tim menjadi begitu asyik dengan pekerjaan mereka sehingga mereka lupa tentang PO.

Mereka membangun struktur dua lantai dan tidak menanyakan detail seperti berapa banyak jendela, atau di mana pintu seharusnya berada atau apakah struktur tersebut memerlukan kamar mandi. Hal yang sama terjadi dengan tim Scrum yang bekerja dalam satu iterasi. Bertindak sebagai Pemilik Produk saat saya menolak pekerjaan mereka, ini adalah kesempatan yang bagus untuk mengingatkan siswa bahwa membangun sesuatu yang tidak diinginkan pelanggan adalah buang-buang waktu, dan PO adalah suara pelanggan. Pastikan bahwa PO tersedia bagi tim dan bahwa tim berinteraksi dengan PO.

Bangun Backlog

Hal terpenting yang dilakukan Pemilik Produk adalah membangun Backlog. Seperti yang saya katakan sebelumnya, PO tidak menulis semua cerita dalam Backlog. Namun, PO bertanggung jawab atas setiap cerita dalam Backlog. Pemilik produk harus memprioritaskan ulang Backlog sesering mungkin. Saya ingin melihatnya terjadi setiap hari. Ya, mungkin berlebihan, tetapi saya ingin PO memikirkan cerita dalam Backlog. Memprioritaskan ulang Backlog setiap hari menjamin bahwa PO sangat familier dengan cerita dalam Backlog. Itulah yang sangat penting. PO harus sangat familier dengan cerita dalam Backlog sehingga ia dapat secara efektif mengartikulasikan nilai pelanggan yang terkandung di dalamnya.

Scrum Master perlu memastikan bahwa PO mampu memfokuskan energinya pada Backlog. Scrum Master harus membimbing Product Owner untuk menciptakan tujuan dengan cerita pengguna seperti Cassie yang membimbing Annie di Bab 6. Terlalu sering, saya melihat cerita yang menekankan bagaimana tim membangun sesuatu. PO harus fokus memberi tahu tim apa yang diinginkan pengguna dan, yang lebih penting, mengapa mereka menginginkannya. "Mengapa" itu penting. Tim perlu melihat cerita pengguna melalui sudut pandang pelanggan. Dengan menjelaskan dengan jelas mengapa pelanggan menginginkan hal ini, PO membantu tim fokus memberikan nilai tersebut. Sebagai Scrum Master, saya dikenal sering hadir dalam rapat Refinement dan terus bertanya mengapa pelanggan menginginkan hal ini.

Peran Scrum Master dalam Mendukung dan Membimbing Product Owner untuk Meningkatkan Kinerja Tim

Sebagai Scrum Master, saya ingin membantu Product Owner sebisa mungkin. Saya akan turun tangan dan melakukan apa pun yang saya bisa untuk mempermudah pekerjaan PO. Saya tidak akan benar-benar menulis cerita, tetapi salah satu cara Scrum Master dapat membantu adalah dengan memfasilitasi rapat. Misalnya, fasilitasi Sprint Review dan demo. PO hadir di sana, dan melakukan banyak presentasi. Ya, tim pengembangan melakukan demo, tetapi PO adalah orang yang berinteraksi dengan pelanggan dan pemangku kepentingan. Wajar bagi PO untuk bertindak sebagai semacam pembawa acara dalam rapat. Scrum Master dapat mengatur rapat, mengatur pemangku kepentingan, dan membantu menjalankan presentasi. PO tetap menjadi titik fokus, tetapi Scrum Master bertanggung jawab untuk

memfasilitasi.

Hal yang sama dapat dilakukan dengan rapat Penyempurnaan. Fasilitasi kapan pun Anda bisa. Ambil alih tanggung jawab menjalankan rapat dari Pemilik Produk sehingga mereka dapat fokus pada cerita bersama tim. Penyempurnaan cerita perlu dilakukan setidaknya seminggu sekali. Terlalu sering saya melihat hal pertama yang harus disingkirkan saat keadaan menjadi sibuk adalah rapat Penyempurnaan. Bekerja samalah dengan PO untuk memastikan bahwa tim menyempurnakan cerita setiap minggu. Scrum Master perlu memastikan bahwa PO diberdayakan untuk melakukan pekerjaan mereka. PO harus dapat membuat keputusan dengan cepat. Jika dia harus meminta masukan manajemen untuk setiap keputusan, tim tidak akan berhasil. Agility sering kali berhenti di tingkat tim. Scrum mengharuskan seluruh organisasi beroperasi dengan cara Agile. Manajemen perlu memberi PO wewenang untuk membuat keputusan dan kemampuan untuk menangani konsekuensinya. Scrum Master dapat membuat perbedaan nyata di sini.

Bekerja samalah dengan manajemen untuk memastikan bahwa mereka memahami pentingnya peran PO dan latih mereka untuk memercayai orang tersebut untuk mengubah persyaratan pelanggan menjadi cerita pengguna yang memberikan nilai. Saya selalu berusaha untuk sering berkonsultasi dengan Product Owner setiap hari, jika memungkinkan. Dengan cara ini, saya mengetahui apa yang ada di dalam pikiran mereka dan melakukan apa pun yang diperlukan untuk membantu mereka menjadi PO yang lebih matang. Suatu kali, saya pernah mendengar seorang PO mengatakan kepada saya bahwa akan sangat bagus jika mereka dapat memvisualisasikan seluruh Backlog. Saya menuliskan setiap cerita dalam Backlog pada catatan tempel dan menempelkannya di dinding ruang konferensi. Saya akui bahwa itu agak berlebihan, tetapi PO tersebut mampu memvisualisasikan Backlog.

PO akan selalu mendorong lebih banyak hal. Itulah sifat pekerjaan; lagipula, merekalah yang dapat dikekang. Scrum Master harus membimbing PO agar tim dapat membuat komitmen dan memenuhinya. Membebani tim secara berlebihan tidak akan menghasilkan apa pun selain masalah. PO dapat mendorong, tetapi Scrum Master harus menolak. Ingatkan PO bahwa ia harus berfokus pada hasil. Tim harus bekerja dengan kecepatan yang berkelanjutan, dan kualitas tidak boleh dikorbankan untuk memenuhi jadwal apa pun. PO harus mengizinkan tim menyelesaikan pekerjaan sementara ia meluangkan waktu untuk memastikan bahwa cerita siap untuk mereka. Begitu tim terbiasa untuk tidak terlalu berkomitmen, dan menyelesaikan cerita, mereka akan meningkatkan kecepatan.

7.2 TIM PENGEMBANG

Manusia diciptakan untuk berada dalam komunitas. Bahkan, kita mendambakannya. Kita tertarik pada individu yang berpikiran sama. Saya pernah mendengar bahwa tim Scrum adalah sebuah suku. Itu analogi yang bagus, tetapi saya lebih suka menganggap tim Scrum sebagai sebuah band.

Scrum sebagai Band Garasi

Sebuah band mencoba mencapai harmoni. Untuk melakukan ini, setiap musisi harus mengetahui peran mereka dan memainkan peran mereka. Gitar biasanya menjadi titik fokus

band. Gitar bass lebih dari sekadar gitar dengan dua senar lebih sedikit. Pemain bass menangani spektrum musik yang rendah dan menjadi fondasi dinding suara yang diciptakan sebuah band. Biasanya bass memainkan oktaf lebih rendah dari gitar. Memainkan nada yang sama dengan gitar tidak terdengar tepat. Bass dan drum menjadi fondasi musik dan menggerakkan ritme. Saat saya memainkan gitar bass, saya mencoba menjaga ketukan dengan bass dan snare drum milik drummer.

Saya ingin membuat drum dan bass hidup berdampingan sehingga gitar dan tuts piano dapat mencapai nada tinggi, dengan drum dan bass yang memberikan denyut di belakangnya. Jika dilakukan dengan benar, hasilnya akan bagus. Jika semua orang tidak menjalankan perannya, hasilnya akan berantakan. Scrum Master bertanggung jawab untuk menjaga ritme dan nada tim. Sama seperti grup musik, setiap anggota tim Scrum harus tahu cara memainkan peran mereka dan menciptakan harmoni. Ada alasan mengapa saya memilih band dan bukan orkestra. Musisi dalam orkestra memainkan musik persis seperti yang tertulis. Orang-orang dalam band bermain-main dengan not dan akord dan mencari tahu bagaimana semuanya cocok. Bermain lebih tentang kolaborasi daripada mengikuti musik secara persis.

Keharmonisan Scrum

Jadi, bagaimana Scrum Master dapat membantu tim mencapai harmoni? Pertama, pahami bahwa transformasi adalah sebuah proses. Memahami posisi tim Scrum dalam proses tersebut akan membantu Scrum Master membimbing dan meningkatkan kemampuan mereka. Misalnya, katakanlah Anda ingin mempelajari cara membuat roti lapis keju panggang yang lezat. Berikut adalah level yang akan Anda lalui:

- Kesadaran: Anda telah mencoba roti lapis keju panggang dan menyukainya. Anda memutuskan bahwa Anda mungkin ingin mempelajari cara membuatnya sendiri.
- Kompetensi: Anda dapat membuat roti lapis dan tidak akan gosong. Pada akhirnya, roti lapis Anda dapat dimakan.
- Kemahiran: Anda dapat membuat roti lapis keju panggang dengan sangat mudah, dan hasilnya benar-benar sangat lezat.
- Kompleksitas: Anda mulai bereksperimen dengan berbagai jenis keju dan roti dan beralih dari membuat roti tawar dan roti lapis keju Amerika menjadi roti lapis yang lebih lezat. Anda dapat mulai membantu dan membimbing orang lain dalam seni membuat roti lapis keju panggang yang lezat.
- Penguasaan: Anda adalah sumber "utama" untuk semua hal tentang keju panggang.

Mengapa saya berbicara tentang keju panggang? Memang, saya berorientasi pada makanan, tetapi ini adalah contoh sederhana dari tahapan yang akan dilalui tim saat mempelajari disiplin baru. Saya tidak berbicara tentang membuat dasbor dan laporan terlihat bagus. Saya ingin memberikan perangkat lunak hebat yang tidak sabar untuk dimiliki pelanggan kami. Scrum Master harus terus-menerus membimbing tim tentang cara menjadi lebih baik dalam Scrum. Penting bagi Scrum Master untuk membantu tim melihat diri mereka sendiri dengan jujur dan mengakui di mana mereka berada dalam proses ini. Jadi, daftar saya yang, eh, klise terlihat seperti ini saat saya menerapkannya pada Scrum:

- Kesadaran: Anda memiliki pemahaman tentang apa itu Scrum, dan Anda mungkin telah

mendiskusikannya dengan beberapa orang.

- Kompetensi: Anda sekarang "menjalankan Agile" yang pada kenyataannya berarti Anda mengadakan banyak rapat dan berusaha keras untuk membuat Scrum ini berhasil. • Kemahiran: Anda benar-benar menjadi ahli dalam hal ini. Anda menyampaikan fitur-fitur setiap Sprint, dan burn-down Anda tidak terlalu buruk. Sebagai bonus tambahan, tidak ada darah di dinding ruang rapat setelah Perencanaan Sprint.
- Kompleksitas: Anda mulai memodifikasi berbagai hal untuk membuat Scrum bekerja lebih baik bagi Anda dan tim Anda. Anda secara konsisten menunjukkan teknik Agile dan mengidentifikasi tempat untuk bereksperimen dengan praktik-praktik baru. Anda tidak lagi "melakukan Agile." Anda "menjalani Agile."
- Penguasaan: Tim menghasilkan nilai dengan kecepatan yang berkelanjutan sambil beradaptasi dengan kebutuhan pelanggan Anda yang terus berubah.

Mengetahui di mana tim berada di sepanjang jalur ini membantu Scrum Master memahami cara terbaik untuk melatih tim. Jika tim berada dalam tahap kompetensi, mereka tidak perlu memodifikasi apa pun dalam kerangka Scrum. Dengan kata lain, pagar pembatas harus agak kaku. Ingat, segala sesuatunya akan terasa aneh, dan tim akan ingin kembali ke praktik Waterfall yang sudah mereka kenal. Seiring dengan kematangan tim, mereka akan mulai melihat nilai Scrum dan menerima nilai dari upacara tersebut. Seiring dengan tim yang melakukan inspeksi dan adaptasi, pagar pembatas dapat menjadi lebih longgar karena tim tidak akan kembali ke cara Waterfall yang sebelumnya nyaman. Tim akan ingin bereksperimen dan mencari tahu cara terbaik untuk menjadi lebih Agile dan mencapai penguasaan.

7.3 PERAN SCRUM MASTER: PEMIMPIN DAN PENGGERAK TIM

Scrum Master adalah pemimpin pelayan tim. Pada kenyataannya, Scrum Master tidak memiliki wewenang. Dengan kata lain, Scrum Master tidak dapat membuat tim melakukan apa pun. Itu tidak berarti bahwa Scrum Master tidak dapat efektif. Rahasianya adalah bahwa Scrum Master perlu meyakinkan tim bahwa dia "berkomitmen penuh" bahwa satu-satunya alasan mereka datang bekerja setiap hari adalah untuk membuat tim sukses. Saya suka mengatakan bahwa saya adalah penggemar dan penyemangat nomor satu tim. Sebagai seorang Scrum Master, saya menghabiskan banyak waktu untuk menyemangati anggota tim, mengamati, dan mendengarkan. Saya suka mengamati dinamika tim dan mengubah beberapa hal untuk meningkatkan kolaborasi dan kerja sama tim. Misalnya, jika seorang anggota tim tidak ikut serta dalam rapat, saya akan mencoba mengarahkan pembicaraan kepada mereka.

Jika seseorang dalam tim berbicara tentang beberapa subjek yang sangat teknis, saya mungkin akan meminta mereka untuk mengklarifikasi, meskipun saya tidak tahu apa yang mereka bicarakan. Saya juga terus-menerus mengukur nilai percakapan tim. Apakah ini sesuatu yang seharusnya ada di tempat parkir, atau apakah penting untuk membiarkannya begitu saja? Saya juga percaya pada menjadi "pembangun tim C dan C." C dan C adalah singkatan dari ucapan selamat dan perayaan. Kapan terakhir kali seseorang mengucapkan "terima kasih" dan mengejutkan Anda? Saya tidak berbicara tentang saat teller bank mengucapkan terima kasih kepada Anda setelah dia menyelesaikan transaksi Anda. Maksud

saya ketika seseorang dengan tulus ingin mengucapkan terima kasih atas sesuatu yang telah Anda lakukan. Bagaimana perasaan Anda? Manusia perlu merasa dihargai. Penghargaan menegaskan bahwa pekerjaan yang telah Anda lakukan dihargai.

Mengetahui hal ini, saya selalu mencari cara untuk menunjukkan penghargaan. Ketika tim mencapai sesuatu, bahkan ketika itu adalah sesuatu yang seharusnya mereka lakukan, saya mengucapkan selamat kepada mereka atas pekerjaan yang telah dilakukan dengan baik. Misalnya, jika tim menyelesaikan semua cerita yang mereka janjikan dalam Sprint, saya akan mengucapkan selamat kepada mereka atas pekerjaan yang telah dilakukan dengan baik. Ada kekuatan dalam penghargaan, dan itu sangat jarang terjadi dalam budaya bisnis saat ini. Rayakan setiap kali memungkinkan, dan bukan hanya untuk alasan "normal" seperti hari ulang tahun. Jangan salah paham di sini. Seperti yang saya katakan sebelumnya, Scrum Master harus mengetahui ulang tahun setiap anggota tim dan mengadakan pesta ulang tahun tim. Saya pikir itu penting karena menunjukkan bahwa kita peduli dengan manusia. Bagaimanapun, sumber daya tidak memiliki hari ulang tahun.

Bawa ke tingkat berikutnya. Rayakan keberhasilan tim. Dalam contoh ini, setelah memberi selamat kepada tim karena telah menyelesaikan semua cerita dalam Sprint, ajak mereka minum kopi. Yang saya maksud dengan kopi bukanlah barang-barang di ruang istirahat. Keluarlah selama sekitar satu jam dan nikmati waktu bersama tim. Tentu saja, dapatkan persetujuan dari manajer fungsional sebelum mengusulkannya kepada tim. Itu seharusnya tidak menjadi masalah, karena manajer fungsional juga harus tertarik untuk membangun dinamika tim yang baik. Saya ingat berbicara dengan seorang anggota tim yang dengan santai menyebutkan bahwa dia dan istrinya sedang menantikan kelahiran anak pertama mereka. "Jadi, kapan istrimu akan melahirkan?" tanya saya. "Minggu depan," jawabnya. Saya terkejut karena saya belum pernah mendengar tentang ini sebelumnya, jadi saya mencari tahu dan menemukan bahwa dia tidak memberi tahu siapa pun di tim tentang acara ini.

Memang dia adalah orang yang "hanya mementingkan bisnis", tetapi saya tetap heran karena dia tidak menceritakan peristiwa penting dalam hidupnya kepada kami. Saya mengumpulkan tim (tanpa anggota tim yang memiliki anak) dan mengusulkan agar kami mengumpulkan hadiah dan memberikan hadiah kepada orangtua dan anak baru tersebut. Tim tersebut tidak hanya setuju, mereka juga bersemangat melakukannya. Setelah anak tersebut lahir, kami memberinya hadiah yang telah kami beli. Ayah baru tersebut tersentuh bahwa tim akan melakukan hal seperti itu. Saya tahu dampak dari hal seperti ini. Dulu ketika putri saya Jorden lahir, tim tempat saya berada melakukan hal yang sama. Saya tahu bagaimana perasaan saya saat itu, dan saya ingin orang lain merasa senang seperti saya saat itu. Melakukan hal-hal seperti ini membangun ikatan antar anggota tim.

Apakah Anda pernah memperhatikan bahwa lini serang tim sepak bola selalu duduk bersama saat mereka tidak berada di lapangan? Biasanya bukan karena pelatih yang menyuruh mereka duduk bersama. Jangan salah paham. Dia ingin mereka bersama-sama sehingga dia dapat menemukan mereka saat dia perlu berbicara dengan mereka, tetapi mereka tidak perlu diberi tahu. Mereka melakukannya secara alami, biasanya berdasarkan posisi. Mereka adalah tim dalam tim, dan ikatan di antara mereka sangat kuat. Sebagai Scrum Master, saya mencoba

membangun persahabatan dan keakraban di antara anggota tim. Saya sangat percaya pada pentingnya mengunjungi setiap anggota tim setiap hari. Saya hanya ingin melihat bagaimana keadaannya, dan apakah ada cara saya dapat membantu.

Sebagai Scrum Master, saya ingin membangun hubungan baik dengan setiap anggota tim. Saya ingin mereka memercayai saya dan percaya bahwa saya memperhatikan mereka. Saya ingin membangun tim dengan segala cara yang memungkinkan. Saya ingin tim menjadi sinkron (harmonis lagi). Saya ingin mencapai titik di mana mereka mengetahui kecenderungan dan ritme satu sama lain sehingga mereka tahu cara bekerja sama dengan cara yang menghasilkan kinerja tinggi.

Keseimbangan Kerja-Hidup

Seperti yang diceritakan, tidak seorang pun pernah berkata saat terbaring di ranjang kematian bahwa mereka berharap dapat menghabiskan lebih banyak waktu di tempat kerja. Seperti yang telah saya katakan sebelumnya, Scrum berbeda. Tim berkomitmen pada pekerjaan, lalu menyelesaikannya. Sebagai Scrum Master, pastikan bahwa pekerjaan yang dipilih anggota tim tidak akan terlalu berat bagi mereka. Saya selalu mengingatkan tim bahwa satu-satunya hal yang Anda dapatkan dari bekerja berjam-jam dan di akhir pekan adalah kelelahan. Penting untuk melatih tim agar menjaga keseimbangan antara apa yang baik untuk orang tersebut, baik untuk perusahaan, dan baik untuk tim.

Scrum Master Lainnya

Bagaimana Anda menjadi lebih baik dalam menjadi Scrum Master? Saya telah ditanya lebih dari sekali apa yang harus dilakukan untuk menjadi Scrum Master yang lebih baik. Saya biasanya menjawab seperti yang Anda harapkan. Pelajari keahliannya. Baca buku dan blog. Hadiri konferensi dan seminar. Jalin hubungan pembinaan dan, yang terpenting, jadilah anggota komunitas Scrum Master. Bergaullah dengan orang-orang yang memiliki pemikiran yang sama sehingga Anda dapat belajar dari mereka, dan mereka dapat belajar dari Anda. Peran Scrum Master dapat menuntut. Akan sangat membantu jika memiliki jaringan pendukung Scrum Master lain untuk bersandar.

Komunitas

Berkumpullah dengan Scrum Master lain dan bicarakan tentang apa yang terjadi di tim Anda. Perspektif yang berbeda memberikan wawasan tentang situasi yang mungkin tidak terlihat oleh orang yang terlibat di dalamnya. Seperti pepatah lama dua kepala lebih baik daripada satu. Hal lain yang sering saya tekankan adalah bahwa apa yang terjadi dalam rapat Scrum Master tetap berada dalam rapat Scrum Master. Jaga kerahasiaan percakapan antara Scrum Master. Jangan biarkan mereka meninggalkan ruangan. Anda harus dapat saling terbuka dan jujur. Bicarakan tentang kesulitan masing-masing. Berikan pendapat dan bagikan solusi. Dengan melakukan ini, Anda akan memperoleh wawasan tentang cara melayani tim Anda dengan lebih baik. Misalnya, mungkin ada ratusan cara untuk melakukan rapat Retrospektif. Saya suka membicarakan teknik apa yang berhasil dengan baik dan mana yang tidak berjalan dengan baik. Anda juga dapat membicarakan apa yang terjadi dengan masing-masing tim Scrum, tetapi berhati-hatilah.

Jangan bagikan apa pun yang tidak ingin dibagikan oleh tim. Hargai batasan tim, tetapi

bagikan. Salah satu hal yang selalu membuat saya kagum tentang angkat beban adalah cara para pengangkat beban memperlakukan satu sama lain. Anda akan berpikir bahwa sekelompok orang yang saling berkompetisi akan saling menatap seperti serigala lapar yang melihat daging panggang. Tidak demikian. Semua orang ada di sana untuk berkompetisi, tetapi kita semua saling menjaga. Para pengangkat beban membutuhkan banyak bantuan pada hari-hari ketika mereka berkompetisi. Mereka membutuhkan bantuan dengan peralatan, seperti mengencangkan ikat pinggang yang berat, atau membalut lutut sebelum mencoba squat, atau bantuan untuk mengenakan pakaian pendukung yang sangat ketat. Saling mendukung. Ketika seorang Scrum Master sedang sakit atau berlibur di pantai di suatu tempat, turun tangan dan fasilitasi pertemuan untuk mereka. Mungkin bahkan membeli donat untuk tim Scrum orang lain sesekali.

Pelatih Agile

Setiap orang punya zona nyaman. Ya, itu adalah tempat yang nyaman dengan sedikit tekanan atau risiko. Itu bukan tempat yang buruk. Kebanyakan orang dapat tampil konsisten saat berada di zona nyaman mereka; namun, Anda tidak akan pernah mencapai performa tinggi jika tidak keluar dari zona nyaman. Salah satu hal yang dilakukan pelatih Agile adalah mendorong Anda untuk mengembangkan diri dan mengeluarkan Anda dari zona nyaman. Mungkin cara yang lebih baik untuk mengatakannya adalah pelatih dapat mendorong Anda keluar dari zona nyaman.

Keluar dari Zona Nyaman

Ketika saya melatih bola basket putri, saya sengaja membuat para gadis menggiring bola dua kali lebih banyak dengan tangan mereka yang tidak dominan. Saya mencoba membuat mereka lebih sulit untuk dipertahankan. Apa yang terjadi dalam bola basket (dan dengan tim Scrum) adalah ketika berada di bawah tekanan, para pemain kembali ke apa yang mereka percayai. Mereka kembali ke zona nyaman mereka. Jika Anda hanya percaya menggiring bola dengan, katakanlah, tangan kanan Anda, itulah yang akan Anda lakukan. Jika saya mencoba membela Anda, saya akan tahu bahwa saya dapat mencari bola di sisi kanan Anda saat saya memberikan tekanan. Itu membuat bola lebih mudah dicuri, terutama jika Anda mencoba bergerak ke kiri. Akhirnya, beberapa dari mereka menjadi sangat ahli dengan tangan nondominan mereka sehingga mereka mempercayai kemampuan mereka di kedua tangan. Seorang pelatih yang baik dapat membuat Scrum Master bertanggung jawab sambil mendorong mereka untuk mencoba hal-hal baru dan belajar dari kegagalan. Scrum Master berfokus pada tim. Pelatih Agile berfokus pada gambaran yang lebih besar.

Pelatih bukan anggota tim Scrum mana pun, dan berfokus pada organisasi. Ia mendukung Scrum Master dan tim yang mereka layani. Pelatih biasanya jauh lebih berpengalaman daripada Scrum Master dan dapat menerapkan pengetahuan dari berbagai metode seperti Scrum, Lean, dan Kanban. Seorang pelatih terutama berfokus pada Scrum Master, tetapi dapat melatih dan membimbing siapa pun dari tim hingga manajemen dan eksekutif. Seorang pelatih yang baik seperti ragi. Mereka menyebar ke semua orang dan membantu mereka tumbuh. Hubungan Scrum Master dengan pelatih seharusnya seperti apa adanya. Bayangkan Sensei dan murid. Pelatih ada untuk membantu Scrum Master tumbuh dan

belajar. Selama beberapa tahun terakhir, saya menghabiskan Sabtu pagi bersama Tony Pharr, yang ditunjukkan pada Gambar 7.1. Kami berdua kembali ke pusat kebugaran, tetapi kami tidak bersiap untuk kompetisi. Ya, kami tidak mempersiapkan diri untuk kompetisi. Kami bekerja dengan generasi berikutnya.

Kami melatih beberapa saudara Tony yang lebih muda dan keponakan saya Max. Mereka berada di ruang angkat beban untuk bersiap menghadapi musim sepak bola mendatang. Tony kembali menggunakan trik lama, memberi tahu mereka bahwa mereka harus kalah untuk menang sambil mengukur usaha mereka saat mereka berusaha melawan palang. Saya mengubah beban dan menentukan kedalaman squat sambil menambahkan dorongan saya sendiri. Kami benar-benar mendorong mereka keluar dari zona nyaman mereka, dan sebagai hasilnya mereka menjadi pemuda yang kuat.

Gambar 7.1 Tony Pharr, Juara Dunia Angkat Beban

Jangan remehkan kekuatan pembinaan. Baik Agile coach maupun Scrum Master akan melatih sebagai bagian dari aktivitas mereka sehari-hari, dan pembinaan mungkin merupakan hal terpenting yang akan dilakukan salah satu dari mereka. Jake Turner adalah paman sahabat saya. Orang yang membimbing saya dan teman-teman saya serta mengajari kami semua tentang angkat beban. Saya masih berbicara dengan Jake Turner sesekali. Sekarang di usia awal tujuh puluhan, ia tidak lagi mengangkat beban, tetapi masih memiliki kecerdasan dan binar di matanya yang saya ingat dari dulu ketika ia membawa sekelompok anak ingusan ke garasinya dan mengajari mereka cara mengangkat beban.

Ia heran bahwa saya akan menulis tentangnya di buku ini. Ia tidak berpikir bahwa ia melakukan sesuatu yang istimewa untuk saya. Wah, ia salah besar. Sadar atau tidak, apa yang Jake lakukan untuk saya adalah contoh hebat tentang apa yang dilakukan seorang coach. Seorang coach melihat nilai dalam diri setiap orang dan membantu mewujudkan nilai itu. Di masa ketika saya tidak terlalu memikirkan diri sendiri, ia melatih saya untuk menjadi hebat. Tidak, saya tidak pergi ke Olimpiade atau memenangkan kejuaraan dunia dalam angkat beban.

Namun, saya menyadari bahwa saya dapat melakukan apa pun jika saya bertekad. Sekarang saya melakukan hal yang sama dengan tim Scrum.

Manajer

Scrum Master dan manajer harus memiliki pemahaman yang sama. Peran mereka serupa. Keduanya berupaya memberi tim apa yang dibutuhkan untuk meraih keberhasilan. Perbedaan besarnya adalah Scrum Master tidak memiliki wewenang. Manajer memiliki wewenang. Scrum Master harus berbagi hampir semua hal dengan manajer. Satu-satunya hal yang tidak saya bagikan dengan manajer tim adalah hal-hal dari Retrospektif yang membuat tim tidak ingin meninggalkan ruangan. Hal yang sama berlaku untuk informasi yang dibagikan dalam sesi pelatihan yang disepakati untuk tidak meninggalkan ruangan. Semua hal lainnya dibagikan. Saya ingin memberi manajer perspektif saya tentang cara kerja internal tim. Manajer terkadang kesulitan dengan peran mereka dalam Agile. Mereka tidak lagi memberi tahu orang lain apa yang harus dilakukan.

Tim sekarang memutuskan bagaimana mereka akan menjalankan bisnis mereka. Peran manajer masih berharga. Hanya saja tidak lagi didasari oleh perintah dan kendali. Tim tidak perlu dikelola secara mikro. Peran manajer dalam Scrum adalah untuk memelihara pengembangan profesional anggota tim. Manajer mengadakan rapat tatap muka rutin dengan anggota tim. Manajer memberikan bimbingan dan pelatihan dalam rapat-rapat ini. Manajer juga mengurus keuangan tim. Mereka juga bertanggung jawab atas tinjauan kinerja. Manajer juga dapat membantu Scrum Master dengan menghilangkan hambatan. Karena mereka memiliki wewenang, manajer benar-benar dapat membantu Scrum Master di area ini.

7.4 PERAN SCRUM MASTER DENGAN PEMANGKU KEPENTINGAN

Saya telah menyebutkan pemangku kepentingan lebih dari satu kali hingga saat ini dalam buku ini. Saya biasanya berbicara tentang pengguna dan pemangku kepentingan, tetapi saya tidak pernah mengidentifikasi siapa pemangku kepentingan itu. Ini sama sekali bukan daftar yang lengkap. Saya akan merinci peran yang paling saya kenal.

Manajer Proyek

Manajer Proyek memberikan dukungan menyeluruh untuk proyek.Pikirkan Scrum di tingkat tim: tim menghasilkan nilai, Scrum Master melindungi tim dan meningkatkan proses Agile, dan PO berinteraksi dengan pelanggan dan membangun Backlog. Siapa yang mengurus hal-hal lain seperti memastikan kami sejalan dengan tata kelola perusahaan sehingga kami dapat merilis? Manajer proyek. Manajer proyek melacak hal-hal seperti memastikan dukungan dilatih dengan benar, bahwa semua persyaratan hukum yang diperlukan untuk merilis perangkat lunak terpenuhi, bahwa proyek tepat waktu, dan mungkin banyak hal lain yang tidak saya ketahui.

Scrum Master perlu terus memberi tahu manajer proyek tentang apa yang terjadi dengan tim. Biasanya Scrum Master atau Pemilik Produk akan berbagi informasi seperti burn-up rilis dan kecepatan tim dengan manajer proyek untuk membantu memberikan gambaran yang akurat apakah rilis tersebut dalam cakupan atau tidak. Saya suka memikirkannya seperti ini: Scrum Master melaporkan kepada tim tentang kesehatan proyek, dan manajer proyek

melaporkan melalui bisnis.

Sponsor

Terkadang dikenal sebagai sponsor bisnis, sponsor proyek memberikan paparan dan terkadang motivasi bagi tim di tingkat tinggi. Biasanya, mereka mengendalikan dana proyek, jadi mereka adalah orang yang sangat tertarik pada hal-hal seperti laba atas investasi (ROI). Kenyataannya, setiap orang dalam tim bertanggung jawab kepada sponsor. Scrum Master sebenarnya tidak banyak berinteraksi dengan sponsor, atau para sponsor. Jika suatu hambatan berada di luar kendali tim, sponsor dapat terlibat, karena mereka seharusnya memiliki wewenang untuk membantu. Manajer proyek biasanya adalah orang yang berinteraksi dengan sponsor.

Dukungan

Orang-orang yang terlibat ketika pelanggan memiliki masalah adalah sumber daya yang belum dimanfaatkan. Dulu saya pernah menjadi perwakilan dukungan. Dahulu kala, saya mendefinisikan peran saya sebagai "orang yang menjawab telepon ketika Anda dalam masalah." Perwakilan dukungan tidak lagi terlalu sering menggunakan telepon, tetapi mereka berinteraksi dengan pelanggan. Jenis pelanggan yang tepat. Yang saya sebut sebagai "on the glass." Orang-orang yang benar-benar menggunakan apa yang dihasilkan tim Scrum. Scrum Master harus melibatkan staf pendukung dalam sebanyak mungkin upacara Scrum. Mereka dapat menghadiri rapat stand-up harian untuk mendapatkan gambaran tentang apa yang sedang dibangun oleh tim. Mereka harus hadir di demo Sprint sehingga mereka dapat melihat fungsionalitas yang sedang berjalan.

Mereka harus berbicara dengan PO tentang jenis masalah pelanggan yang mereka lihat sehingga PO memiliki gambaran tentang beberapa kesulitan yang dialami di lapangan. Ini bukan hanya tentang cacat, tetapi pengalaman pelanggan secara keseluruhan. Perangkat lunak dapat bebas dari cacat, tetapi jika proses instalasi merupakan cobaan berat, pelanggan mungkin tidak senang. PO dapat mengenali hal ini dan membuat cerita Backlog untuk mengatasinya. Scrum Master harus berbicara dengan perwakilan dukungan secara teratur untuk memastikan bahwa mereka mendapatkan pelatihan yang mereka butuhkan tentang perangkat lunak baru yang sedang diproduksi. Sering kali, dukungan adalah "wajah" perusahaan. Pastikan mereka memiliki apa yang mereka butuhkan untuk membuat pelanggan senang.

Pengguna

Pemilik Produk tidak memiliki akses eksklusif ke pengguna. Saya melatih tim yang bekerja dengan saya agar tidak takut berbicara dengan pengguna kapan saja selama iterasi. Kami ingin pengguna menjadi bagian dari proses pengembangan, jadi masuk akal jika mereka dapat berbicara dengan siapa pun di tim saat mereka tersedia. Itu termasuk Scrum Master. Katakan pada diri sendiri, "Lihat, saya tidak akan dapat berbicara tentang detail teknis atau fitur produk, tetapi saya ada di sana untuk mendukung tim. Jika saya dapat membantu pelanggan, saya akan melakukannya." Dalam bab ini, kita melihat bagaimana Scrum Master berinteraksi dengan peran Scrum lainnya. Scrum Master adalah pemimpin pelayan tim terlebih dahulu, dan juara Agile bagi semua orang. Dalam bab berikutnya, kita akan mengeksplorasi

soft skills yang dibutuhkan Scrum Master.

BAB 8
KEAHLIAN LUNAK SCRUM MASTER

Seorang Scrum Master adalah pemimpin-pelayan tim Scrum. Mudah diucapkan, tetapi apa artinya sebenarnya? Bagi saya, seorang Scrum Master harus menjadi pelatih dan pendorong perubahan yang konstan. Tugas Scrum Master adalah menjaga api Agile tetap menyala. Untuk terus mengingatkan tim tentang apa yang ingin mereka capai, dan melatih mereka agar benar-benar mencapainya. Adik saya, Jason, baru-baru ini beralih karier. Dia telah menjadi guru sekolah dasar dan pelatih sepak bola selama yang saya ingat. Bayangkan keterkejutan saya ketika saya mendapat panggilan telepon suatu hari di mana dia bertanya kepada saya apa itu Product Owner, karena dia telah bergabung dengan sebuah perusahaan perangkat lunak. Product Owner adalah peran barunya. Ya, dia banyak berbicara dengan kakak laki-lakinya sejak saat itu... Salah satu hal yang menjadi jelas bagi Jason dengan sangat cepat adalah bahwa ada perbedaan antara pelatihan profesional dan apa yang Anda lihat dalam olahraga.

8.1 PELATIH PROFESIONAL

Apa yang terlintas di benak Anda saat saya mengucapkan kata "pelatih?" Biasanya, gambaran yang muncul di benak adalah kursi-kursi yang dilempar melintasi lapangan basket, atau pelatih sepak bola yang wajahnya memerah saat memarahi wasit setelah keputusannya tidak sesuai harapan. Saya pernah mengalami pelatihan yang baik dan yang saya anggap sebagai pelatihan yang buruk dalam hidup saya. Saya pernah memiliki pelatih sepak bola yang mengatakan kepada saya untuk "menahan diri" saat saya terbaring di lapangan dengan pergelangan kaki yang patah. Itu bukan akhir dari semuanya. Di akhir latihan, tim meninggalkan lapangan. Saya tidak bisa. Lapangan latihan terletak di bawah lapangan. Untuk kembali ke ruang ganti, Anda harus menaiki tangga lalu berjalan menanjak. Karena cedera saya, saya tidak bisa menaiki tangga. Saya terjebak di lapangan latihan dengan semua perlengkapan saya. Pelatih saya meninggalkan saya di sana. Akhirnya, beberapa teman saya datang mencari saya. Mereka membantu saya tertatih-tatih ke ruang ganti sehingga saya bisa mengganti perlengkapan saya.

Seperti yang mungkin Anda duga, kurangnya perhatian dan empati dalam situasi ini tidak berdampak positif pada jiwa remaja awal saya. Pelatih itu, yang namanya tidak dapat saya ingat, memberi dampak pada saya. Saya juga memiliki pengalaman pelatihan yang positif. Jake Turner, yang saya anggap sebagai pelatih angkat beban pertama saya, biasa mengatakan bahwa gerakan squat saya menyerupai monyet yang sedang melakukan sesuatu dengan bola yang tidak dapat saya tulis di buku ini. Anggap saja itu buruk... sangat buruk. Dalam angkat beban, squat adalah salah satu dari tiga gerakan yang digunakan untuk menunjukkan kekuatan. Untuk melakukan squat, Anda meletakkan barbel yang dibebani di punggung dan menekuk lutut dalam-dalam. Gerakan ini dinilai berdasarkan kedalamannya.

Lipatan pinggul Anda harus berada di bawah bagian atas tempurung lutut saat dilihat dari samping. Awalnya, saya tidak bisa mencapai kedalaman dengan sapu di punggung saya, apalagi dengan barbel yang dibebani. Alih-alih menganggap saya tidak berdaya, ia melatih saya. Ia biasa mengatakan bahwa kaki saya cukup kuat, saya hanya tidak tahu cara menggunakannya. Ia melatih saya, memberi saya kalimat yang tepat, dan membangun kepercayaan diri saya. Jake menanamkan kepercayaan diri dan pengetahuan kepada saya, dan hasilnya adalah seorang atlet angkat beban yang cukup bagus, jika boleh saya katakan.

Salah satu momen yang paling membanggakan bagi saya adalah ketika Jake melihat saya sebagai atlet angkat beban terakhir yang melakukan squat dalam kontes angkat beban. Itu berarti saya mencoba squat terberat hari itu. Tidak ada monyet atau bola di sekitar hari itu terutama karena ia meluangkan waktu untuk melatih saya. Melatih bukan tentang meniup peluit dan menyuruh orang berlari beberapa putaran ketika mereka melakukan kesalahan. Bagi saya, melatih adalah tentang mengeluarkan kehebatan dari orang-orang. Untuk menjadi pelatih yang efektif, ingat satu hal: Setiap orang berharga. Itu saja. Setiap orang yang Anda temui sepanjang hari memiliki sesuatu untuk diberikan. Tugas Anda sebagai pelatih adalah mengeluarkannya dari mereka.

8.2 MEMBANTU TIM MENERIMA AGILE

Jujur saja, sebagian besar tim tidak menerima Agile pada awalnya. Terserah kepada Scrum Master untuk membuat mereka memahami dan menyadari manfaat Agile. Saya suka menggunakan teknik yang disebut beaconing. Untuk mengilustrasikan apa yang saya bicarakan, saya ingin menggunakan kisah Alkitab tentang Yosua. Menurut cerita, ketika kedua belas suku Israel menyeberangi sungai Yordan menuju tanah perjanjian, Yosua meminta pemimpin setiap suku mengambil sebuah batu dari sungai dan menambahkannya ke tumpukan batu di tepi seberang. Ketika semua orang telah menyeberangi sungai, ada dua belas batu yang ditumpuk. Ketika ditanya mengapa ia meminta suku-suku tersebut melakukan tindakan ini, Yosua berkata bahwa batu-batu ini merupakan monumen untuk hal besar yang telah terjadi pada hari itu.

Di masa mendatang, ketika anak-anak Anda bertanya mengapa tumpukan batu itu ada di sana, Anda dapat menceritakan kepada mereka kisah tentang kedua belas suku yang menyeberangi sungai Yordan. Saya mencoba melakukan hal yang sama dengan tim Scrum. Tidak, saya tidak menyuruh mereka menumpuk batu. Saya terus mengingatkan mereka tentang hal-hal baik yang telah terjadi. Ketika tim memiliki pengalaman yang baik dengan Agile, itu adalah sebuah sinyal. Katakanlah tim sedikit mengubah apa yang sedang dibangun sebagai reaksi terhadap umpan balik pemangku kepentingan.

Saya akan mengingatkan tim tentang perubahan perilaku itu setiap kali ada kesempatan. Ketika semuanya berjalan dengan baik, saya akan mengarahkannya ke sinyal tersebut. Ketika semuanya tidak berjalan sebaik yang diharapkan, saya akan mengarahkannya ke sinyal tersebut. Saya terus-menerus dan sengaja mengingatkan tim bahwa mereka menjadi lebih Agile. Terus-menerus mengingatkan tim bahwa mereka dapat berhasil di Scrum menyebabkan budaya berubah. Kemenangan kecil membangun kepercayaan. Biarkan

kemenangan kecil itu mulai menumpuk, dan tim akan mulai mempercayai prosesnya. Pekerjaan akan mulai mengalir. Kemudian tim akan sampai pada titik di mana mereka dapat membuat lebih banyak keputusan, lebih cepat.

Berkomunikasi di Awal dan Sering

Transparansi adalah salah satu pilar Scrum. Terkadang, bersikap transparan tidak semudah kedengarannya. Misalnya, sangat sulit untuk membuat tim benar-benar terbuka saat keadaan tidak berjalan baik. Ini membutuhkan kepercayaan, dan banyak tim Scrum memiliki masalah kepercayaan dari pengalaman sebelumnya dengan perintah dan kendali. Pikirkan ini. Lima pemain lini ofensif yang sangat besar melindungi quarterback yang jauh lebih kecil. Para pemain lini perlu memberi quarterback waktu untuk melempar bola, menggerakkan tim di lapangan, dan mencetak gol. Quarterback harus percaya bahwa para pemain lini akan melakukan tugas mereka, sehingga ia dapat fokus menyelesaikan operan. Jika tidak ada kepercayaan yang terjalin, mata quarterback akan tertuju pada para pemain bertahan yang menyerang, bukan di lapangan. Hal-hal buruk akan terjadi. Perintah dan kendali bermuara pada kurangnya kepercayaan setidaknya menurut saya.

Ketika tim tidak dipercaya, manajemen mikro terjadi. Manajer akan menjadi sangat tertarik pada siapa yang melakukan apa setiap hari, bukan pada bagaimana tim tersebut bekerja. Ada sisi emosional dari kepercayaan juga. Anda tahu, di mana Anda mengekspos kekurangan Anda dengan harapan keterbukaan Anda tidak akan dimanfaatkan. Meminta tim untuk beroperasi secara transparan bukanlah hal yang nyaman bagi tim, mungkin karena dosa-dosa perintah-dan-kendali di masa lalu. Jika dipikir-pikir, Agile sebenarnya dibangun atas dasar kepercayaan. Alih-alih seorang direktur atau manajer yang "mengelola" proyek perangkat lunak, tim Scrum diberi banyak kebebasan. Tim Scrum-lah yang menentukan bagaimana pekerjaan akan dilakukan dan kapan pekerjaan akan selesai. Pemilik Produk perlu memercayai tim dalam hal ini. Jika PO berubah menjadi manajer pengembangan, perintah dan kendali mulai merayap kembali ke dalam dinamika tim.

Hal ini menyebabkan silo tim, moral yang rendah, dan peningkatan tekanan kerja, dan ini akan mengurangi kualitas perangkat lunak. Seorang Scrum Master yang baik perlu membantu tim mengatasi masalah kepercayaan dan menjadi lebih transparan. Komunikasi sangat penting. Tempat yang paling jelas untuk terjadinya hal ini adalah rapat tatap muka harian. Setiap hari, tim harus keluar dari rapat 15 menit itu dengan pemahaman yang lengkap tentang apa yang sedang terjadi. Tetapi bagaimana dengan orang-orang yang bukan bagian dari tim? Tentu, siapa pun dapat menghadiri rapat tatap muka harian, tetapi saya melatih tim saya untuk bersikap sangat transparan sehingga siapa pun, mulai dari CEO hingga petugas kebersihan, tahu apa yang sedang terjadi.

Saya suka menggunakan radiator informasi untuk melakukan ini. Informasi tentang kemajuan kolektif setiap tim Scrum harus ditampilkan. Seperti yang saya katakan, "ditampilkan dengan bangga, bahkan dengan cara yang tidak pantas di tempat umum." Radiator harus menunjukkan informasi Sprint seperti papan Scrum, bagan burn-down Sprint, bagan burn-up rilis, dan hambatan apa pun yang sedang diatasi oleh Scrum Master. Berkomunikasilah secara berlebihan sepanjang waktu, dalam setiap percakapan yang Anda lakukan dengan setiap orang

yang Anda temui.

Mendengarkan dengan Responsif sebagai Pemimpin Pelayan

Seperti yang saya katakan sebelumnya, istri saya mengatakan bahwa saya tidak mendengarkannya. Tentu saja saya tidak setuju. Bagaimanapun, saya duduk tepat di sebelahnya... Tetapi, dia benar. Saya tidak mendengarkannya. Saya menceritakan pertandingan bisbol tadi malam di kepala saya, atau memikirkan sesuatu yang jenaka untuk dikatakan, atau mungkin saya hanya melamun. Intinya adalah bahwa saya tidak fokus padanya. Perhatian saya ada di tempat lain. Multitasking membunuh percakapan. Untuk benar-benar mendengar apa yang seseorang coba katakan, Anda perlu memfokuskan perhatian penuh pada orang tersebut. Tentu saja, itu berarti meletakkan ponsel Anda. Saat berbicara dengan seseorang, saya selalu meletakkan ponsel saya menghadap ke bawah di atas meja.

Jika tidak ada tempat untuk meletakkan ponsel, saya akan menaruhnya di saku. Saya tidak ingin ada yang mengalihkan perhatian saya dari orang yang saya ajak bicara, dan itulah yang dilakukan ponsel pintar. Jika saya memeriksa email, mengirim pesan, atau memperbarui media sosial selama percakapan, saya tidak memberikan perhatian penuh kepada orang yang berbicara. Saya tidak mendengarkan. Jika mantra pembinaan saya benar dan setiap orang benar-benar berharga, maka mereka layak mendapatkan perhatian Anda. Namun, menyingkirkan gangguan dan fokus pada percakapan tidaklah cukup. Mendengarkan dengan baik juga mengharuskan Anda untuk mematikan otak dan benar-benar fokus pada apa yang sedang dikatakan. Saat saya sedang mengobrol, otak saya terus bekerja.

Begitulah saya. Saya harus melawan godaan untuk pergi dan memikirkan solusi, atau apa yang akan saya katakan saat giliran saya berbicara, atau apa yang akan saya makan. Saat saya mencoba untuk benar-benar fokus pada orang yang sedang berbicara, percakapan di kepala saya seperti ini: “Saya heran apakah kita perlu…” “Hentikan! Kamu melakukannya lagi…” “FOKUS!”Saya perlu terus fokus pada pembicara dan tidak membiarkan pikiran saya melayang. Saya juga suka mengulang apa yang dikatakan pembicara dan memantulkannya kembali kepada mereka. Pada titik-titik tertentu dalam percakapan, saya akan mengatakan, "Pemahaman saya tentang apa yang Anda katakan adalah…” dan saya akan memparafrasekan apa yang baru saja dikatakan pembicara. Ini memberi pembicara kesempatan untuk memverifikasi pemahaman saya tentang percakapan tersebut.

8.3 MENGGUNAKAN ALAT YANG TEPAT DALAM PEMBINAAN TIM

Seorang tukang kayu yang baik memiliki kotak peralatan dan tahu peralatan mana yang harus digunakan dalam situasi tertentu. Ayah saya memiliki perusahaan konstruksi. Ketika saya masih remaja, saya bekerja untuk ayah saya di musim panas. Saya di sini untuk memberi tahu Anda, pekerjaan konstruksi bukan untuk saya. Saya tidak memiliki keterampilan yang diperlukan untuk melakukan pekerjaan itu. Saya berterima kasih kepada ayah saya atas musim panas yang canggung yang saya habiskan dengan memukul ibu jari saya dengan palu dan jatuh dari atap. Mereka menunjukkan kepada saya bahwa saya tidak akan berhasil dalam bisnis keluarga. Alih-alih konstruksi, saya menekuni teknologi informasi. Saya mempelajari beberapa hal selama saya bekerja di bisnis konstruksi.

Salah satunya adalah pepatah ayah saya: "Persiapan!" Ayah saya akan mengatakan itu setiap pagi sebelum krunya naik untuk bekerja di atap. Idenya adalah memiliki semua peralatan yang Anda butuhkan untuk hari itu. Seorang pelatih memiliki peralatan yang siap digunakan. Salah satu peralatan adalah pembinaan. Ketika membina, pembina memahami bahwa orang yang dibina sudah mengetahui jawaban yang mereka cari. Hal itu perlu dipelajari dari mereka. Alat lainnya adalah pendampingan. Saya suka menggambarkan pendampingan sebagai "sudah pernah mengalaminya."

Saat menjadi pendamping, pelatih memiliki pengalaman dengan jenis situasi yang dipertanyakan, dan menjelaskan bagaimana masalah tersebut diselesaikan di masa lalu. Terakhir, ada pengajaran. Pengajaran adalah tempat pelatih mentransfer pengetahuan kepada siswa. Seperti tukang kayu, pelatih yang baik tahu kapan pembinaan, pendampingan, atau pengajaran cocok untuk suatu situasi. Terkadang, situasi mengharuskan ketiga teknik tersebut digunakan. Pastikan bahwa sebagai pelatih, Anda menggunakan alat yang tepat untuk pekerjaan tersebut.

Memutus Perkelahian

Saya bekerja sebagai tukang pukul untuk menambah penghasilan saya sebelum menikah. Pekerjaan itu persis seperti yang Anda bayangkan. Saya menghabiskan sebagian besar waktu saya di pintu depan untuk memeriksa identitas semua orang sebelum mereka memasuki kelab malam. Saya harus memastikan bahwa tidak seorang pun yang di bawah umur bisa masuk. Ketika perkelahian terjadi, saya disebut sebagai "WMD" atau senjata pemusnah massal. Ini bukan karena keterampilan bertarung saya yang luar biasa (yang tidak saya miliki). Melainkan karena ukuran tubuh saya. Ketika saya muncul, perkelahian biasanya berhenti. Pekerjaan saya sebagai tukang pukul bukanlah untuk memukul orang. Melainkan untuk mengeluarkan pelanggan yang berkelahi, atau akan berkelahi, dari kelab sebelum mereka menyebabkan kerusakan atau cedera fisik.

Saya suka mengatakan bahwa saya telah melerai lebih banyak perkelahian sebagai Scrum Master daripada sebagai tukang pukul. Mungkin bukan perkelahian, tetapi orang-orang dalam tim Scrum tentu mengalami banyak perselisihan yang hebat. Terkadang orang mengalami hari yang buruk dan bertindak dengan cara yang kurang profesional. Terkadang, perilaku Waterfall lama muncul misalnya, perselisihan antara pengembang dan penguji. Terus terang, terkadang orang-orang menjadi panik. Saya suka berpikir bahwa kebanyakan orang bukanlah psikopat, jadi perilaku seperti itu biasanya memiliki akar penyebab. Orang-orang memiliki keluarga dan tekanan baik di tempat kerja maupun di luar tempat kerja yang dapat menyebabkan mereka kewalahan. Ketika seseorang mengalami hari yang buruk, biasanya ada alasan di baliknya. Hal yang sama berlaku ketika dua anggota tim berselisih.

Biasanya, ada alasan di balik perselisihan tersebut. Terserah kepada Scrum Master dan pelatih untuk membantu orang-orang ini mengatasi masalah ini. Seorang Scrum Master adalah seorang pelatih. Ya, ada peran Agile Coach, dan tugas orang tersebut adalah untuk melatih Scrum Master dan tim Scrum. Namun, Scrum Master tertanam dalam tim. Mereka akan terbiasa dengan orang-orang dan kepribadian dalam tim Scrum dan harus membangun tingkat kepercayaan yang memungkinkan pengalaman pembinaan yang lebih berdampak daripada

yang dapat diberikan oleh Agile Coach. Seorang Scrum Master harus berusaha untuk membina siapa pun dalam tim jika memungkinkan. Ketika konflik muncul, hal pertama yang harus dilakukan adalah membatasi kemungkinan kerusakan. Perasaan dapat dengan mudah terluka, tetapi konflik juga dapat menjadi hal yang sehat. Faktanya, tim yang kolaboratif harus menerima konflik yang sehat.

Seorang Scrum Master harus membiarkan konflik terjadi selama konflik tersebut sehat. Biarkan saya menjelaskan pemikiran itu. Biarkan konflik terjadi... sampai pada titik tertentu. Batasi waktu acara tersebut sehingga anggota tim lainnya tidak duduk dalam rapat menyaksikan kedua orang ini berdebat selama satu jam. Jika konflik berlangsung terlalu lama atau berubah dari sehat menjadi menyakitkan, turun tangan dan minta tempat parkir. Menghentikan acara akan meredakan eskalasi apa pun dan memungkinkan para peserta untuk sedikit tenang. Pada titik tertentu, orang-orang yang berkonflik harus duduk berhadapan dan mencari cara untuk menyelesaikan situasi tersebut. Scrum Master harus sangat berhati-hati. Jika pihak-pihak yang terlibat belum siap untuk rapat tatap muka, jadwalkan beberapa rapat tatap muka dengan masing-masing individu baik dengan Scrum Master atau manajer.

Beri orang ruang untuk mengomel, tetapi batasi juga waktu untuk ini. Saya biasanya memberi seseorang waktu sepuluh menit, lalu mengatakan sesuatu seperti, "Sekarang setelah Anda mengungkapkannya, langkah apa yang dapat kita ambil untuk mencapai hasil yang positif?" Bimbing semua orang menuju langkah-langkah positif yang akan menyelesaikan masalah. Bimbing anggota tim menuju situasi yang saling menguntungkan. Salah satu tanggung jawab Scrum Master adalah menyingkirkan hambatan. Bagi saya, konflik internal tim merupakan hambatan besar yang terkadang terabaikan. Masalah antara anggota tim, atau bahkan tim dan Pemilik Produk, perlu diungkap dan ditangani. Sebuah tim tidak akan pernah berkinerja tinggi jika ada disfungsi dalam jajarannya. Konflik akan menghasilkan resolusi atau kebuntuan. Anda ingin pihak-pihak yang berkonflik mencapai konsensus, yang berarti setiap orang dapat menerimanya dan mendukungnya untuk maju. Hal terakhir yang Anda inginkan adalah seseorang bergumam, "Tidak mungkin".

8.4 MENANGANI PERILAKU EMOSIONAL, SOMBONG, ATAU NEGATIF

Seberapa sering Anda melihat atlet profesional benar-benar kehilangan akal sehatnya? Saya berbicara tentang pemain sepak bola yang mengoceh kepada wasit dan mendapat penalti 15 yard yang akhirnya membuat timnya kalah dalam pertandingan. Pendingin air tampaknya cukup sering dipukuli di ruang istirahat pemain bisbol. Di semua tingkatan olahraga, rasa frustrasi menyebabkan pemain dan pelatih melempar peralatan, memaki ofisial, atau terlibat dalam berbagai praktik yang tidak terlihat profesional. Tidak ada perbedaan antara tim Scrum dan tim olahraga. Tim adalah tim, dan perilaku negatif bukanlah hal yang Anda inginkan di lingkungan tersebut. Biasanya ada alasan untuk jenis perilaku ini. Sembilan dari sepuluh kali, itu karena frustrasi. Scrum Master perlu mencari tahu apa yang membuat anggota tim frustrasi dan membuat rencana untuk mengatasinya. Misalnya, katakanlah salah satu pengembang senior di tim tersebut bertindak tidak senonoh.

Scrum Master harus bertemu dengan mereka sambil minum kopi dan melihat apakah

semuanya baik-baik saja. Selama percakapan, anggota tim tersebut berbagi bahwa mereka khawatir akan dianggap tidak berguna jika mereka menyerahkan keahlian mereka. Tidak ada yang lebih jauh dari kebenaran. Tim Scrum membutuhkan orang-orang dengan kemampuan coding, dan seorang pengembang senior mencapai posisi tersebut karena kemampuan mereka untuk mendesain dan menulis kode. Yang telah berubah adalah bahwa kita sekarang menghargai kerja tim dan kolaborasi. Perubahan adalah hal yang menakutkan, dan dapat membuat orang frustrasi. Begitu pula dinamika tim, manajemen, dan banyak hal lainnya. Selidiki akar permasalahannya dan ambil langkah untuk mengatasinya. Namun, tegaskan bahwa tim tidak akan menoleransi perilaku buruk yang terus berlanjut.

Saya pernah melatih beberapa tim yang menggunakan kartu kuning dan merah untuk membantu mengendalikan perilaku. Kartu ini didasarkan pada kartu penalti yang dikeluarkan oleh wasit dan wasit dalam permainan sepak bola. Dalam sepak bola, kartu kuning adalah peringatan. Misalnya, kartu kuning dikeluarkan saat seorang pemain bertindak tidak sportif. Saat seorang pemain mendapat kartu kuning, mereka dapat tetap bermain. Kartu merah dikeluarkan untuk pelanggaran mencolok, dan pemain tersebut dikeluarkan dari permainan. Kami menggunakan kartu kuning sebagai cara untuk menghentikan rapat. Kartu kuning menarik perhatian pada perilaku mengganggu seseorang, dan biasanya mengarah pada diskusi tim. Bendera merah adalah pilihan terakhir. Jika kartu merah dikeluarkan, rapat segera dihentikan. Dengan senang hati saya laporkan bahwa saya tidak pernah mengalami kartu merah dikeluarkan.

Mendekati Orang yang Pendiam

Beberapa orang secara alamiah menghindar dari pusat perhatian. Mereka tidak nyaman untuk berbicara. Entah itu bukan "kesukaan" mereka, atau ada semacam rasa takut yang berperan dalam situasi ini. Takut ditolak, takut berbicara di depan umum, mungkin mereka takut berbicara karena kurang percaya diri. Apa pun masalahnya, seorang Scrum Master perlu memfasilitasi upacara Agile dengan cara yang mendorong komunikasi bagi seluruh tim. Membiarkan hanya sebagian dari tim mendominasi percakapan berarti merugikan tim. Untuk menjadi tim yang sehat dan berkinerja tinggi, semua orang perlu berpartisipasi. Beberapa teknik dapat digunakan untuk memastikan bahwa seluruh tim terlibat.

Banyak aktivitas tim Scrum dirancang untuk memungkinkan seluruh tim berpartisipasi misalnya, seluruh tim mengajukan tebakan poin cerita mereka pada saat yang sama, atau semua orang menempelkan catatan tempel untuk setiap bagian retrospektif. Satu hal yang dapat dilakukan oleh Scrum Master adalah memandu percakapan kepada orang-orang yang tidak berpartisipasi. Cukup minta mereka untuk berbagi pemikiran mereka. Teknik lain yang dapat digunakan oleh Scrum Master adalah memberi seseorang yang mencoba mendominasi percakapan sebuah pekerjaan. Misalnya, mintalah mereka untuk menuliskan ide setiap orang pada sebuah sticky note dan kemudian Anda mengambil mereka dan menempelkannya pada flipchart. Ingat, setiap orang dalam tim itu berharga. Scrum Master perlu membuat setiap anggota tim merasa bahwa mereka adalah anggota tim yang berharga.

Mencerahkan Suasana

Ketika saya diminta untuk melakukan presentasi, saya suka memulainya dengan

lelucon. Lelucon dapat mencairkan suasana dan membuat audiens tidak terlalu defensif. Hal yang sama berlaku untuk tim Scrum. Scrum Master yang baik tahu cara membaca situasi dan kapan melontarkan lelucon atau melakukan sesuatu yang konyol adalah hal yang dibutuhkan tim.Saya pernah bekerja dengan manajer yang akan marah jika kami mengizinkan orang mengubah latar belakang desktop mereka. Itulah yang saya sebut sebagai manajemen pusat data lama. Manajer yang kaku, yang terbiasa dengan perintah dan kendali, memandang orang yang bekerja untuk mereka sebagai sumber daya dan bersikap seperti Tyrannosaurus Rex. Mereka tidak akan pernah mengizinkan perayaan tim dalam bentuk apa pun. Hal ini tercermin dari moral tim secara umum. Ketika tim mencapai tonggak sejarah seperti rilis, adakan pesta. Ajak tim makan siang. Buat mereka merasa dihargai.

Memfasilitasi Pengorganisasian Diri dan Lintas Fungsi

Saat Anda meminta tim untuk bekerja lintas fungsi, Anda meminta mereka untuk menyisihkan sebagian keahlian mereka agar menjadi lebih serba bisa. Dulu, anggota tim yang ahli dihargai. Kini, keahlian tetap dihargai, tetapi perlu dibagikan. Seperti yang saya katakan, anggota tim perlu menjadi lebih luas dan tidak terlalu mendalam. Tidak ada yang salah dengan menjadi ahli; namun, tim lintas fungsi perlu terdiri dari anggota yang dapat terjun langsung dan mengerjakan apa pun. Tidak diinginkan jika hanya memiliki satu orang yang ahli dalam fungsi atau modul kode tertentu. Bagaimana jika orang ini menang undian besok dan tidak pernah terdengar kabarnya lagi? Bagaimana jika mereka sakit atau pergi berlibur panjang dan tidak bisa bekerja? Scrum Master perlu mendorong tim untuk menjadi sefamiliar mungkin dengan setiap bagian dari apa yang mereka kerjakan.

Gunakan teknik seperti pemrograman berpasangan, di mana dua pengembang mengerjakan hal yang sama secara bersamaan sambil duduk bersebelahan. Biasanya satu pengembang mengetik saat mereka berkolaborasi bersama, sehingga mereka memiliki pemahaman yang sama tentang kode tersebut. Hal yang sama berlaku untuk pengujian. Ketika saya pertama kali mulai berkecimpung di bidang rekayasa perangkat lunak, para penguji menguji semuanya secara manual. Waktu sistem mahal, terutama pada mainframe. Sekarang sudah tidak demikian lagi. Dengan munculnya teknologi seperti ZPDT, sistem lebih mudah diakses daripada sebelumnya. Para penguji harus berpendapat bahwa pengujian yang konstan mengurangi variabilitas. Pengujian harus diotomatisasi jika memungkinkan dan dijalankan secara terus-menerus.

Rencana pengujian harus ditulis sesegera mungkin dalam proses pengembangan. Menurut pendapat saya, tim harus mengambil langkah-langkah untuk bergerak menuju pengembangan berbasis pengujian, di mana otomatisasi pengujian ditulis terlebih dahulu, kemudian kode ditulis sehingga lulus pengujian. Tim perlu beralih dari pembuat kode dan penguji menjadi anggota tim. Pengembang dapat menguji. Bisakah penguji membantu di mana saja dan menulis kode? Semua anggota tim perlu mengembangkan keterampilan mereka sehingga tim menjadi benar-benar lintas fungsi. Tim Scrum perlu menjadi lebih lintas fungsi sehingga mereka dapat menghadapi lebih banyak tantangan.

Biaya untuk menjadi lebih lintas fungsi adalah anggota tim kehilangan keahlian. Dorong anggota tim untuk menghadiri pertemuan komunitas praktik, tempat sekelompok

pakar dalam subjek tertentu berkumpul untuk membahas ide dan memperdalam pengetahuan mereka. Saya melatih Scrum Master untuk berkumpul seminggu sekali untuk berbicara tentang metodologi Agile. Pengembang dan penguji harus melakukan hal yang sama. Berkumpul dan biarkan besi menajamkan besi.

Melatih Anggota Tim dan Mengembangkan Kompetensi Mereka

Tim Scrum perlu memiliki semua keterampilan yang diperlukan untuk memberikan nilai pada setiap Sprint. Untuk melakukan ini, beberapa kapasitas perlu disisihkan untuk mempelajari keterampilan baru bahasa pemrograman baru, atau menjadi terbiasa dengan sistem operasi baru, atau apa pun. Tim Scrum perlu mampu meningkatkan diri. Terlalu sering, tim hanya memikirkan penyelesaian Backlog dan lupa mempersiapkan keterampilan untuk masa depan. Tidak ada kapasitas yang disisihkan untuk menjadi lebih baik. Scrum Master perlu mengingatkan tim bahwa pengembangan profesional itu penting dan memastikan bahwa Product Owner memahami bahwa ini adalah situasi di mana kita mengorbankan sedikit kecepatan sekarang untuk melaju lebih cepat nanti.

Mempromosikan Kemajuan Karier Anggota Tim

Scrum Master harus menjadi advokat bagi tim Scrum dan anggotanya. Saya telah diberitahu oleh berbagai orang di posisi manajemen atas bahwa mereka sangat senang melihat orang-orang mereka bekerja dengan baik. Bagian dari menjadi pemimpin pelayan adalah mengutamakan kepentingan tim. Untuk menggunakan istilah hip hop, Scrum Master adalah penyemangat tim. Penyemangat adalah rapper cadangan yang tugasnya mendukung rapper utama. Tugas penyemangat adalah membuat penonton bersemangat dan mempertahankannya. Menyunting dan mensosialisasikan semua yang dilakukan anggota tim Scrum. Dorong anggota tim untuk mengejar posisi baru dan meningkatkan karier mereka. Lakukan apa pun yang dapat Anda lakukan untuk membantu setiap anggota tim mencapai kesuksesan profesional. Rayakan keberhasilan ini. Bagaimanapun, ketika tim menang, kita semua menang…

Melangkah Maju dan Mundur

Salah satu hal yang menjadi tantangan bagi saya adalah kapan harus menghentikan percakapan dan kapan harus membiarkannya berlanjut. Sebagai fasilitator rapat, saya ingin memastikan bahwa rapat tersebut bermanfaat bagi seluruh tim dan tidak melebihi batas waktu yang disepakati. Sebagai Scrum Master, saya ingin tim berkolaborasi dan melakukan percakapan tatap muka yang bermanfaat. Yang saya coba lakukan adalah mengenali saat hal ini terjadi dan menanyakan kepada tim apa yang ingin mereka lakukan. Jika mereka ingin membiarkan percakapan berlanjut, saya akan menyetel pengatur waktu lima menit. Di akhir lima menit, saya akan menanyakan kepada tim apakah mereka ingin melanjutkan percakapan selama dua menit lagi. Saya akan melanjutkan batas waktu dua menit ini hingga percakapan selesai, atau tim ingin melanjutkan di tempat parkir.

Memberikan Masukan

Jika saya belum cukup jelas sampai di titik ini, izinkan saya menyampaikannya sejelas mungkin. Scrum Master tidak menulis kode dalam keadaan apa pun. Scrum Master ada untuk melayani tim, memimpin tim, dan menantang mereka untuk menjadi lebih Agile dari hari ke

hari. Ya, Scrum Master adalah anggota tim. Mereka tidak terlibat dalam membantu tim melakukan pekerjaan mereka. Jaga agar tim tetap fokus pada pemeriksaan dan adaptasi.

Jaga Agar Tim Tetap Termotivasi, Bersemangat, Berdaya, dan Kohesif

Ada kekuatan nyata dalam sikap positif. Anda mungkin mengabaikan kekuatan berpikir positif, tetapi saya di sini untuk memberi tahu Anda bahwa itu menghasilkan keajaiban. Misalnya, saya dulu senang berbicara tentang orang-orang yang tertabrak bus. Saya akan mengatakan hal-hal seperti, "Bagaimana jika Gene tertabrak bus besok? Siapa yang bisa mengambil pekerjaannya jika Anda tidak mengerti apa yang dia lakukan?" Di permukaan, tidak ada yang salah dengan pernyataan itu. Jika Gene tertabrak bus, tim tidak akan melihatnya untuk sementara waktu. Itu tidak dapat disangkal; namun, itu juga negatif. Sekarang, saya lebih suka mengatakannya seperti ini: "Bagaimana jika Gene memenangkan lotre, membeli pulau pribadinya sendiri, dan kita tidak pernah melihatnya lagi? Siapa yang bisa mengambil pekerjaannya jika Anda tidak mengerti apa yang sedang dilakukannya?"

Kedengarannya jauh lebih baik, bukan? Saya suka menggambarkan diri saya sebagai "positif yang memuakkan." Saya biasanya tipe orang yang selalu melihat gelas setengah penuh. Saya hanya sedikit menaikkan levelnya. Itu menular ke tim. Pikirkan apa yang keluar dari mulut Anda, karena itu tidak hanya memengaruhi orang-orang di sekitar Anda, tetapi juga sikap Anda. Saya mencoba untuk tidak menggunakan kata-kata yang mengancam kesuksesan seperti mungkin, mungkin, tidak bisa, tidak akan, jika, Saya mencoba untuk menahan diri saat mengucapkannya dan menggantinya dengan kata-kata yang akan memungkinkan kesuksesan.

8.5 MELINDUNGI TIM DARI GANGGUAN

Salah satu tugas utama Scrum Master adalah melindungi tim dari gangguan luar sehingga mereka dapat fokus untuk mencapai tujuan Sprint. Ini lebih mudah diucapkan daripada dilakukan. Saya suka melatih Scrum Master untuk menjadi penyaring. Segala sesuatu yang masuk ke tim harus terlebih dahulu melewati penyaring, Scrum Master. Ketika saya mengatakan segalanya, saya benar-benar memaksudkan segalanya. Misalnya, tenaga penjualan mendapat tekanan dari pelanggan untuk menambahkan satu fitur itu ke produk sebelum mereka akan membeli. Perwakilan dukungan ingin mengajukan pertanyaan. Jika ada acara pembuktian konsep di lokasi pelanggan atau beberapa jenis demo, orang-orang di lapangan ingin pengembangan dilibatkan.

Saya tidak mengatakan bahwa tim Scrum harus mengabaikan permintaan seperti ini. Sama sekali tidak. Yang ingin saya katakan adalah bahwa semua yang dilakukan tim Scrum harus dimasukkan ke dalam Backlog dan diprioritaskan. Tim membuat komitmen pada rapat perencanaan Sprint. Ini adalah jumlah pekerjaan yang dapat mereka selesaikan. Tidak adil untuk menambahkan pekerjaan ke apa yang sudah ada di Sprint Backlog. Semua yang dikerjakan tim Scrum perlu diperiksa oleh Pemilik Produk. Setelah Pemilik Produk memahami permintaan tersebut, permintaan tersebut dapat dimasukkan ke dalam Backlog dan diprioritaskan. Bagaimana jika apa yang diminta memiliki prioritas yang sangat tinggi? Tidak apa-apa. Namun, jika tim perlu mengerjakannya sekarang, sesuatu perlu dihapus dari Sprint saat ini. Tidak seorang pun dapat menumpuk lebih banyak pekerjaan pada tim.

Mendidik Organisasi

Kesalahan yang dilakukan banyak organisasi adalah berfokus pada mendidik tim Scrum. Sekilas, ini adalah ide yang bagus. Tim Scrum harus mengetahui Scrum. Itu masuk akal. Namun, untuk mengubah budaya, seluruh organisasi memerlukan pelatihan Agile. Meskipun orang-orang tidak bekerja secara langsung pada tim Scrum, mereka perlu memahami Scrum. Seperti yang sering saya katakana setiap orang perlu berbicara dalam bahasa yang sama. Seorang Scrum Master harus melakukan webinar, membuat blog, membuat video edukasi menggunakan media apa pun yang tersedia untuk mengedukasi semua orang tentang manfaat Scrum. Seluruh organisasi perlu memahami kerangka kerja Scrum, karena seluruh organisasi perlu menjadi lebih Agile, bukan hanya tim pengembangan. Dalam bab ini, kita melihat beberapa soft skills yang dibutuhkan oleh peran Scrum Master. Mulai dari melerai pertengkaran hingga mencairkan suasana hingga melindungi tim dari gangguan, soft skills ini merupakan bagian penting untuk menjaga tim tetap senang dan puas. Dalam bab berikutnya, kita akan membahas keterampilan teknis Scrum Master.

BAB 9
KETERAMPILAN TEKNIS SCRUM MASTER

9.1 PERAN SCRUM MASTER: KETERAMPILAN DAN ALAT BANTU TIM

Scrum Master tidak dianggap sebagai peran teknis. Bahkan, saya berpendapat bahwa orang-orang nonteknis menjadi Scrum Master terbaik karena mereka dapat lebih fokus pada Scrum itu sendiri dan keterampilan interpersonal yang telah kita bahas di bab sebelumnya. Namun, ada beberapa keterampilan teknis yang perlu dimiliki oleh Scrum Master. Kita akan membahasnya di bab ini.

Hal terpenting yang perlu diingat adalah Anda harus memulai dari suatu tempat. Terlalu sering, saya melatih tim melalui penciptaan DoD atau DoR dan itu berubah menjadi frustrasi. Kelumpuhan karena analisis tim cenderung terlalu banyak berpikir tentang hal-hal seperti ini. Salah satu cara yang saya sukai untuk mencegah tim terlalu banyak berpikir adalah dengan menggunakan latihan catatan tempel. Setiap orang di tim mendapat catatan tempel dan pena. Saya meminta setiap anggota tim untuk menuliskan sesuatu yang ingin mereka lihat di DoD pada catatan tempel. Saya kemudian mengumpulkannya dan menempelkannya pada flip chart. Saya kemudian menghapus duplikat, dan mendiskusikan apa yang tersisa. Jika tim ingin menghapus apa pun, harus ada konsensus.

Setelah kita melewati rangkaian catatan tempel pertama, kita masuk ke babak kedua. Setelah beberapa babak, Anda seharusnya memiliki DoD yang bisa digunakan. DoD harus menjadi dokumen yang terus berkembang. Sebuah tim tidak boleh membuat Definisi Selesai dan tidak pernah mengubahnya. Misalnya, mungkin ada sesuatu yang muncul dari Sprint Retrospective yang mengharuskan DoD dimodifikasi. Saya sangat menyarankan agar Scrum Master meninjau kembali DoD dengan tim setiap beberapa Sprint atau lebih untuk memastikan tidak ada yang perlu ditambahkan atau dihapus.

Memilih Alat Bantu Tim

Sebagai Scrum Master, saya mencoba untuk tidak terpaku pada alat bantu tim. Secara pribadi, saya lebih suka menggunakan catatan tempel di dinding. Ini solusi yang sederhana, tetapi berhasil. Saya juga mengerti bahwa ini tidak realistis, terutama jika tim tidak berada di tempat yang sama. Biarkan tim memilih apa yang ingin mereka gunakan. Lakukan eksperimen dan cari tahu apa yang terbaik untuk tim dan situasi Anda. Akan selalu ada faktor eksternal bagi tim (terutama anggaran) yang akan memengaruhi keputusan akhir. Intinya adalah bahwa alat hanyalah itu alat bantu. Semua fitur canggih di dunia tidak akan membantu jika Anda tidak dapat menggunakan alat bantu tersebut untuk menyelesaikan apa yang perlu Anda lakukan.

Membangun Desain dan Arsitektur ke dalam Product Backlog

Salah satu prinsip Agile adalah, "Arsitektur, persyaratan, dan desain terbaik muncul dari tim yang mengorganisasikan diri sendiri." Baik desain maupun arsitektur dipandang sebagai sesuatu yang muncul dalam Scrum. Itu berarti kita menyadari bahwa hal-hal ini akan muncul saat cerita dikerjakan. Lakukan pekerjaan desain secukupnya di awal dan tambahkan arsitektur dalam cerita yang sesuai yang ditambahkan ke Sprint sesuai kebutuhan. Hal ini menyebabkan

orang-orang panik karena mereka merasa bahwa aspek rekayasa dari pekerjaan pengembangan diabaikan. Rekayasa harus terus berlanjut sepanjang siklus pengembangan. Agile berusaha sebaik mungkin untuk tidak melakukan penelitian dan desain yang berjalan lama di awal. Mulailah membangun sesuatu lebih awal dan cari tahu arsitekturnya seiring berjalannya waktu.

Dalam Scrum, kita menggunakan frasa seperti "minimal layak" dan "maksimalkan jumlah pekerjaan yang tidak dilakukan." Kita tidak suka melakukan banyak pekerjaan di awal yang mungkin harus kita buang. Arsitektur yang muncul datang dari dalam tim saat kebutuhan ditemukan. Misalnya, saat menyempurnakan cerita, menjadi jelas bahwa ada beberapa pekerjaan arsitektur yang harus dilakukan. Arsitek harus bekerja dengan tim Scrum untuk menentukan persyaratan arsitektur ini dan memasukkannya ke dalam Backlog sesegera mungkin. Pemilik Produk akan memprioritaskannya dengan pekerjaan lainnya. Arsitek harus bekerja dengan tim Scrum saat arsitektur muncul. Ia tidak boleh menulis kode secara langsung, tetapi harus bekerja dengan tim untuk memastikan bahwa desainnya konsisten dan memenuhi standar arsitektur.

9.2 REFACTORING

Refactoring adalah mengubah kode komputer internal tanpa mengubah perilaku eksternal. Dengan risiko terlalu menyederhanakan, saya menjelaskan refactoring sebagai membuat kode menjadi lebih baik tanpa mengubah perilaku produk. Saya suka memasak. Saya juga suka makan, jadi saya rasa kecintaan saya pada memasak sejalan dengan itu. Meskipun saya pikir saya cukup pandai memasak, saya buruk dalam membersihkan setelah saya selesai memasak. Saya asyik memasak dan tiba-tiba, saya dikelilingi oleh panci dan wajan yang kotor dan berbagai macam kekacauan yang harus dibersihkan.

Ibu mertua saya berbeda. Dia membersihkan sambil bekerja dan tidak harus berhenti di akhir untuk menghabiskan banyak waktu membersihkan. Itulah refactoring kode. Tujuannya adalah menjaga kode tetap bersih dan mudah dibaca. Ini bukan tentang memperbaiki bug atau menulis ulang sesuatu. Refactoring adalah tentang melakukan hal-hal seperti menghapus kode duplikat selama sesi pengodean. Kami tidak ingin kode menjadi besar atau sulit diatur saat kami melakukan iterasi atau beradaptasi dengan perubahan persyaratan. Saat kami menambahkan kode, kami terus-menerus melakukan refaktor untuk menghilangkan kerumitan.

Integrasi Berkelanjutan

Misalnya tim Anda baru saja menyelesaikan demo Sprint. Pelanggan sangat terkesan dengan apa yang telah dilakukan tim Anda sehingga mereka menginginkannya sekarang. Ya, Anda telah membangun produk yang layak secara minimal, tetapi pelanggan tidak meminta itu. Mereka menginginkan produk yang lengkap. Dengan kata lain, rilis. Menerbitkan produk yang layak secara minimal setiap dua minggu bisa jadi sulit dicapai. Mungkin tim Scrum perlu mengubah cara mereka menjalankan bisnis. Tujuan akhir dari integrasi berkelanjutan adalah dapat digunakan kapan saja. Untuk menjaga agar pembuatan produk, pengujian, dan infrastruktur rilis lainnya tetap mutakhir sebanyak mungkin. Sebagian besar tim yang bekerja

dengan saya memiliki trunk atau repositori utama yang menyimpan semua file sumber untuk proyek tertentu.

Anggota tim "memeriksa" atau membuat salinan modul di trunk dan menyalinnya ke stasiun kerja lokal mereka. Setelah modifikasi selesai, pengembang "memeriksa berkas" atau menyalinnya kembali ke repositori sumber. Sekarang berkas di repositori sumber harus berisi semua perubahan pengembang dan cocok dengan berkas di stasiun kerjanya. Biasanya ada beberapa jenis kontrol sumber yang tidak akan mengizinkan berkas sumber dikerjakan oleh lebih dari satu anggota tim secara bersamaan. Gabungkan (atau integrasikan) perubahan kode tim ke dalam repositori sumber terintegrasi setiap hari. Semakin lama waktu yangtim di antara penggabungan, semakin sulit menemukan masalah.

Integrasi berkelanjutan membawa hal-hal ke tahap selanjutnya. Ketika sesuatu dicentang ke sumber, pembangunan akan dimulai secara otomatis. Setelah pembangunan selesai, produk baru secara otomatis diinstal dan dikonfigurasi pada mesin uji integrasi berkelanjutan, dan rangkaian uji otomatis dijalankan terhadap perangkat lunak yang baru dibangun. Seluruh proses berjalan lancar, artinya proses ini sepenuhnya otomatis. Dengan cara ini, setiap perubahan diuji secara menyeluruh. Tim dapat yakin bahwa semua yang dicentang tidak hanya diuji secara menyeluruh, tetapi juga berfungsi bersama. Tidak perlu dikatakan lagi bahwa jika proses gagal, seluruh tim berhenti untuk memperbaikinya sebelum hal lain dilakukan.

Otomatiskan Proses Pembangunan

Sebagian besar bahasa pemrograman seperti C atau C++ mengharuskan kode dikompilasi. Anda mulai dengan biner atau file sumber yang diproduksi oleh tim Scrum dan berakhir dengan perangkat lunak yang dapat digunakan. Dengan kata lain, file sumber yang dapat dibaca manusia diubah menjadi objek yang dapat dibaca mesin yang dapat diinstal oleh pengguna akhir. File tersebut mencakup hal-hal seperti file .exe dan .dll pada komputer Windows. Proses pembangunan sederhana terlihat seperti ini:

- Kompilasi kode.
- Tautkan objek.
- Buat paket instalasi.
- Jalankan laporan.

Pembuatan dapat dilakukan secara otomatis, manual, atau gabungan keduanya. Pembuatan manual mengharuskan perintah dikeluarkan dan dijalankan satu per satu. Sebagian besar proyek menyertakan banyak berkas sumber. Pengembang dapat mengompilasi setiap berkas ini secara individual. Tidak berlebihan jika dikatakan bahwa pembuatan manual bisa jadi membosankan. Daripada membangun semuanya secara manual, tim Scrum dapat mengotomatiskan proses pembuatan sehingga seluruh proyek dapat dibangun dengan satu perintah. Tim saya menyebutnya "make." Program tersebut menelusuri daftar berkas sumber, yang disebut makefile, dan memanggil kompiler untuk setiap berkas.

Kecanggihan dapat dibangun ke dalam make sehingga hanya berkas yang diubah yang dikompilasi, atau sehingga program lain dapat dijalankan setelah dikompilasi. Mengotomatiskan proses pembuatan menjadikan integrasi berkelanjutan menjadi kenyataan.

Mampu membangun dengan menekan tombol atau memiliki acara seperti seseorang yang memeriksa kode ke manajemen sumber memulai pembuatan memungkinkan penerapan cepat.

9.3 PENGEMBANGAN BERBASIS PENGUJIAN

Idenya di sini adalah bahwa tidak ada kode yang dapat dipercaya. Awalnya hal itu tampak agak kasar, tetapi jika Anda melihat lebih jauh, itu masuk akal. Pengembangan berbasis pengujian (TDD) adalah tempat tim menulis pengujian kecil yang spesifik untuk cerita tertentu. Pengujian ditulis sebelum pengodean spesifik apa pun dilakukan untuk cerita tersebut. Saat pengujian pertama kali dijalankan, pengujian tersebut gagal. Kode kemudian ditulis untuk lulus pengujian dan tidak lebih. Pengembangan berbasis pengujian penerimaan (ATDD) lebih merupakan upaya kolaboratif. Ini melibatkan tim pengembangan, Pemilik Produk, dan bahkan orang-orang dari bisnis tersebut.

Saya bahkan pernah mendengar tentang pelanggan yang terlibat. Pengujian penerimaan ditulis sebelum kode sumber ditulis. Saat kode ditulis, kode ditulis untuk lulus pengujian. Untuk lebih jelasnya, saya tidak berbicara tentang kriteria penerimaan di sini. Kriteria penerimaan dapat digunakan untuk menulis pengujian penerimaan, tetapi pengujian penerimaan tidak boleh hanya didasarkan pada kriteria tersebut. Tim harus mempertimbangkan semuanya. Ini seharusnya tidak terlalu sulit untuk dipahami. Sebelum satu baris kode ditulis, tim melihat hal-hal berikut:

- Persona cerita
- Kasus penggunaan
- Kriteria penerimaan
- Apa pun yang mereka yakini diperlukan untuk memberikan nilai yang diminta

Pengujian penerimaan yang sangat terperinci kemudian ditulis. Pengujian ini memastikan bahwa nilai yang tepat diberikan melalui skenario pengujian apa pun yang ditetapkan. Hal ini tampaknya mengubah dunia, karena fokusnya sekarang adalah menulis pengujian. Kode ditulis hanya untuk lulus pengujian. Tim tidak menulis kode produksi apa pun kecuali mereka memiliki pengujian yang gagal. TDD adalah tentang memastikan kode berfungsi; misalnya, suatu fungsi memberikan kode pengembalian yang tepat.

ATDD lebih tentang hasil; misalnya, pengguna mengklik tautan dan diarahkan ke halaman yang benar. Awalnya, hal itu akan terasa tidak nyaman, aneh, dan canggung bagi tim. Mereka tidak akan terbiasa bekerja dengan cara seperti ini. Anda benar-benar perlu menunjukkan nilai TDD dan membuat tim ikut serta. Teknik-teknik ini hampir menjamin bahwa tidak akan ada cacat dalam kode produksi. Kedengarannya sangat berharga bagi saya.

Memperbarui Rencana Rilis Setelah Setiap Sprint

Rencana Rilis adalah dokumen yang menunjukkan apa yang direncanakan tim untuk disampaikan kepada pelanggan dan pemangku kepentingan. Kedengarannya cukup sederhana. Cukup beri tahu pelanggan tentang semua fitur luar biasa yang akan dibangun oleh tim Scrum, dan mulailah bekerja… Jangan terburu-buru. Ingat, kita perlu merencanakan variabilitas saat membangun perangkat lunak. Sebagai sebuah tim, kita perlu mengantisipasi

perubahan.“Pilih pembuka Anda agar Anda dapat ikut serta dalam perlombaan.” Saya mendengar pelatih angkat beban saya Jake Turner mengatakan itu ratusan kali. Ia berbicara tentang merencanakan upaya dalam perlombaan angkat beban. Saya pikir itu sejalan dengan cara menyiapkan rencana pelepasan untuk proyek Agile. Dalam olahraga angkat beban, seorang pengangkat melakukan tiga angkatan: squat, bench press, dan deadlift. Setiap peserta mendapat tiga kali percobaan untuk setiap angkatan.

Angkatan tertinggi yang berhasil dalam squat, bench press, dan deadlift dijumlahkan. Jumlah total angkatan tertinggi di setiap kelas berat menang. Jika seorang pengangkat tidak melakukan angkatan yang berhasil setelah tiga kali percobaan di salah satu dari tiga angkatan tersebut, ia didiskualifikasi dan total angkatannya tidak dihitung. Ini disebut sebagai “keluar dari perlombaan”. Saya hanya keluar dari perlombaan dua kali dalam karier angkat beban saya. Dalam kedua kasus tersebut, percobaan pertama saya terlalu berat. Dalam satu kasus, penyebabnya adalah deadlift saya. Saya sangat lelah setelah squat dan bench press sehingga saya tidak bisa mengangkat barbel dari lantai. Di waktu lain, saya tidak bisa mencapai kedalaman dalam upaya squat saya. Penilaiannya sulit, tetapi tidak adil. Saya seharusnya melakukan squat yang lebih ringan pada upaya squat pertama saya. Saya rasa bukan kebetulan bahwa kedua kali saya gagal, Jake tidak melatih saya. Aturan yang merugikan saya adalah bahwa setelah seorang lifter meminta beban, beban tersebut tidak dapat dikurangi.

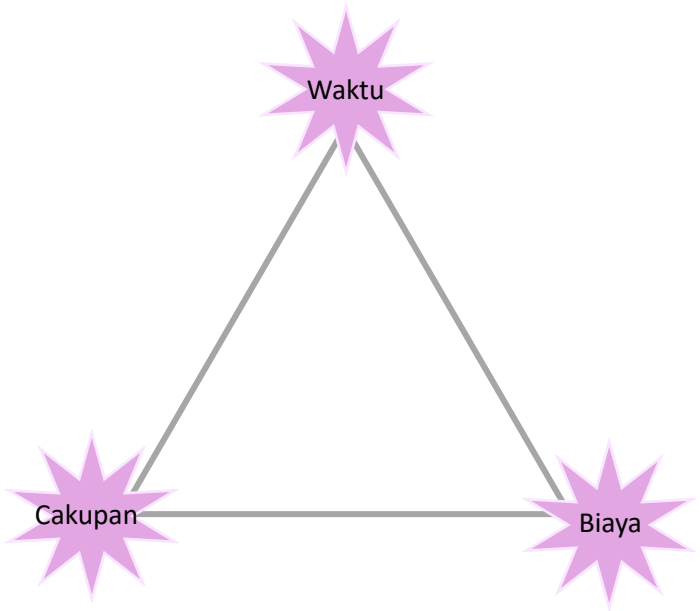
Dengan kata lain, jika seorang lifter meminta lima ratus pon pada upaya pertama mereka dan gagal mengangkat beban, ia hanya dapat mengulang lima ratus pon atau menaikkan beban. Jake akan terus mengatakan bahwa Anda harus memilih pembuka untuk tetap berada dalam pertandingan. Dengan kata lain, seorang lifter harus sangat konservatif dengan upaya pembukaan mereka pada setiap angkatan. Ia akan mendorong saya untuk memilih beban yang dapat saya lakukan dengan mudah untuk tiga kali repetisi. Dengan memilih beban yang ringan untuk pembuka, lifter memperhitungkan variabilitas. Peralatan yang digunakan dalam kontes berbeda dengan yang biasa digunakan oleh atlet angkat besi. Atlet angkat besi mungkin harus bepergian ke tempat pertandingan. Atlet angkat besi mungkin harus mengurangi berat badan untuk mengangkat beban di kelas berat tertentu.

Penilaiannya bisa ketat, atau bahkan tidak masuk akal. Terkadang mengangkat beban di depan banyak orang dapat menimbulkan kecemasan. Semua hal ini dapat memengaruhi performa atlet angkat besi. Saya akan memberikan contoh. Setiap kali juri tertentu (yang tidak akan disebutkan namanya) sedang bertugas di suatu pertandingan, saya tahu saya tidak akan mendapatkan jeda yang cukup saat melakukan bench press. Aturan angkat besi menyatakan bahwa dalam bench press, barbel harus diletakkan di dada atlet angkat besi hingga berhenti. Hal ini dilakukan agar atlet angkat besi tidak memantulkan barbel dari dadanya dan menggunakan momentum untuk menyelesaikan angkatan. Juri utama akan mengawasi turunnya barbel, dan memberikan perintah lisan "Tekan!" untuk memberi isyarat kepada atlet angkat besi untuk menyelesaikan angkatan.

Itulah aturannya dan saya melatih bench press saya sesuai dengan aturan tersebut. Namun, seyakin matahari akan terbit di timur, saya akan menunggu tiga atau empat detik untuk perintah pers jika orang ini duduk di kursi juri utama. Aturan Jake adalah Anda membuka

dengan ringan, upaya kedua Anda adalah apa yang menurut Anda cukup dapat Anda angkat hari itu, dan upaya ketiga Anda adalah untuk upaya memecahkan rekor pribadi atau berat yang harus Anda menangkan.

Jika saya membuka dengan sesuatu yang lebih dari sekadar triple yang mudah (berat yang dapat saya angkat untuk tiga kali), jeda yang lama mungkin akan menyebabkan saya melewatkan upaya itu. Sejujurnya, setiap juri berbeda. Terkadang perintah datang begitu cepat sehingga mengejutkan saya. Terkadang saya ingin bertanya kepada juri apakah dia tertidur. Itu adalah contoh yang baik tentang bagaimana variabilitas dapat berdampak negatif bahkan pada rencana yang paling matang. Seorang pengangkat beban biasanya membutuhkan waktu sepuluh hingga dua belas minggu untuk mempersiapkan diri menghadapi pertandingan. Itu adalah usaha yang cukup sia-sia jika Anda gagal. Saat berbicara tentang rencana pelepasan, lihat segitiga besi seperti yang ditunjukkan pada Gambar 9.1.



Gambar 9.1 Segitiga Besi Terdiri Dari Waktu, Biaya, Dan Ruang Lingkup

Segitiga besi adalah diagram yang menunjukkan tiga faktor yang memengaruhi setiap proyekwaktu, biaya, dan ruang lingkup. Saat berbicara tentang proyek Agile, hanya tiga hal ini yang dapat Anda ubah. Misalnya, anggaplah Anda memiliki proyek yang sedang berjalan dan tiba-tiba Anda mengetahui bahwa proyek tersebut harus diselesaikan pada tanggal tertentu. Hanya ada dua opsi yang dapat Anda sesuaikan, yaitu biaya dan ruang lingkup. Bayangkan sebuah perahu, perahu besar yang berada di atas air. Salah satu perahu panjang Viking dengan dayung, atau trireme Yunani. Jadi, ada sekelompok orang yang duduk di atas perahu yang mengoperasikan dayung, dan mereka menentukan seberapa cepat atau lambat perahu akan melaju dengan seberapa cepat atau lambat mereka mendayung. Perahu tersebut juga memiliki muatan yang harus mencapai tujuan. Orang-orang yang duduk di dayung adalah tim Scrum yang mengerjakan sebuah proyek.

Mereka mewakili faktor biaya dari segitiga besi. Orang-orang membutuhkan biaya.

Orang-orang yang mengoperasikan dayung adalah yang mendorong perahu. Anda tidak dapat meminta mereka untuk mendayung lebih cepat untuk mengurangi waktu yang dibutuhkan untuk mencapai tujuan Anda (atau menyelesaikan proyek). Hal yang sama berlaku untuk tim Scrum. Anda tidak dapat meminta mereka untuk membuat kode lebih cepat atau bekerja lebih keras. Ya, Anda bisa... mereka akan bekerja di akhir pekan dan bekerja dengan jam kerja yang tidak menentu untuk sementara waktu, tetapi mereka akan kelelahan pada akhirnya. Kelelahan itu tidak akan menyenangkan. Inilah yang kita lihat di Waterfall. Tim akan melakukan apa pun untuk menyelesaikan rilis pada tanggal GA yang dijanjikan, dan orang-orang merasa tidak senang… Ini bukanlah yang kami inginkan untuk tim kami. Jika Anda ingin perahu mencapai tujuannya lebih cepat, Anda hanya memiliki dua pilihan.

Anda dapat menambahkan lebih banyak dayung dan lebih banyak orang untuk mendayung. “Mesin” yang lebih besar ini akan mendorong perahu lebih cepat. Pilihan kedua adalah membuang sebagian muatan dari perahu dan membuatnya lebih ringan (mengurangi cakupan). Dengan muatan yang lebih ringan, kru dapat mendorong perahu lebih cepat tanpa harus bekerja lebih keras. Setiap proyek memiliki tiga factor cakupan, biaya, dan waktu. Hal ini tercermin dalam prinsip Agile kedelapan “Proses Agile mendorong pembangunan berkelanjutan. Sponsor, pengembang, dan pengguna harus dapat mempertahankan kecepatan yang konstan tanpa batas waktu.” Ketika Anda mencoba menentukan tingkat cakupan yang tepat, biaya yang tepat, dan waktu yang tepat pada suatu proyek, Anda menjamin kegagalan karena tidak ada fleksibilitas jika terjadi perubahan. Anda tidak memperhitungkan variabilitas atau mengharapkan hal-hal berubah.

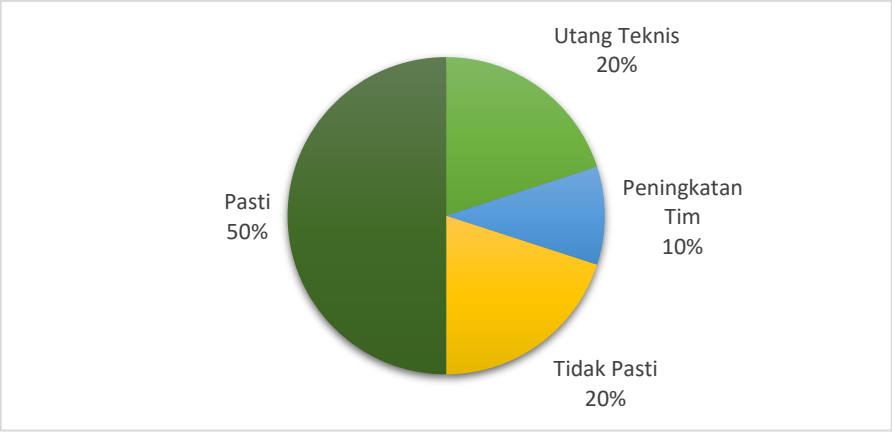
Salah satu alasan kami melakukan Scrum adalah karena kami menyadari bahwa mustahil merencanakan semuanya di awal proyek. Perhatikan bahwa kualitas tidak dapat diubah. Begitu pula Definisi Selesai. Sama seperti memilih upaya pembukaan dalam angkat beban, jangan buat rencana rilis Agile Anda terlalu kaku sehingga segala jenis perubahan atau variabilitas akan mengganggu rilis. Saat membangun rencana rilis, mulailah dengan epik. Epik adalah kelompok cerita yang memiliki tema umum dan memberikan fitur atau fitur yang diinginkan pelanggan. Misalnya, jika saya sedang membangun situs web komersial, epik mungkin menerima pembayaran. Yang lain mungkin keranjang belanja, atau menjelajahi produk. Pemilik Produk dan tim akan memberikan epik ukuran yang sangat kasar. Saya menyebutnya SWAG (Tebakan Liar Ilmiah).

Biasanya, SWAG akan cukup besar, dan harus mencerminkan ukuran relatif epik. Sama seperti poin cerita, tetapi dalam skala yang jauh lebih besar. Epik terbagi menjadi dua kelompok: pasti dan tidak pasti. Epik pasti adalah sesuatu yang kami rencanakan untuk disampaikan dalam rilis. Kami mensosialisasikan epik pasti dengan pelanggan dan pemangku kepentingan. Inilah hal-hal yang kami ingin mereka sukai sehingga mereka bermitra dengan kami dan terlibat dalam proses pengembangan. Saat menyiapkan rencana rilis, epik pasti harus mencakup 50 persen dari total pekerjaan yang ditetapkan dalam rilis. Tim, Pemilik Produk, dan bisnis harus menyetujui seberapa banyak pekerjaan atau kapasitas yang dapat ditangani tim untuk rilis mendatang.

Faktor-faktor seperti kecepatan tim rata-rata, lamanya rilis, dan iklim bisnis umum

harus dipertimbangkan. Biasanya, biaya juga merupakan faktor besar. Anda ingin menyeimbangkan biaya dan laba atas investasi (ROI) saat mencoba mencari tahu cara membangun rencana rilis. Epik tidak pasti harus mencakup sekitar 20 persen dari kapasitas rilis. Ini adalah epik dan cerita yang kemungkinan akan dilakukan, tetapi tidak dijanjikan. Ini adalah salah satu cara tim merencanakan perubahan. Pelanggan dan pemangku kepentingan tidak diberi tahu tentang pekerjaan ini. Pekerjaan yang tidak pasti dapat diselesaikan, atau mungkin tidak. Pada dasarnya, pekerjaan tersebut dapat dikorbankan.

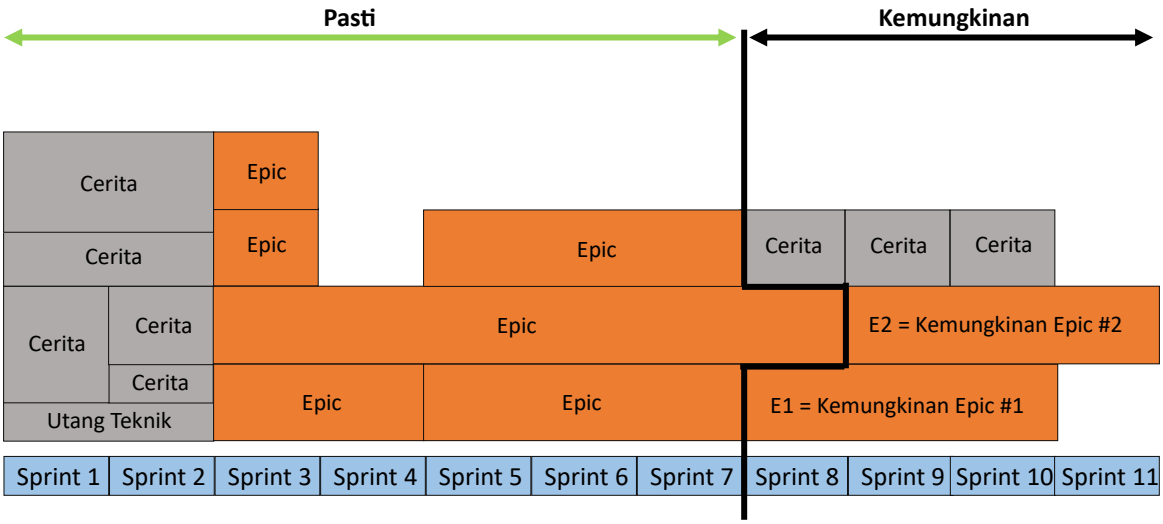
Pekerjaan tersebut memang bagus, tetapi tidak kritis. Pekerjaan yang pasti harus diselesaikan sebelum pekerjaan yang tidak pasti dipertimbangkan. Jadi, cakupan rilis harus sekitar 70 persen dari kapasitas rilis. Bagaimana dengan 30 persen lainnya? 30 persen lainnya digunakan untuk mengatasi utang teknis dan peningkatan tim. Ingatlah bahwa transparansi adalah salah satu dari tiga pilar Scrum. Kami ingin tim dapat meningkatkan diri dengan mempelajari bahasa pemrograman baru atau mempelajari sistem operasi atau teknik pengembangan yang akan datang. Kami juga tidak ingin mengabaikan utang teknis atau pemfaktoran ulang kode yang ada.



Gambar 9.2 Epik, Utang, Dan Peningkatan Tim

Alih-alih menyembunyikannya atau mengharapkan tim untuk mencari tahu cara menyesuaikan pekerjaan ke dalam jadwal yang sudah maksimal, kami memasukkannya langsung ke dalam rencana rilis. Teman saya dan sesama pelatih Agile Jay Cohen suka melihat 10 persen disisihkan untuk peningkatan tim dan 20 persen untuk utang teknis. Setiap tim berbeda, tetapi saya ingin melihat setidaknya 30 persen kapasitas rilis difokuskan pada sesuatu selain rilis itu sendiri sehingga tim memiliki kemampuan untuk mengatasi kekurangan secara realistis.

Gambar 9.2 menunjukkan rinciannya. Pada titik ini, mulailah membangun garis waktu. Uraikan epik dengan prioritas tertinggi menjadi cerita dan isi dua atau tiga Sprint pertama, seperti yang ditunjukkan pada Gambar 9.3. Gunakan kecepatan rata-rata tim sebelumnya untuk mengetahui berapa banyak pekerjaan yang harus dijadwalkan untuk Sprint. Jika Anda tidak mengetahui kecepatan rata-rata tim, buatlah tebakan terbaik Anda. Scrum Master akan menghitung ulang kecepatan rata-rata setelah tiga Sprint pertama dan setiap Sprint setelahnya, sehingga angka tersebut mungkin akan berubah.



Gambar 9.3 Garis Waktu Rencana Rilis

Setelah tiga atau lebih Sprint pertama terisi, cobalah untuk memasukkan sisa cerita epik yang pasti ke dalam sisa Sprint. Dengan hanya mengisi 50 persen rilis dengan pekerjaan yang pasti, tim memiliki "ruang gerak." Kami hanya ingin memiliki dua atau tiga Sprint yang diisi dengan cerita pada setiap titik selama pekerjaan rilis. Melangkah lebih jauh ke masa depan adalah pemborosan waktu dan pekerjaan. Kami memperkirakan banyak hal akan berubah karena umpan balik pelanggan dan perubahan persyaratan. Jika umpan balik pelanggan mengubah cakupan salah satu cerita epik yang pasti, kami memiliki fleksibilitas untuk melakukan pekerjaan itu, memuaskan pelanggan, dan tetap memberikan semua cerita epik yang pasti, dan hal yang sama jika perubahan persyaratan memaksa kami untuk menambahkan sesuatu ke rilis.

9.4 MEMPRIORITASKAN ITEM DAN PERENCANAAN SPRINT

Saya terus mengatakan bahwa Pemilik Produk harus secara rutin memprioritaskan Backlog. Bagaimana dia melakukannya? Ada banyak cara untuk melakukan ini. Yang ingin saya lakukan adalah melatih Pemilik Produk untuk memikirkan tiga hal: nilai bisnis, kompleksitas, dan biaya menunggu. Saya melatih Pemilik Produk untuk memprioritaskan cerita yang memberikan ROI terbesar. Seimbangkan apa yang dapat Anda selesaikan dengan cepat dengan apa yang diinginkan pelanggan dengan cepat. Pertimbangkan biaya penundaan. Anda jelas tidak dapat melakukan semuanya sekarang. Apa yang dapat ditunda dan tidak berdampak buruk pada produk? Apa yang perlu dilakukan sekarang untuk memastikan bahwa pelanggan senang dan mempertahankan bisnis? Jangan lupakan kompleksitas. Sebuah cerita mungkin berada di urutan teratas daftar prioritas, tetapi sangat rumit. Apakah masuk akal untuk mengerjakan cerita yang kurang rumit yang juga berprioritas tinggi tetapi akan membutuhkan waktu lebih sedikit untuk diselesaikan? Ingat, ini tentang memaksimalkan ROI.

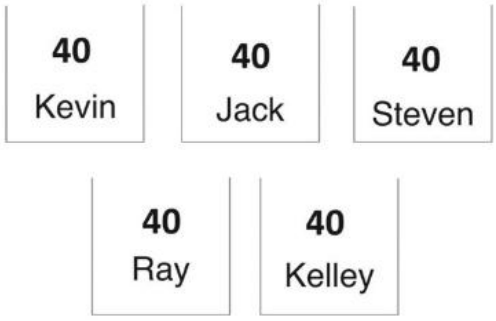
Menjalankan Rapat Perencanaan Sprint

Rapat Perencanaan Sprint adalah tempat cerita dibawa ke Sprint, diurai menjadi tugas,

dan ditugaskan kepada anggota tim. Tim setuju untuk menyelesaikan sejumlah cerita dalam rapat dan kemudian merencanakan cara menyampaikannya. Saya suka merencanakan kapasitas tim dalam "bucket". Ambil setiap anggota tim dan tentukan ketersediaan mereka untuk Sprint. Ingat, kami menggunakan hari-hari ideal. Saya suka lima jam per hari untuk setiap anggota tim. Jangan lupakan hal-hal seperti hari libur dan liburan. Sekarang Anda punya angka. Untuk Sprint dua minggu tanpa waktu libur, setiap anggota tim harus memiliki sekitar lima puluh jam dalam jadwal masing-masing, seperti yang terlihat pada Gambar 9.4.

Gambar 9.4 Bucket individual

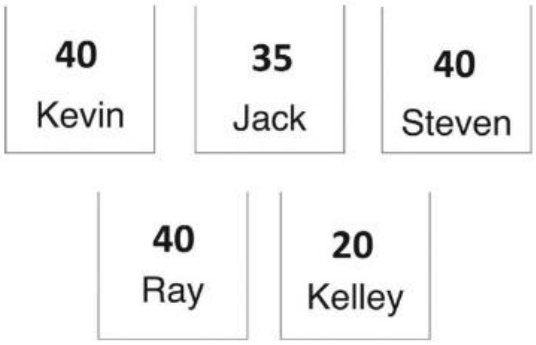
Berikutnya, saya mengurangi bucket setiap anggota tim sebanyak sepuluh jam untuk rapat dan upacara Agile seperti yang terlihat pada Gambar 9.5. Anda memiliki Perencanaan Sprint (rapat yang sedang Anda jalankan), Penyempurnaan Backlog, Tinjauan Sprint, Retrospektif, rapat tim lainnya, dan hal-hal seperti rapat balai kota dan semacamnya. Semua rapat tersebut sebenarnya tidak mencapai sepuluh jam. Saya juga ingin sedikit waktu tambahan.



Gambar 9.5 Bucket Yang Dikurangi

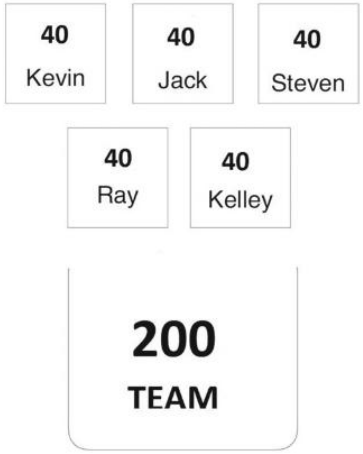
Mengapa?

Saya merencanakan variabilitas. Saya menyediakan ruang untuk mengubah rencana jika perlu. Jika Anda memenuhi Sprint secara berlebihan, perubahan apa pun akan membuat tim tidak dapat menyelesaikan semua pekerjaan dalam Sprint. Jika tim dukungan memerlukan bantuan untuk masalah berprioritas tinggi atau anggota tim jatuh sakit, tim memiliki ruang untuk menyesuaikan dan tetap menyelesaikan pekerjaan Sprint. Tentu saja, jika anggota tim tidak dapat bekerja setiap hari dalam Sprint karena liburan atau pameran dagang, atau semacamnya, waktu mereka akan berkurang seperti yang ditunjukkan pada Gambar 9.6.



Gambar 9.6 Kelly Dan Jack Mengambil Cuti Pada Sprint Ini

Sekarang Anda telah mengetahui kapasitas tim. Untuk tim Scrum yang beranggotakan lima orang, total kapasitas tim adalah 200 jam untuk Sprint dua minggu seperti yang terlihat pada Gambar 9.7.



Gambar 9.7 Bucket Tim

Di awal rapat, Pemilik Produk harus menetapkan sasaran Sprint. Kemudian tim memilih cerita dengan prioritas tertinggi dan menguraikannya menjadi tugas-tugas yang diperkirakan.

Menguraikan Cerita dan Tugas

Dalam rapat Perencanaan Sprint, tim mengambil cerita dan menguraikannya menjadi tugas-tugas. Bersikaplah konservatif dengan perkiraan. Tidak ada yang lebih penting daripada menjaga perkiraan tugas tetap wajar. Ingatlah bahwa komitmen Sprint harus dipatuhi. Jika cerita terlalu besar, bagilah. Jangan takut untuk mengembalikan cerita ke Backlog untuk penyempurnaan lebih lanjut jika perlu. Saat tim bekerja menguraikan cerita menjadi tugas-tugas, latih mereka untuk menyempurnakan tugas-tugas tersebut sekecil mungkin. Seperti yang saya katakan sebelumnya, jangan hanya membuat tugas untuk pengembangan, tugas untuk pengujian, dan tugas untuk dokumentasi. Cobalah untuk menguraikan tugas-tugas menjadi langkah-langkah pengembangan.

Ini akan membantu para penguji dan staf dokumentasi terlibat dalam proses lebih

awal. Tentu saja, jika tim melakukan pengembangan berbasis pengujian, itu bukan masalah. Sebagian besar tim tidak demikian, jadi mereka menderita apa yang saya sebut sebagai "hockey stick Scrum," Di sinilah para penguji diharapkan melakukan semua pengujian selama beberapa hari terakhir Sprint. Hockey stick menyebabkan stres pada para penguji dan cerita tidak selesai dalam Sprint. Jangan menendang lebih jauh dari jangkauan Anda. Pernahkah Anda melihat ini? Anda sedang menonton pertandingan sepak bola dan si punter melakukan tendangan yang sangat jauh. Anda akan berpikir bahwa ini adalah hal yang baik; lagipula, Anda ingin mendukung serangan tim lain sebanyak mungkin. Namun, yang biasanya terjadi adalah tendangan tersebut dikembalikan untuk sebuah touchdown. Mengapa? Karena tim jangkauan tidak dapat turun ke lapangan dengan cukup cepat.

Pengembali dapat mundur, menangkap bola, dan memiliki banyak waktu untuk mengidentifikasi blok dan memanfaatkan celah dalam jangkauan tendangan. Hal yang sama dapat terjadi pada tim Scrum. Pengembang ingin menulis kode. Saya mengerti itu, tetapi ada lebih banyak hal yang harus dilakukan dengan sebuah cerita daripada sekadar melengkapi kode. Hal-hal perlu diuji dan didokumentasikan. Scrum dengan tongkat hoki menyebabkan para penguji kewalahan dengan pekerjaan di akhir Sprint sehingga cerita tidak selesai. Intinya, kami mengungguli liputan kami. Masalah besar di sini adalah bagaimana pengembang dan penguji bekerja di bawah Waterfall, yang mengharuskan Anda menguji di akhir proyek, dan Anda menulis kasus pengujian dari dokumen spesifikasi desain.

Mencoba bekerja dengan cara ini dalam Sprint sama sekali tidak berjalan dengan baik. Cara mengatasinya adalah dengan melibatkan penguji selama proses pengembangan. Tim perlu melakukan apa pun untuk menyelesaikan pengujian sedini mungkin. Kasus pengujian perlu berkembang saat cerita sedang dikerjakan, dan mungkin mengharuskan penguji diberi hal untuk diuji saat cerita berkembang—bukan saat kodenya selesai. Ini memerlukan kolaborasi yang erat antara penguji dan pengembang. Mereka harus (aduh!) benar-benar berbicara. Bagian kerennya adalah bahwa pengembangan menjadi sangat menyadari upaya pengujian, sehingga mereka dapat memberikan umpan balik yang bagus tentang penulisan pengujian yang efektif.

Selain itu, pengujian "akhir" pada cerita menjadi jauh lebih mudah karena penguji sangat familier dengan cerita tersebut. Terakhir, tim menjadi jauh lebih baik dalam mengukur cerita, karena pemahaman yang lebih mendalam tentang upaya yang diperlukan untuk menyelesaikannya. Ya, saya tahu bahwa nirwana bagi tim Scrum adalah tidak adanya peran yang ditetapkan... Anda tahu, tidak ada lagi pengembang, QA, atau apa pun. Semua orang membuat kode, semua orang menguji, apa pun yang diperlukan. Sebagai sebuah tim, jangan menempatkan diri Anda pada posisi di mana Anda tidak dapat memberikan apa yang telah Anda janjikan. Selalu lebih baik untuk tidak menjanjikan banyak hal dan memberikan lebih banyak hal.

9.5 MENYEMPURNAKAN ESTIMASI

Tim Scrum tidak boleh mengubah ukuran cerita. Oke, mungkin tidak pernah adalah kata yang terlalu kuat. Tim jarang boleh mengubah ukuran cerita. Menentukan ukuran cerita

adalah aktivitas tim dan sifatnya relatif. Jika cerita ini adalah tiga, seberapa besar cerita lainnya ini dalam kaitannya dengan yang pertama? Saat cerita didefinisikan, tim lebih memahami apa yang diminta oleh Product Owner untuk disampaikan dan pekerjaan yang dibutuhkan. Dalam kasus ini, ukurannya dapat berubah karena tim lebih memahami pekerjaan tersebut. Namun, itu bukanlah sesuatu yang sering terjadi. Skenario yang lebih sering terjadi adalah ketika tim mengukur cerita secara tidak tepat. Biasanya, ukurannya terlalu kecil. Tim mulai mengerjakan cerita dan menemukan bahwa ada lebih banyak pekerjaan yang harus dilakukan daripada yang awalnya diperkirakan.

Misalnya, cerita yang awalnya dianggap tim sebagai tiga ternyata menjadi tiga belas. Dalam kasus ini, tim tergoda untuk kembali dan mengubah ukuran semua cerita lainnya di Backlog. Di permukaan, ini tampak seperti hal yang benar untuk dilakukan. Tidak. Poin cerita digunakan untuk menghitung kecepatan tim. Hal yang lucu tentang kecepatan adalah bahwa kecepatan itu akan muncul dengan sendirinya. Saya suka mengatakan bahwa semuanya akan terlihat setelahnya. Bahkan jika Anda terlalu membesar-besarkan atau mengecilkan cerita, kecepatan rata-rata akan tetap akurat. Jika cerita yang Anda kira tiga ternyata tiga belas, kecepatan rata-rata tim akan sedikit menurun tetapi akan kembali normal seiring waktu. Bahkan jika semua cerita yang dikira tim adalah tiga ternyata tiga belas, kecepatan akan tetap secara akurat memprediksi kapan tim akan menyelesaikan Backlog.

Pengecualian mungkin terjadi jika tim menemukan bahwa cerita perlu dipecah. Estimasi baru perlu diberikan untuk cerita yang dihasilkan. Idealnya, ini akan terjadi sebelum Perencanaan Sprint, tetapi mungkin terjadi dalam Perencanaan Sprint. Apa pun itu, begitu Anda mulai mengerjakan cerita, jangan menoleh ke belakang! Mengukur dan memperkirakan adalah dua hal yang berbeda. Ketika tim menguraikan cerita menjadi tugas, setiap tugas perlu diperkirakan. Tim mencapai kesepakatan tentang berapa banyak waktu yang dibutuhkan untuk menyelesaikan setiap tugas. Ini disebut estimasi. Setiap tim memperkirakan tugas secara berbeda. Saya telah melihat tim mencapai konsensus tentang berapa lama waktu yang dibutuhkan untuk menyelesaikan suatu tugas, dan saya telah melihat masing-masing anggota tim mengukur tugas mereka sendiri.

Kedua pendekatan tersebut dapat diterima di mata saya. Kuncinya adalah semua tugas diestimasikan. Jika seorang anggota tim mengerjakan tugas yang awalnya diperkirakan memakan waktu 6 jam tetapi ternyata sangat kurang dari perkiraan, tambah jam tugas tersebut. Jika awalnya diperkirakan memakan waktu 6 jam, tetapi sekarang membutuhkan waktu 20 jam untuk menyelesaikannya, ubahlah. Bagan burn-down Sprint digunakan untuk mengukur semua pekerjaan yang tersisa dalam Sprint dan seberapa cepat tim menyelesaikannya. Tim tidak hanya perlu "membakar" jam tugas saat mereka menyelesaikannya, tetapi juga memberikan gambaran realistis tentang berapa banyak pekerjaan yang tersisa untuk menyelesaikan tugas tersebut. Tidak seperti poin cerita, estimasi burn-down harus seakurat mungkin.

Jika estimasi tidak tepat, ubahlah sehingga burn-down Sprint menjadi akurat. Jika ada masalah, tim, Scrum Master, dan Product Owner ingin dapat melihatnya dengan baik sebelumnya. Alasan kami ingin melihat segala sesuatunya dengan jelas adalah untuk

mengambil tindakan sebelum masalah muncul. Jika burn-down menunjukkan bahwa tim tidak akan dapat menyelesaikan semua tugas sebelum akhir Sprint, Product Owner dan Scrum Master harus mengambil tindakan.

Mungkin Product Owner dapat mengurangi cakupan dengan memindahkan cerita keluar dari Sprint. Scrum Master harus mencari tahu apa yang terjadi. Apakah itu benar-benar hanya perkiraan yang salah, atau ada hal lain yang perlu diselidiki? Ini tidak seperti rapat status Waterfall lama di mana manajer mencari alasan untuk menghukum orang karena tertinggal. Scrum Master, Product Owner, dan manajer harus mencari cara untuk membantu tim mencapai tujuan mereka. Tim perlu bekerja dengan kecepatan yang dapat dipertahankan.

Menangani Cacat yang Ditemukan dalam Sprint

Sebuah tim ingin menghormati komitmen Sprint-nya. Apa yang terjadi ketika penguji menemukan cacat? Lebih buruk lagi, apa yang terjadi ketika cacat ditemukan di akhir iterasi? Anda memperbaikinya. Saya melatih tim bahwa pekerjaan Sprint dengan prioritas tertinggi adalah memperbaiki cacat. Terlalu sering, saya melihat cacat diabaikan. Ini adalah proses yang saya sebut membangun gunung cacat. Pada akhirnya, cacat ini harus ditangani, dan itu akan menyakitkan. Saya lebih suka melihat tim tidak menyampaikan semua cerita yang dijanjikan dalam Sprint sehingga cacat dapat diperbaiki. Itulah pentingnya. Aturannya adalah Anda memperbaiki cacat dalam iterasi tempat cacat tersebut ditemukan. Jika cerita perlu dihapus dari Sprint dan dimasukkan kembali ke Backlog untuk memperhitungkan waktu yang dibutuhkan untuk memperbaiki cacat, biarlah.

Terkadang ada baiknya melihat situasi dari perspektif yang berbeda untuk memahami dampaknya sepenuhnya. Istri saya membelikan saya pelacak kebugaran untuk ulang tahun saya. Anda tidak melihat banyak pria berusia 80 tahun dengan berat 300 pon berjalan-jalan, jadi fokus saya beralih dari mengangkat beban berat menjadi bergerak. Pelacak kebugaran adalah gadget yang memberi tahu saya hal-hal seperti berapa banyak langkah yang saya ambil dalam sehari, kalori yang terbakar, dan berapa mil yang telah saya tempuh. Saya mendapati diri saya terus-menerus memeriksa benda itu dan berusaha keras untuk mencapai target langkah harian saya. Kemudian, alat itu berhenti bekerja. Mengatakan bahwa saya tidak senang adalah pernyataan yang meremehkan. Barang itu baru berusia empat bulan. Saya menghubungi perusahaan itu dan mereka mengganti perangkat itu.

Akan tetapi, saya tidak pernah mendapatkan kembali antusiasme awal saya untuk menggunakannya. Bahkan, saya baru-baru ini menggantinya dengan merek lain. Tanpa cacat berarti seperti itu. Tanpa cacat. Pelanggan kami mengharapkan apa yang kami buat memberi mereka nilai yang mereka harapkan dan, ya berfungsi. Kualitas lebih dari sekadar slogan. Ini adalah komitmen kepada pelanggan kami untuk melakukan yang terbaik dalam menghadirkan perangkat lunak yang berfungsi. Dalam bab ini, kita membahas beberapa keterampilan teknis yang dibutuhkan oleh Scrum Master, seperti mengotomatiskan pembuatan dan menyusun rencana rilis. Dalam bab berikutnya, kita akan membahas beberapa keterampilan darurat yang mungkin dibutuhkan oleh Scrum Master.

BAB 10
KETERAMPILAN KONTINJENSI SCRUM MASTER

Saya tumbuh di tahun tujuh puluhan, saat kita tidak menggunakan sabuk pengamankita hampir tidak menggunakan kursi mobil. Berkendara di belakang truk pikap tidak dianggap sebagai aktivitas yang berbahaya. Salah satu hal yang paling saya rindukan dari tahun tujuh puluhan adalah program televisi. Mungkin karena kita hanya punya tiga saluran (empat jika Anda jago menggunakan aluminium foil). Salah satu hal yang saya ingat dari program tersebut adalah pasir hisap. Saya pribadi tidak pernah mengalami pasir hisap, tetapi itu ada di mana-mana di televisi. Anda akan mengurus urusan Anda sendiri, dan hal berikutnya yang Anda tahu Anda berada di pasir hisap.

Saat saya mengingat banyak hal, semakin Anda berjuang, semakin cepat Anda tenggelam. Anda harus meminta bantuan. Seseorang harus menarik Anda keluar. Selain nostalgia, saya pikir hambatan proyek seperti pasir hisap. Ketika saya mengucapkan kata "penghalang", saya yakin Anda berpikir tentang batu raksasa di tengah jalan. Terkadang hambatan lebih seperti pasir hisap. Anda berpikir, "Saya bisa mengatasinya," tetapi semakin Anda berjuang, semakin Anda terpuruk. Dapatkan bantuan sebelum terlambat.

10.1 MENGIDENTIFIKASI DAN MENGHILANGKAN HAMBATAN

Saat hambatan muncul, hal pertama yang saya tanyakan adalah: "Apakah ini benar-benar hambatan atau sesuatu yang dapat ditangani oleh tim?" Jika seorang anggota tim sakit dan akan absen selama sisa Sprint, dapatkah tim melanjutkan pekerjaannya? Dapatkah tim mengatur ulang dan tetap mencapai tujuan Sprint? Sekarang, jika anggota tim tersebut memenangkan lotre besar dan membeli pulau pribadi di suatu tempat, itu adalah hambatan. Kita perlu mengganti orang tersebut dalam tim atau merencanakan pengurangan kecepatan. Sesuatu menjadi hambatan saat melampaui kemampuan tim. Saat menangani hambatan, penting untuk memahami cakupannya.

Dengan kata lain, siapa yang berwenang untuk menanganinya? Sebagai Scrum Master, Anda tidak memiliki banyak wewenang. Pahami siapa yang dapat membantu Anda mengatasi masalah tersebut dan libatkan mereka sesegera mungkin. Setelah hambatan diidentifikasi, saya suka mencantumkannya di papan hambatan saya. Papan hambatan adalah radiator informasi yang diletakkan di tempat yang dapat dilihat semua orang. Dengan cara ini, tidak ada pertanyaan tentang kondisi hambatan yang sedang ditangani oleh Scrum Master. Scrum Master juga dapat membantu mereka mengatasi hambatan tersebut. Saya juga melatih tim saya untuk tidak menunggu rapat rutin harian jika hambatan tersebut mendesak. Jangan biarkan protokol membuat Anda terpuruk. Libatkan Scrum Master sesegera mungkin.

Mengubah Komposisi dan Personel Tim

Singkatnya, tidak…. Tim Scrum adalah tempat terjadinya keajaiban. Saya pernah mengatakan sebelumnya bahwa tidak ada yang tidak dapat dilakukan oleh tim Scrum. Bagi saya, tim adalah landasan Scrum. Saya tidak suka mengacaukan tim. Menambah atau

mengurangi anggota akan mengganggu dan harus dihindari jika memungkinkan. Daripada mengganggu tim, alihkan pekerjaan ke tim. Jika pekerjaan tersebut cukup penting untuk memecah tim, pertimbangkan untuk meminta seluruh tim melakukannya. Memindahkan pekerjaan ke tim memungkinkan tim untuk tetap bersatu dan bisnis untuk memenuhi kebutuhannya.

10.2 MEMPERSIAPKAN DAN MENJALANKAN TINJAUAN SPRINT

Tinjauan Sprint adalah acaranya Pemilik Produk. Ya, tim dapat memamerkan semua hal keren yang mereka buat, tetapi PO membangun hubungannya dengan pelanggan. Mereka dapat menyoroti bahwa nilai pelanggan sedang direalisasikan. Saya suka memfasilitasi rapat perencanaan Sprint untuk PO. Hal ini memungkinkan PO untuk fokus pada pelanggan dan pemangku kepentingan dan tidak perlu khawatir tentang biaya operasional rapat. Saya ingin PO tahu bahwa saya mendukung mereka. Ini adalah salah satu cara saya melakukannya untuk mereka. Titik fokus rapat adalah demonstrasi dari apa yang telah dibangun, jadi pastikan Anda telah menyiapkannya terlebih dahulu. Pastikan Anda tahu anggota tim mana yang akan mendemonstrasikan fungsionalitas apa. Tidak, saya tidak melakukan demo. Saya tidak memiliki keahlian teknis dan mungkin tidak akan memberikan demonstrasi fungsionalitas yang sangat baik. Ditambah lagi anggota tim dapat "memamerkan" sedikit.

Pastikan demo yang direncanakan berfungsi dengan baik, dan tidak menunjukkan apa pun yang bersifat rahasia atau khusus perusahaan. Pastikan Anda tidak memamerkan lebih dari yang Anda inginkan. Karena itu, jangan takut untuk memamerkan proses "pembuatan sosis". Pelanggan dan pemangku kepentingan ini adalah bagian dari proses pengembangan. Saya tidak berharap demo-demo tersebut akan dipoles seperti sesuatu yang akan ditunjukkan perusahaan di pameran dagang. Selanjutnya, baik PO atau saya membuat slide deck kecil. Apa!?! Tuan "Saya benci PowerPoint dan berharap kita membatalkan lisensi perusahaan" ingin membuat slide deck? Yang saya benci dari cara orang-orang membuat slide deck untuk tinjauan Sprint adalah mereka berlebihan.

Yang ingin saya tunjukkan adalah di mana kita berada dalam rencana rilis cerita apa yang telah dibuat dan hal-hal apa yang akan kita kerjakan di iterasi berikutnya. Saya mungkin akan memasukkan slide lain yang mencantumkan siapa saja yang ada di tim Scrum dan peran apa yang mereka isi. Itu saja. Untuk rapat itu sendiri, saya membuka panggilan dan menyapa pelanggan melalui telepon serta siapa pun di ruangan itu. Ya, di ruangan itu. Anda ingin mendorong pelanggan, pemangku kepentingan, anggota tim, manajemen, dan siapa pun yang dapat Anda pikirkan untuk menghadiri rapat secara langsung.

Secara realistis, sebagian besar pelanggan tidak akan berada di sekitar Anda sehingga Anda perlu menggunakan teknologi untuk menunjukkan kepada mereka apa yang telah dibuat. Saya biasanya melakukan absensi pelanggan sehingga kami mengetahui siapa yang datang, dan memperkenalkan tim. PO kemudian menjelaskan di mana kami berada dalam rencana rilis dan memperkenalkan demo. Setelah setiap demo, mintalah umpan balik. Di akhir rapat, saya memastikan untuk mengucapkan terima kasih kepada semua orang karena telah datang ke rapat dan memberi tahu mereka bahwa jika mereka memiliki pertanyaan atau

umpan balik tambahan, mereka dapat menghubungi PO atau Scrum Master.

Memfasilitasi Rapat Virtual

Sudah jelas bahwa kami lebih mementingkan komunikasi tatap muka daripada hal lainnya. Itu tidak selalu memungkinkan, jadi bagaimana Anda memastikan bahwa rapat virtual bermanfaat bagi semua pihak yang terlibat? Di sinilah keterampilan fasilitasi Scrum Master harus tepat sasaran. Dorong komunikasi yang berlebihan setiap saat. Saya sering bertanya pada diri sendiri, "Apa masukan dari Anda yang menelepon??" Pastikan Anda memanfaatkan setiap kesempatan untuk melibatkan semua orang (baik yang lokal maupun yang jauh) dalam percakapan. Jika memungkinkan, gunakan webcam sehingga Anda dapat melihat orang-orang yang jauh dan mereka dapat melihat Anda. Tidak berada di ruangan yang sama dapat menjadi keterbatasan; namun, itu dapat diatasi dengan fasilitasi yang tepat.

Mengkoordinasikan Pekerjaan Satu Backlog Produk dengan Beberapa Tim

Terus terang, mengoordinasikan beberapa tim dengan sukses pada satu Backlog produk bergantung pada komitmen dan pengelolaan Pekerjaan yang Sedang Dikerjakan (WIP). Jika Backlog diprioritaskan dengan tepat, tim seharusnya tidak memiliki masalah dalam memahami pekerjaan apa yang perlu dilakukan. Tim harus memiliki rencana Sprint bersama, dan mencapai kesepakatan mengenai cerita mana yang akan mereka masukkan ke dalam Sprint Backlog. Jika ada ketergantungan antar cerita yang mencakup tim, ketergantungan tersebut perlu diidentifikasi. Dalam kasus ini, tim harus membuat komitmen satu sama lain untuk memastikan ketergantungan tersebut terpenuhi.

Setelah semua tim berkomitmen pada pekerjaan di Sprint Backlog, mereka harus memberikan perhatian yang sangat cermat pada WIP dan mengelolanya. Sangat penting bagi setiap tim untuk menyelesaikan cerita berprioritas tinggi mereka. Rapat Scrum of Scrum, seperti yang dijelaskan dalam Bab 4, harus diadakan untuk memastikan transparansi antar tim. Scrum Master dari semua tim yang bekerja dari Backlog ini harus membantu memfasilitasi rapat Scrum of Scrum. Paling tidak, mereka harus memulainya.

Mendapatkan Bantuan

Seorang Scrum Master tidak harus tahu cara memperbaiki situasi apa pun. Mereka harus tahu siapa yang bisa. Saya suka mengatakan bahwa seorang Scrum Master tidak memiliki wewenang, karena pada kenyataannya mereka tidak memiliki wewenang. Ketahui siapa yang memiliki kewenangan yang tepat untuk menangani situasi saat ini dan jangan malu. Serius, Sama seperti tim Scrum yang tidak boleh membiarkan hambatan berkembang biak, seorang Scrum Master perlu menangani hambatan secepat mungkin. Jangan biarkan rasa takut atau kesombongan menghalangi. Mintalah bantuan saat Anda membutuhkannya. Singkirkan hambatan.

10.3 MENGATASI SCOPE CREEP

Scope creep terjadi saat proyek berkembang melampaui apa yang awalnya direncanakan. Biasanya akan muncul di Sprint burn-down atau Release Backlog. Awalnya cukup polos. Seorang pelanggan bertanya kepada PO apakah "sebagai bantuan" mereka dapat menambahkan sesuatu ke Backlog. Pengembang yang bermaksud baik menambahkan

beberapa tugas ke cerita karena ia merasa itu adalah hal yang benar untuk dilakukan. Terkadang scope creep berasal dari sesuatu yang eksternal bagi tim, seperti dukungan untuk fitur sistem operasi baru. Terkadang itu karena visi proyek tidak didefinisikan dengan baik atau cerita tidak disempurnakan dengan cukup. Dari mana pun asalnya, scope creep dapat menjadi pembunuh.

Waspadalah terhadap bagan burn-down yang datar atau perubahan radikal dalam burn-up rilis. Keduanya menunjukkan hal-hal yang ditambahkan ke rilis. Sebagai Scrum Master, bicaralah dengan PO dan coba lihat dari mana pekerjaan tambahan itu berasal. Ketika teridentifikasi, ambil langkah-langkah yang tepat untuk menangani pekerjaan tambahan tersebut. Jika itu karena masalah komunikasi, ambil langkah-langkah yang tepat untuk memastikan semua orang memahami scope proyek. Jika hal itu berasal dari pelanggan, pastikan prioritas disesuaikan dengan benar dan hal ini tidak akan menghambat tim untuk menyampaikan cerita yang pasti dalam rencana rilis.

Mengurangi Cakupan

Ada banyak alasan bagus untuk mengurangi cakupan proyek. Mengatasi pertambahan cakupan tentu saja salah satunya. Ingat segitiga besi. Jika waktu atau biaya pada proyek berubah, cakupan mungkin perlu disesuaikan. Mungkin kecepatan tim terbukti merepotkan. Saat mengurangi cakupan, cobalah untuk memastikan bahwa tim masih dapat menyampaikan cerita yang pasti dalam rencana rilis. Salah satu alasan kami tidak mensosialisasikan cerita yang tidak pasti dengan pelanggan adalah untuk memberi kami "ruang gerak" yang diperlukan saat situasi seperti ini muncul. Jika Anda diminta untuk memotong cerita yang pasti, hal ini perlu didiskusikan dengan pelanggan sesegera mungkin. Bersikaplah transparan dan beri tahu pelanggan dengan tepat bagaimana rilis akan berbeda dari yang awalnya dipikirkan.

Membatalkan Sprint

Ini bukanlah sesuatu yang terjadi secara teratur, tetapi ada kalanya perlu untuk membatalkan Sprint. Product Owner memiliki kewenangan untuk membatalkan iterasi. Mungkin ada keadaan eksternal yang tidak terduga yang menyebabkan tujuan Sprint tidak lagi masuk akal. Jika pekerjaan yang dilakukan dalam Sprint tidak lagi menghasilkan nilai, batalkan Sprint. Ketika ini terjadi, Scrum Master harus melakukan Retrospeksi tentang situasi tersebut. Apa yang menyebabkan keadaan menjadi buruk, dan bagaimana tim dapat memeriksa dan beradaptasi? PO akan memiliki beberapa pekerjaan prioritas ulang untuk dilakukan sehingga Backlog menjadi benar. Tim kemudian harus merencanakan Sprint lain dan mulai lagi. Penting untuk menjaga keselarasan dengan jadwal yang tercermin dalam rencana rilis, sehingga tim mungkin harus merencanakan Sprint yang lebih pendek dari biasanya atau yang lebih panjang dari biasanya.

Mendokumentasikan Keputusan

Semakin tua usia saya, semakin sedikit yang saya ingat. Setidaknya saya menyalahkan usia tua. Saat mendekati ulang tahun kelima puluh, saya mendapati diri saya melakukan hal-hal seperti masuk ke sebuah ruangan dan lupa mengapa saya masuk ke sana. Kunci mobil saya sepertinya tidak pernah berada di tempat saya meninggalkannya. Saat tim membuat keputusan, sebaiknya tuliskan di suatu tempat. Dengan demikian, keputusan tersebut dapat

dirujuk saat tidak ada yang ingat mengapa keputusan itu dibuat. Saya lebih suka mendokumentasikan keputusan tim di suatu tempat secara elektronik, misalnya di halaman wiki. Berikan akses kepada seluruh tim dan dokumentasikan semuanya di sana. Anda akan senang melakukannya.

Melaporkan Kinerja Tim

Manajer program ingin melacak kinerja tim sehingga ia dapat memastikan bahwa proyek tersebut berjalan dengan baik. Mereka juga ingin dapat berbagi informasi tersebut dengan sponsor. Informasi ini disebut metrik. Metrik memberikan informasi yang dapat diandalkan kepada para pengambil keputusan. Metrik mencerminkan kesehatan proyek dan upaya pengembangan. Metrik memberikan pengukuran tentang bagaimana keadaan berjalan, dan memberikan dasar untuk mengukur dampak dari setiap perbaikan yang mungkin telah diterapkan oleh tim. Ada beberapa metrik yang dapat digunakan untuk membantu pengambilan keputusan.

Sprint Burn-down

Sprint burn-down menunjukkan seberapa cepat tim menyelesaikan pekerjaan Sprint mereka. Ini memprediksi dengan andal apakah semua pekerjaan yang dikomit dalam iterasi saat ini dapat diselesaikan pada akhir iterasi. Sprint burn-down menunjukkan seberapa efisien tim bekerja, dan apakah ada masalah yang perlu ditangani.

Release Burn-up

Mirip dengan Sprint burn-down, release burn-up memprediksi kapan tim akan menyelesaikan semua cerita dalam Backlog rilis. Release burn-up menunjukkan kesehatan rilis secara keseluruhan. Ini dapat digunakan untuk menentukan apakah cakupan perlu dikurangi untuk merilis tepat waktu, atau fungsionalitas mungkin dapat ditambahkan.

Tren Cacat

Berapa banyak cacat yang saat ini terbuka? Berapa banyak yang telah ditutup sejauh ini dalam proyek? Cacat perlu ditanggapi dengan serius. Tren cacat dapat dianalisis untuk memastikan bahwa kebijakan tanpa cacat dipatuhi. Seperti yang saya katakan, ada banyak cara untuk mencerminkan kinerja tim. Temukan apa yang berharga bagi tim dan bisnis Anda dan bersikaplah transparan. Tidak seorang pun akan memberi Anda satu juta dolar dan berkata, "Sampai jumpa dalam enam bulan." Bisnis akan ingin memahami berapa laba atas investasinya, jadi mereka jelas ingin tahu tentang kesehatan proyek.

BAB 11
MENYATUKAN SEMUANYA

11.1 MEMBANGUN KEBERHASILAN DALAM SCRUM

Saya akan menjadi orang pertama yang mengakui bahwa ketika saya menulis, saya hanya mendokumentasikan apa yang terlintas dalam pikiran saya saat itu. Di akhir buku ini, saya ingin mengakhiri semuanya dan berbagi beberapa pemikiran saya tentang apa yang diperlukan untuk menjadi sukses di Scrum dan peran Scrum Master. Berdiri di depan tim Scrum bisa jadi menakutkan. Akan ada orang-orang yang bersemangat, skeptis, apatis, dan takut. Anda adalah orang yang diharapkan oleh tim untuk memimpin mereka ke dalam "hal-hal" Agile ini. Ini mengingatkan saya ketika saya diminta untuk melatih tim sepak bola U6 (di bawah 6 tahun) putri saya. Putri saya ingin bermain, jadi kami mendaftarkannya dan membelikannya satu set sepatu dan pelindung tulang kering kecil. Sebelum latihan seharusnya dimulai, telepon berdering. Orang di seberang telepon menjelaskan bahwa mereka benar-benar membutuhkan pelatih. Ketika saya mengatakan bahwa saya tidak tahu apa-apa tentang sepak bola, tanggapannya adalah, "Tidak apa-apa, Mereka masih anak-anak.

Minta saja mereka mengejar bola.”Karena saya orang yang mudah terpikat dengan cerita sedih, saya setuju untuk menjadi pelatih. Pada latihan pertama, saya membawa tas berisi bola sepak, kerucut, dan peluit. Saya berdiri di tengah lingkaran anak-anak berusia empat dan lima tahun yang menatap saya dengan mata besar seperti anak-anak. Saya ketakutan setengah mati. Saya tidak tahu harus berbuat apa. Hal yang sama terjadi pada saya saat berdiri di depan tim Scrum. Scrum memiliki konsep yang sangat sederhana. Praktiknya mudah, tetapi pelaksanaannya sangat sulit. Saya menyatakan minat untuk menjadi Scrum Master, dan tiba-tiba saya ditugaskan ke sebuah tim.

Dengan tim sepak bola, jelas bahwa saya tidak tahu apa yang saya lakukan, jadi saya belajar sendiri. Saya berbicara dengan pelatih lain, membaca buku dan materi daring, dan meminta bantuan salah satu teman saya yang benar-benar pernah bermain sepak bola. Anak-anak itu berubah dari berlarian seperti orang gila menjadi melakukan latihan dan tahu di mana harus berbaris. Mereka mulai mengoper bola satu sama lain alih-alih mengambil bola dan berlari kencang menuju gawang. Dengan tim Scrum pertama saya, saya terlalu fokus untuk mencoba membuat mereka senang. Ini adalah kesalahan karena saya membiarkan kompromi yang ternyata merugikan tim. Misalnya, tim tidak melihat nilai dari rapat stand-up harian, jadi saya berkompromi dan mengadakan rapat stand-up tiga hari seminggu.

Mengapa tim melakukan itu? Karena mereka tidak tahu apa-apa. Mereka mencoba melakukan apa yang mereka yakini. Sama seperti seekor gajah. Penafian Meskipun saya bergaul dengan beberapa orang yang cukup besar, saya belum pernah melatih seekor gajah. Saya tidak tahu apakah cerita berikut ini benar atau tidak. Sejauh yang saya tahu, ini hanya omong kosong yang dibuat-buat. Bagaimana cara mencegah gajah seberat 8 ton melakukan apa pun yang diinginkannya? Anda menjepit kakinya ke rantai ringan yang diikatkan ke paku

yang ditancapkan ke tanah. Hewan darat terbesar di Bumi dapat ditopang oleh rantai ringan yang diikatkan ke paku yang dapat dengan mudah ditarik keluar dari tanah tanpa perlu berpikir. Gajah bahkan tidak akan mencoba menarik rantai tersebut karena ia yakin tidak dapat menarik paku tersebut keluar dari tanah.

Seperti cerita yang beredar, tak lama setelah seekor gajah lahir, seorang pelatih gajah akan mengambil bayi gajah tersebut dan merantainya ke paku yang ditancapkan ke tanah. Bayi gajah tersebut akan menarik, menarik, dan mencoba melepaskan diri, tetapi pada akhirnya ia akan menyerah. Ia tidak akan pernah mencoba memutuskan rantai itu lagi bahkan setelah ia tumbuh menjadi makhluk terbesar dan terkuat yang pernah ada. Saya kira itulah sebabnya mereka mengatakan gajah tidak pernah lupa. Banyak profesional perangkat lunak yang saya kenal juga demikian. Mereka terbelenggu pada cara lama dalam melakukan sesuatu. Ketika Anda menyajikan ide-ide seperti pengembangan berbasis pengujian, atau tidak menghabiskan waktu berbulan-bulan mengembangkan dokumen desain super-teknis, atau bahkan mengadakan rapat stand-up harian, mereka tidak percaya itu akan berhasil. Mereka tidak melihat nilainya.

Mereka terbelenggu pada taruhan itu. Saya telah menghabiskan sebagian besar waktu saya sebagai Scrum Master dan pelatih yang bekerja dengan tim mainframe. Anda tahu, Big Iron. Tim yang membuat kode dalam bahasa mesin dan mendukung produk yang dianggap penting oleh perusahaan besar. Ketika saya pertama kali mengemukakan ide Sprint empat minggu, tanggapannya adalah bahwa tidak mungkin mereka dapat menulis cerita yang dapat dilakukan dalam Sprint. Ini adalah mainframe. Basis kode warisan yang besar dengan segala macam hal rumit yang terjadi. Saya senang mengatakan bahwa tim-tim ini telah lepas dari belenggu. Hari ini, mereka menyelesaikan cerita dalam Sprint mereka dan melihat nilai Scrum. Sebagian besar tim bahkan menjalankan Sprint dua minggu, dan pelanggan mereka terlibat dalam proses pengembangan.

Bantu saya. Lain kali Anda berkata kepada diri sendiri "itu tidak akan pernah berhasil" karena birokrasi atau takut gagal atau apa pun, pikirkan tentang gajah yang percaya bahwa ia tidak dapat membebaskan diri. Tim tempat saya bekerja tidak bermaksud tidak masuk akal. Mereka hanya tidak mempercayai kerangka kerja Scrum. Mereka dikondisikan untuk berpikir bahwa cara mereka bekerja di bawah Waterfall adalah satu-satunya cara untuk menjadi sukses. Ada juga rasa takut yang sehat akan perubahan. Seperti yang saya katakan sebelumnya, mereka biasa bertanya kepada saya, "Kapan saya bisa menulis kode?" Itu agak sarkastis tetapi menunjukkan kurangnya kepercayaan pada kerangka kerja Scrum. Di Waterfall, pengembang dinilai berdasarkan seberapa banyak kode yang ditulis. Penguji dinilai berdasarkan berapa banyak bug yang mereka temukan.

Tim menemukan kenyamanan dalam perencanaan. Mereka tidak mencoba menghalangi usaha saya. Mereka benar-benar mengira mereka mencoba menyelamatkan penilaian kinerja mereka. Tim itu benar-benar tidak tahu apa yang tidak mereka ketahui. Mereka melihat Scrum melalui sudut pandang "kami selalu melakukannya dengan cara ini." Sama seperti tim sepak bola, saya benar-benar tidak tahu apa yang saya lakukan ketika menjadi Scrum Master, jadi saya membenamkan diri dalam semua hal Scrum. Saya sangat kacau, jadi

saya menjalin hubungan pembinaan dengan pelatih Agile setempat. Saya berjejaring dengan Scrum Master lainnya. Saya berpartisipasi dalam lokakarya dan mendapatkan pelatihan. Saya mengasah kapak saya.

Berikut cerita lainnya. Seorang penebang kayu bergabung dengan kru baru. Sebagai orang baru, ia benar-benar ingin membuat tim terkesan, jadi ia bertanya kepada mandor tentang jumlah pohon terbanyak yang pernah ditebang oleh siapa pun dalam kru ini dalam sehari. "Enam," kata mandor. Jadi penebang kayu itu keluar, mengerahkan upaya yang sangat besar, dan menebang banyak pohon. Di penghujung hari, ia bertanya kepada mandor berapa banyak pohon yang telah ditebangnya. "Lima," kata mandor. Didorong oleh kenyataan bahwa ia sudah hampir mencapai tujuan, penebang kayu itu keluar keesokan harinya dan bekerja lebih keras lagi. Sekali lagi, ia bertanya kepada mandor berapa banyak pohon yang telah ia tumbangkan ke tanah. "Empat," kata mandor. Hal ini membuat si penebang kayu kesal, jadi keesokan harinya dia mengerahkan semua yang mungkin bisa dia kumpulkan.

Di penghujung hari, dia hampir tidak punya cukup energi untuk bertanya kepada mandor berapa banyak pohon yang telah ditebangnya hari itu. "Tiga," kata mandor. Si penebang kayu berlutut dan meletakkan kepalanya di tangannya. Mandor meletakkan tangannya di bahu si penebang kayu. Dia berkata, "Selama bertahun-tahun saya tidak pernah melihat orang bekerja sekeras Anda, tetapi izinkan saya bertanya sesuatu." "Apa?" tanya si penebang kayu. "Tiga hari terakhir ini... apakah Anda punya waktu untuk mengasah kapak Anda?" tanya mandor. Saya meluangkan waktu untuk mempelajari lebih lanjut tentang Agile. Anggota tim Scrum perlu melakukan hal yang sama. Sisihkan kapasitas selama Sprint agar anggota tim dapat memperbaiki diri. Asah kapak Anda! Setelah saya memahami Scrum dan mendapatkan pelatihan, saya mulai menerapkan apa yang saya pelajari ke tim yang saya layani. Ada begitu banyak hal yang harus dilakukan, dan saya termotivasi untuk menjadi agen perubahan.

Namun, pelatih saya meminta saya untuk sedikit mengurangi antusiasme saya. Seperti yang dikatakannya, jangan keluar dan mencoba merebus lautan. Ketika saya mengikuti pelatihan dasar Angkatan Udara, instruktur latihan melakukan segala daya mereka untuk melucuti semua hal tentang kehidupan sipil. Mereka menyuruh kami mencukur semua rambut wajah, mencukur kepala, dan mengenakan seragam tanpa tanda nama agar kami tampak sama. Kami tidur di ruang terbuka yang sama. Kami berlatih, makan, dan pergi ke kelas bersama. Kami tidak lagi memiliki individualitas. Yang kami miliki hanyalah satu sama lain. Itu tidak terjadi dengan tim Scrum, meskipun saya pernah mengancam untuk meminta anggota tim melakukan push-up satu atau dua kali. Ketika transformasi Agile dimulai, sepertinya semuanya perlu diubah. Seperti yang saya katakan, saya tergoda untuk mencoba keluar dan mengubah semuanya sekaligus. Itu bukan cara untuk melakukan transformasi.

Jadi, bagaimana Anda menerapkan Scrum? Dua hal yang membuat Scrum berhasil adalah melibatkan pelanggan dalam proses pengembangan dan bekerja sesuai Definisi Selesai. Mulailah dengan berfokus pada melibatkan pelanggan dan menyelesaikan DoD. Hal terpenting yang dapat dilakukan tim Scrum adalah mendefinisikan DoD. Kriteria penerimaan, DoR, dan DoD membuat kebijakan tim Anda terlihat. Setiap kali cerita "diserahkan", hal-hal seperti DoR

dan DoD menghilangkan dugaan dalam proses dan membuat semuanya berjalan lancar. Tim bekerja untuk memuaskan DoD. Dengan cara ini, tidak diragukan lagi bahwa ceritanya sudah selesai. Setelah bertahun-tahun bermain gim untuk merilis tepat waktu dengan kualitas yang dipertanyakan, kami sekarang meminta tim untuk mengerjakan cerita hingga selesai. Bukan kode yang lengkap. Bukan "semacam diuji." Selesai.

Mulailah dengan dasar-dasarnya. Bangun Backlog dan fokus pada upacara Scrum. Buat tim menyetujui Definisi Siap dan Definisi Selesai. Jangan terjebak dalam perdebatan tentang DoR dan DoD. Batasi waktu rapat dan tuliskan sesuatu dan tempel di dinding bahkan jika menurut Anda itu menyebalkan. Biasakan tim dengan irama Scrum dan lakukan inspeksi serta adaptasi. Hasilnya akan lebih baik. Oh ya, dan mulailah mengadakan rapat berdiri setiap hari. Tidak adanya dukungan eksekutif dan tidak bekerja sebagai tim akan menghancurkan Scrum lebih cepat daripada apa pun. Pastikan bahwa orang-orang yang memiliki wewenang untuk benar-benar mendorong perubahan "sepenuhnya terlibat" dengan transformasi Agile. Mencoba untuk melakukan inspeksi dan adaptasi sebagai tim Scrum dalam struktur manajemen perintah dan kontrol yang kaku dan tidak Agile tidak akan berhasil.

Scrum membutuhkan tingkat kepercayaan yang tinggi baik di tingkat tim maupun organisasi. Perintah dan kontrol adalah apa yang diterapkan ketika Anda tidak memercayai orang-orang yang melakukan pekerjaan. Saya yakin ada orang-orang di luar sana yang hanya mengikuti arus. Saya hanya belum pernah bertemu dengan mereka, dan mereka jelas bukan bagian dari tim Scrum saya, yang memiliki satu hal yang saya inginkan di semua tim saya kebanggaan. Saat saya remaja, Sony Walkman adalah barang baru yang sedang tren, dan musik Anda ada dalam bentuk kaset (atau vinil). Cakram padat belum populer saat itu. Ayah sahabat saya punya Sony Betamax dan beberapa film yang direkamnya dari stasiun baru Ted Turner yang bernama "Superstation TBS." Selama musim panas, saya bekerja di konstruksi hingga perkemahan sepak bola dimulai. Musim panas tahun 1983, saya berusia 16 tahun, menjaga seorang tukang batu bernama Tom Aquino di pabrik batu bata dan mortir. Itu pekerjaan yang berat sangat berat.

Tidak diragukan lagi bahwa saya mengerjakan perangkat lunak hari ini karena saya tahu betapa sulitnya bekerja di konstruksi. Saat itu awal Agustus; sebenarnya, jika saya ingat dengan benar, itu adalah salah satu pekerjaan terakhir yang saya lakukan tahun itu. Perkemahan sepak bola akan segera tiba, dan saya akan segera menukar sepatu bot kerja saya dengan sepasang sepatu khusus. Kami sedang membangun teras depan rumahnya dari batu bata pagi itu. Saya kembali melakukan pekerjaan saya seperti biasa mencampur adukan semen dan memilah batu bata memastikan bahwa saya hanya akan memberikan yang bagus kepada Tom. Tom sedang mempersiapkan pekerjaan memasang tali untuk memastikan semuanya tegak lurus dan persegi. “Tegak lurus” itu seperti mantra bagi orang itu. Saya rasa saya mendengarnya jutaan kali musim panas itu saja. Setelah memasang senarnya, Tom hanya berdiri di sana selama setidaknya 10 menit sambil memandangi bagian depan rumahnya. Bagian bawah sudah digali dan menunggu kami, saya sudah menyiapkan batu bata dan mortirnya, tetapi Tom masih berdiri di sana.

Akhirnya, saya bertanya, "Apakah kamu sudah siap?" "Ayo mulai," jawab Tom.

Menjelang makan siang, kami telah menyelesaikan hampir separuh teras. Sudut kiri sudah selesai, dengan kolom bata setinggi sekitar 6 kaki. Jadi, saya duduk di sana sambil makan dan memperhatikan Tom. Dia berdiri di tempat yang sama seperti sebelumnya... sambil melihat. Kemudian hal yang tak terduga terjadi... Dia mendorong kolom itu. Sejujurnya, saya pikir dia sudah gila. "Apa yang kamu lakukan?" teriak saya. "Itu tidak benar," kata Tom. "Hah?" Saya tergagap. "Itu persegi... Saya sendiri yang mengukurnya." Tom hanya menatap saya dan berkata, "Itu persegi dan tegak lurus sempurna, tetapi tidak sesuai dengan standar saya. Setiap kali seseorang melewati rumah ini, mereka akan melihat teras ini dan tahu bahwa saya yang membangunnya." Dia melepas kacamatanya dan menatap saya dengan tatapan tajam.

"Dan jika itu bukan yang saya inginkan, saya tidak ingin itu dikaitkan dengan nama saya." Saya ingat ketika saya menjadi teknisi servis di toko Computerland. Setiap kali saya membuka casing Apple Macintosh asli, saya akan melihat bagian dalamnya. Tepat di bagian dalam casing plastik itu terdapat tanda tangan setiap orang yang merakit Mac tersebut. Para pengembang, penguji, dan staf dokumentasi yang bekerja dengan saya memiliki kebanggaan seperti itu. Akhirnya, organisasi menjadi lebih Agile. Tim memeriksa dan beradaptasi, dan kami menjadi lebih baik. Sepak bola juga berjalan dengan baik. Kami bersenang-senang, berolahraga, dan mempelajari permainannya. Kalau boleh saya katakan, tim putri saya menikmati camilan terbaik di babak pertama.

11.2 PERAN PELATIH DALAM SCRUM MASTER

Pelatihan itu seperti otot; Anda akan menggunakannya atau kehilangannya. Seperti yang telah saya katakan sebelumnya dalam buku ini, ada perbedaan antara pelatihan, pengajaran, dan pendampingan. Ada juga perbedaan antara pelatihan Agile dan pelatihan profesional. Pelatihan Agile adalah tempat Anda membantu orang-orang dengan kompetensi dan keterampilan Agile mereka. Tujuan pelatihan profesional adalah untuk membantu individu meningkatkan kinerja. Ini tentang pemberdayaan dan tanggung jawab pribadi. Pelatih membantu "yang dibina" dengan mengubah pola pikir dan perilaku. Ketika anggota tim Scrum tidak memenuhi harapan, mereka membutuhkan pembinaan. Ajukan pertanyaan terbuka. Mereka akan membantu orang yang dibina untuk berpikir dan menghasilkan solusi yang dapat dipercaya oleh mereka. Mereka memiliki solusi karena mereka yang menemukannya. Karena Scrum Master tidak memberi tahu siapa pun apa yang harus dilakukan, mereka dipandang sebagai rekan, bukan sebagai ahli.

Anggota tim tahu apa yang harus dilakukan. Tugas pelatih adalah membantunya menemukan bahwa ini masalahnya. Sebelum terlibat dalam sesi pelatihan, pelatih perlu memastikan bahwa anggota tim terbuka terhadap gagasan untuk dilatih. Saya ingat seorang teman saya bercerita kepada saya bagaimana dia pergi menemui chiropractor-nya dan orang itu langsung menariknya, melemparkannya ke meja, dan mulai menanganinya. Chiropractor itu mengejutkannya. Hal buruknya adalah teman saya memiliki pulpen di saku bajunya, dan dia tidak pernah sempat mengeluarkannya. Tak perlu dikatakan lagi, chiropractor itu harus membelikan baju baru untuk teman saya ketika pulpennya patah.

Hal yang sama berlaku untuk pelatihan. Anda tidak ingin langsung melatih orang.

Jelaskan bahwa Anda ingin memulai hubungan pelatihan dengan mereka dan apakah mereka terbuka terhadap gagasan tersebut. Setelah orang tersebut terbuka untuk dilatih, buatlah pertemuan. Saya berusaha untuk menyingkirkan semua gangguan selama sesi pelatihan berlangsung. Tidak boleh ada laptop atau tablet dalam pertemuan. Saya berusaha untuk mengambil ponsel saya dan meletakkannya di atas meja di depan saya. Ini adalah isyarat simbolis yang menunjukkan bahwa sesi pembinaan adalah hal terpenting yang sedang berlangsung saat ini. Awali pertemuan dengan membiarkan orang tersebut berbicara. Pada titik ini, dengarkan. Maksud saya, dengarkan dengan sungguh-sungguh. Ingat, pembina tidak ada di sini untuk memecahkan masalah. Jangan duduk di sana mencoba memikirkan sesuatu yang cerdas untuk dikatakan atau bagaimana memecahkan masalah bagi orang tersebut. Dengarkan saja. Biarkan mereka mengeluarkan semua uneg-uneg mereka.

Membiarkan semuanya keluar dan menjernihkan suasana merupakan hal yang cukup membersihkan dan menyegarkan. Pada titik tertentu, pembina akan mulai mengajukan pertanyaan terbuka. Misalnya, "Apa yang telah Anda coba lakukan hingga titik ini?" Salah satu pertanyaan favorit saya adalah "Jika ini berjalan dengan sempurna, seperti apa bentuknya?" Dengan meminta orang yang dibina menjelaskan bagaimana skenario yang sempurna akan berhasil, Anda mungkin akan membuat mereka langsung mengatakan solusinya. Pada titik tertentu dalam percakapan ini, orang yang dibina akan mengalami apa yang saya sebut "momen sejuta dolar." Mereka akan mulai berbicara tentang cara-cara yang masuk akal untuk menangani situasi tersebut. Sebagai seorang pembina, saya memahami itu dan tidak akan melepaskannya. Mengajukan pertanyaan terbuka memberdayakan anggota tim.

Ketika seorang pelatih memberi tahu seseorang apa yang harus dilakukan, ada kesan superioritas di sana. Ketika pelatih membimbing orang tersebut ke jawaban, pelatih tersebut lebih terlihat seperti rekan sejawat. Hal ini menciptakan penerimaan di pihak orang yang dibimbing. Mereka sekarang merasa bahwa mereka memiliki solusi tersebut. Pada titik ini, harus ada kesepakatan tentang tindakan yang harus dilakukan dan rapat tindak lanjut yang dijadwalkan. Pelatih perlu memastikan bahwa orang tersebut menindaklanjuti tindakan yang harus dilakukan. Ada juga kemungkinan besar bahwa tindakan tersebut dapat mengarah pada sesi pelatihan lebih lanjut di masa mendatang. Setiap orang di tim Scrum memiliki bakat yang dapat membantu membuat tim menjadi lebih baik. Terserah kepada pelatih untuk membuka bakat tersebut.

Saya akan mengatakan bahwa sesi pelatihan dapat menjadi emosional. Orang mungkin menangis. Orang mungkin menyuarakan rasa frustrasi mereka. Frustrasi bukanlah hal yang buruk. Saya telah menghadapi berbagai macam frustrasi selama menjadi Scrum Master. Anggota tim yang frustrasi, manajer yang frustrasi, Pemilik Produk yang frustrasi; saya sendiri pun pernah merasa frustrasi. Saya mulai belajar bahwa frustrasi adalah bagian dari kemajuan. Jika Anda dapat belajar dari frustrasi Anda, itu bisa sangat produktif. Terus terang saja, Anda harus peduli terhadap sesuatu untuk menjadi frustrasi karenanya. Daripada mencoba melatih diri saat frustrasi, saya belajar melatih diri saat frustrasi. Cari tahu apa penyebab sebenarnya dari frustrasi tersebut dan latih orang tersebut sehingga mereka menemukan cara untuk mengatasi masalah tersebut.

Jadi, Apa yang Dilakukan Agile Coach? Jika Scrum Master melatih, mengapa organisasi membutuhkan Agile coach? Ketika tim sepak bola memenangkan kejuaraan, apakah mereka menyingkirkan semua pelatih? Bagaimanapun, tim tersebut jelas tahu cara menang. Pemain terbaik kini dapat melatih tim saat mereka bermain. Tidak ada tim yang melakukan itu. Tim sepak bola membutuhkan pelatihan. Begitu juga tim Scrum. Agile coach berkomitmen untuk menciptakan tim dengan kinerja tinggi. Ya, sebagian dari itu adalah mengembangkan pemimpin, tetapi itu bukan satu-satunya fungsi pelatih. Pelatih Agile bekerja untuk meningkatkan kompetensi Scrum Master, Pemilik Produk, dan manajer organisasi.

Pemikiran Acak

Saya ingin menyampaikan beberapa hikmah yang saya pelajari dalam perjalanan Agile saya. Tidak ada maksud atau alasan untuk apa yang saya bagikan di sini. Hanya hal-hal yang menurut saya penting untuk disampaikan.

Mengapa Scrum?

Scrum menyediakan sarana untuk mendapatkan umpan balik sebelum hal yang Anda bangun benar-benar dibangun. Melibatkan pelanggan secara aktif memberi mereka visibilitas yang tak tertandingi dalam proses pengembangan. Pelanggan dan pemangku kepentingan melihat perubahan dan penyempurnaan produk utama secara "waktu nyata" saat terjadi. Ini berarti menghasilkan produk yang tepat. Bahkan serangkaian persyaratan yang jelas mengharuskan tim untuk membuat beberapa asumsi, dan persyaratan biasanya berubah sebelum akhir proyek.

Saat persyaratan muncul dan mengubah Scrum, tim memiliki kemampuan untuk menerima perubahan dan memberikan nilai yang benar-benar dibutuhkan pelanggan. Fleksibilitas dibangun dalam kerangka kerja Scrum. Perubahan memang diharapkan, dan Scrum menyediakan perangkat untuk merencanakan variabilitas dan menerima perubahan. Sifat iterasi mengungkap masalah sejak dini. Pengujian tertanam dalam kerangka kerja, memberikan pelanggan produk yang lebih stabil karena cacat ditemukan dan diperbaiki lebih awal dalam proses pengembangan.

Jangan Lakukan Hal-hal Bodoh

Command-and-control mengasumsikan bahwa Anda perlu diberi tahu cara melakukan pekerjaan Anda. Ya, itu kasar, tetapi itulah kenyataannya. Scrum berasumsi bahwa sebuah tim akan menemukan jawabannya dengan bereksperimen, memeriksa, dan beradaptasi. Frasa "kami selalu melakukannya dengan cara ini" harus dihapus dari kosakata Anda. Pertanyakan semuanya dan jangan takut untuk memeriksa dan beradaptasi. Budaya Scrum menghargai keterbukaan dan kolaborasi. Jangan takut untuk menegur sesuatu jika itu tidak masuk akal.

Namun, sadarilah bahwa Anda mungkin yang bodoh. Pernahkah Anda berada di tengah-tengah pertengkaran dengan seseorang dan menyadari bahwa Anda salah tetapi terus berdebat karena ego? Ya, itu canggung, dan saya akui bahwa saya telah melakukannya lebih dari sekali. Bersikaplah cukup rendah hati untuk memahami bahwa Anda mungkin perlu memikirkan kembali posisi Anda. Kakak saya selalu menggambarkan saya sebagai "licik seperti palu godam." Saya bisa bersikap sombong, keras kepala, berisik, kasar, keras kepala, dan tidak perhatian (terkadang semuanya sekaligus). Saya juga tulus. Saya harap itu bisa menebus hal-

hal lain yang saya sebutkan.

Gagal Cepat, Belajar Cepat

Kegagalan adalah guru terbaik di dunia. Salah satu hal yang saya anjurkan agar dilakukan tim adalah bereksperimen dan gagal jika perlu. Saya hanya ingin itu terjadi dengan cepat. Tim yang gagal berulang kali bukanlah hal yang sehat dan tidak memberikan nilai kepada pelanggan. Gagal cepat, belajarlah darinya, periksa dan beradaptasi, lalu lanjutkan. Tim yang sering bereksperimen dan belajar dari kegagalan mereka sedang dalam perjalanan menjadi tim yang berkinerja tinggi. Tim harus bereksperimen sehingga mereka dapat mempelajari apa yang bisa dan tidak bisa dilakukan... titik. Tidak ada yang seperti eksperimen untuk membantu tim memahami kemampuannya.

Secara pribadi, saya merasa bahwa kegagalan dalam hidup tidak hanya tidak dapat dihindari, tetapi juga merupakan persyaratan untuk menjadi lebih baik. Ingat pepatah di bagian belakang kaus yang diberikan saudara laki-laki saya kepada saya: Anda harus kalah untuk menang. Siapa pun yang mengatakan sebaliknya adalah pembohong. Segala sesuatu yang Anda lakukan dalam hidup akan disertai dengan keberhasilan dan kegagalan. Apa yang Anda lakukan dengan kegagalan itulah yang menentukan seberapa sukses Anda di masa depan. Anda tidak boleh takut gagal; dan ketika Anda gagal, Anda perlu belajar dan berkembang. Untuk lebih jelasnya, saya tidak mengatakan bahwa Anda harus menantang pelari cepat Olimpiade untuk berlomba lari.

Saya tidak tertarik membangun budaya kegagalan. Yang ingin saya katakan adalah kita harus menetapkan tujuan dan mencapainya seperti gorila punggung perak yang baru saja menghabiskan empat minuman berenergi. Lihatlah dari sudut pandang ini: meskipun kegagalannya dahsyat, lebih besar dari yang pernah Anda bayangkan; meskipun tidak ada yang bisa diselamatkan dari percobaan yang gagal, Anda hanya mengacaukan satu Sprint. Jika kita tidak menjadi lebih baik, kita kehilangan inti permasalahan di sini. Bagaimana Anda tahu apa yang berhasil dan tidak berhasil jika Anda tidak mencoba hal-hal baru? Rayakan keberhasilan, belajar dari kegagalan, dan jadilah yang terbaik yang Anda bisa.

11.3 MENYELESAIKAN SEMUA CERITA DALAM SPRINT BACKLOG

Tim perlu fokus untuk menyelesaikan semua cerita yang mereka komitmenkan dalam Perencanaan Sprint setiap Sprint. Sangat penting bagi tim untuk tidak membiarkan cerita tergelincir ke Sprint berikutnya. Cerita menerima poin kecepatan saat cerita tersebut selesai dan diterima oleh Pemilik Produk. Cerita yang terbawa tidak menerima kredit sebagian. Semua poin cerita diakui dalam Sprint saat cerita tersebut selesai. Membiarkan cerita tergelincir akan berdampak buruk pada kecepatan tim. Ini lebih dari sekadar kecepatan. Tidak menyelesaikan cerita dalam satu Sprint membuat kita tergoda untuk kembali ke perilaku bergaya Waterfall. Ingat, idenya adalah untuk menyampaikan perangkat lunak yang berfungsi setiap Sprint.

Membiarkan cerita terlewat tidak akan memungkinkan tim untuk mendemonstrasikan produk minimal yang layak yang telah mereka janjikan untuk disampaikan. Hal ini juga menciptakan efek domino. Ketika sebuah cerita terlewat ke Sprint berikutnya, ada kemungkinan besar tim tidak akan dapat menyelesaikan semua cerita di Sprint berikutnya.

Jadi, tim tidak menyelesaikan semua cerita lagi. Cerita terlewat lagi, dan lagi, dan lagi. Estimasi yang ambisius atau komitmen tim yang terlalu antusias adalah penyebab umum ketika sebuah cerita gagal ditutup dalam Sprint-nya. Tim Scrum harus mencoba untuk tidak menjanjikan banyak hal dan memberikan banyak hal dalam hal komitmen Sprint. Sempurnakan cerita dan tugas sekecil mungkin sehingga lebih mudah untuk melakukan sejumlah pekerjaan yang diperlukan untuk menyelesaikannya. Ingat, Agile adalah tentang menjadi lebih baik dan menyampaikan hal-hal yang diinginkan pelanggan dengan lebih cepat.

Agile menantang Anda untuk melakukan apa yang masuk akal bukan untuk mengikuti protokol seperti robot, tetapi untuk melakukan apa yang masuk akal. Menjadi lebih baik. Yang terpenting, lakukan apa yang dibutuhkan tim Scrum untuk membuat tim tersebut sukses. Tidak ada yang namanya waktu luang selama Sprint. Jika Anda merasa malas, berarti Anda melakukan sesuatu yang salah. Prioritas pertama Anda harus selalu pekerjaan yang telah disepakati tim selama rapat Perencanaan Sprint, tetapi jangan terlalu terpaku pada protokol hingga Anda benar-benar melambat. Agile memaksa Anda untuk fokus pada apa yang ingin Anda capai.

Ada juga masalah menghabiskan terlalu banyak waktu untuk membicarakan cerita dan tidak cukup waktu untuk benar-benar mengerjakannya. Kesempurnaan adalah musuh bebuyutan tim Scrum. Ingat, perangkat lunak yang berfungsi adalah yang kita cari. Bangun sesuatu dan sampaikan ke tangan pelanggan. Daripada berdebat tentang cara kerja suatu fungsi, serahkan kepada pelanggan dan biarkan mereka memberi tahu apa yang mereka suka dan tidak suka tentangnya. Apa gunanya mendapatkan sesuatu yang tepat di mata tim tetapi salah total di mata pelanggan? Buat fungsionalitas ke titik yang cukup baik untuk saat ini, selesaikan ceritanya, dapatkan umpan balik pelanggan, dan lakukan sesuatu dengannya. Kekuatan Datanglah Tanggung Jawab Agile (Scrum) seharusnya membawa kegembiraan bagi tim. Tim harus "melakukan tugasnya" tanpa ada yang mengendalikan pekerjaan mereka. Pemilik Produk dan Scrum Master tidak memiliki suara dalam cara tim memberikan nilai. Tim mengelola dirinya sendiri.

Itu bisa menjadi kejutan budaya yang cukup besar bagi tim yang terbiasa memiliki manajer pengembangan yang memberi tahu mereka bagaimana ia ingin sesuatu dibangun. Ini adalah kebebasan yang sangat besar beranikah saya katakan kekuatan yang besar? Dan dengan ini datanglah tanggung jawab. Merupakan tanggung jawab tim untuk melakukan pekerjaan... untuk membuat keputusan sendiri dan memberikan apa yang diminta kepada mereka. Tim membutuhkan wewenang untuk membuat keputusan, dan kemampuan untuk menangani dampak dari keputusan tersebut. Tidak masuk akal bagi tim Scrum untuk harus menunggu seseorang yang berada empat tingkat di atas rantai manajemen untuk memberi mereka lampu hijau untuk melakukan sesuatu.

Scrum mengharuskan mereka yang paling terdampak oleh keputusan untuk membuat keputusan. Saya pernah mendengar ritual Agile (stand-up, Sprint Planning, Retrospectives, dan sebagainya) digambarkan sebagai gangguan yang menghalangi pekerjaan pengembangan yang "nyata". Hm... tidak. Pertemuan ini bukan "overhead." Pertemuan ini diadakan untuk menyediakan struktur dan ritme yang diperlukan agar pekerjaan dapat diselesaikan, dan agar

informasi penting dan pembaruan dapat dikomunikasikan di antara anggota tim. Keputusan tentang arah dibahas dan komitmen dibuat satu sama lain. Pertemuan memungkinkan seluruh tim untuk berpartisipasi, berkolaborasi, dan (berani saya katakan) mengatur diri sendiri. Tidak boleh ada penantian bagi seseorang di luar tim untuk membuat keputusan tentang apa pun. Tim harus memiliki wewenang yang tepat untuk melakukan pekerjaan tersebut. Bukan tanggung jawab Product Owner untuk memutuskan bagaimana tim mengimplementasikan suatu fitur.

PO perlu fokus pada gambaran yang lebih besar, memprioritaskan dan membangun Backlog. Mari kita hadapi, pengembang ingin membangun sesuatu. Dengan memprioritaskan fitur, PO menjaga tim "tetap fokus" dan tidak terganggu oleh hal-hal yang tidak penting. Jika PO terlalu terlibat dalam pekerjaan pengembangan tim, dia tidak menyempurnakan dan memprioritaskan. Jika Anda benar-benar merasa bahwa "ritual" Agile ini tidak bermanfaat atau hanya sesuatu yang manajemen minta Anda lakukan, maka Anda dan tim Anda tidak bertanggung jawab atas apa yang sedang terjadi. Scrum Master terbaik di dunia tidak akan dapat membuat ritual Agile apa pun bermanfaat jika tim tidak tertarik. Kenyataannya, tim Scrum memegang semua kendali. Tidak ada lagi manajer pengembangan yang harus ditanggapi. Timlah yang memiliki kekuatan.

Dan tanggung jawab…

Kunci Waktu Pekerjaan Anda

Saya rasa saya belum mendefinisikan apa itu kunci waktu. Itu lucu bagi saya, karena itulah yang memungkinkan saya menulis buku ini sejak awal. Kunci waktu adalah membatasi jumlah waktu yang dihabiskan untuk sesuatu. Ya, sesederhana itu. Gunakan kunci waktu untuk memastikan Anda tetap produktif. Saya mengakuinya; saya tidak menyukainya saat pertama kali mendengarnya. Sekarang, saya tidak bisa hidup tanpanya. Kunci waktu mengendalikan kecenderungan alami saya untuk menunda-nunda. Ini membantu saya mengatur waktu dengan lebih baik. Misalnya, saya menyisihkan satu jam dalam sehari untuk mengerjakan buku saya. Saya menyetel pengatur waktu dan mulai menulis. Setelah enam puluh menit, saya beralih ke tugas berikutnya. Ada sesuatu tentang jam yang bergerak itu yang tampaknya memungkinkan saya untuk lebih fokus.

Ini juga merupakan cara yang bagus untuk menjaga rapat stand-up harian tetap terkendali. Jika ada satu keluhan yang menurut saya paling sering saya dengar selama menjadi Scrum Master, itu adalah ini: "Kami mengadakan terlalu banyak rapat." Tanggapan saya yang biasa adalah Anda tidak akan merasa seperti itu jika rapat itu bernilai, dan kemudian saya menantang mereka untuk menciptakan nilai itu. Namun, saya telah melihat fenomena aneh terjadi. Orang-orang mungkin membenci semua rapat ini, tetapi mereka tidak akan berhenti berbicara saat mereka berada di dalamnya. Ambil contoh rapat stand up harian. Stand-up seharusnya berlangsung tidak lebih dari 15 menit. Secara teknis, di sinilah anggota tim seharusnya membuat komitmen satu sama lain.

Itulah mengapa disebut stand-up tidak nyaman untuk berdiri dalam waktu lama, jadi semua orang berdiri dalam rapat, membuatnya singkat. Bukan itu yang terjadi. Tim memperlakukannya seperti rapat status Waterfall gaya lama. Orang-orang membicarakan hal-

hal yang seharusnya ada di rapat di tempat parkir, atau di tempat-tempat yang tidak jelas. Solusinya adalah dengan membatasi waktu rapat. Cara saya menerapkan pembatasan waktu adalah dengan menggunakan penghitung waktu mundur yang menjengkelkan. Saya cukup menyetel penghitung waktu selama 15 menit dan membiarkannya menghitung mundur. Hasilnya luar biasa. Representasi visual waktu yang terus berdetak membuat semua orang tetap pada jalurnya, dan stand-up jarang berlangsung lebih dari 15 menit.

Beban Progresif

Bagaimana tim Scrum melakukan peningkatan bertahap dan konstan dari waktu ke waktu? Pernahkah Anda duduk dan memikirkannya? Bagaimana Anda benar-benar dapat melakukannya? Itu bisa menjadi tugas yang cukup menakutkan. Bagaimana Anda menumbuhkan budaya peningkatan berkelanjutan? Ini mengingatkan saya pada kisah Milo dari Croton. Milo adalah pegulat di Yunani kuno yang pada dasarnya tidak terkalahkan. Dia memenangkan enam medali Olimpiade, dan melakukan banyak prestasi kekuatan. Ia terkenal karena memenangkan pertandingan gulat dan makan (ia konon makan 20 pon daging sehari), tetapi ia paling terkenal karena legenda tentang cara ia berlatih. Legenda mengatakan bahwa ia akan mengambil anak sapi yang baru lahir, mengangkatnya, lalu meletakkannya di pundaknya.

Ia kemudian akan berjalan-jalan di kota Croton dengan anak sapi itu di pundaknya. Ia melakukan ini setiap hari tanpa henti. Saat anak sapi itu tumbuh, ia menjadi semakin berat, dan akhirnya Milo berjalan-jalan dengan seekor banteng dewasa di pundaknya. Kuncinya di sini adalah bahwa Milo memulai dengan tugas yang dapat dikelola. Saya yakin bahwa saat anak sapi itu tumbuh, dibutuhkan usaha untuk mengangkatnya. Usaha itu memicu pertumbuhan, dan tubuh Milo beradaptasi dengan beban tersebut. Prinsip ini disebut beban progresif. Anda mulai dengan beban yang dapat dikelola dan seiring waktu menambah sedikit lebih banyak setiap latihan. Hal yang sama juga berlaku untuk tim Scrum. Seperti Milo, kita perlu terus mencari cara untuk menjadi lebih baik. Kuncinya adalah membuat perubahan kecil dari waktu ke waktu.

Periksa apa yang Anda lakukan, beradaptasi dengan membuat perubahan kecil, dan jangan pernah berhenti berkembang. Anda tidak naik lift menuju ketangkasan. Anda naik tangga satu langkah pada satu waktu. Salah satu bahaya yang dihadapi oleh tim Scrum adalah rasa puas diri yang sama saja. Kita perlu waspada agar tidak jatuh ke dalam perangkap itu. Jika kita benar-benar menginginkan peningkatan berkelanjutan, fokus kita seharusnya adalah ini—bagaimana kita membuat semuanya lebih baik? Jika Anda tidak menemukan nilai dalam apa yang Anda lakukan, terserah Anda untuk menentukan cara memperbaiki situasi itu. Periksa dan beradaptasi! Dua kata paling berbahaya dalam bahasa Inggris adalah "Saya tahu." Jangan pernah berhenti berusaha untuk menjadi sedikit lebih baik setiap hari.

Jadilah Termostat, Bukan Termometer

Pernahkah Anda masuk ke sebuah ruangan dan percakapan yang sedang berlangsung terhenti sebelum Anda tiba? Itu sering terjadi pada saya. Saya orang yang "minum Kool Aid." Penginjil Agile. Saya bisa menerima label-label itu, tetapi saya lebih suka dikenal sebagai termostat. Termometer memberi tahu Anda suhu. Termostat mengontrol suhu. Dalam hal

Agile, saya benar-benar seperti penginjil. Saya merasa seperti mengulang-ulang diri saya sendiri. Periksa dan adaptasi. Bersikaplah transparan. Bagaimana kita dapat memberikan nilai lebih cepat? Saya percaya bahwa mengulang pesan tanpa henti pada akhirnya akan membuatnya melekat. Jika saya mendengar Anda mengeluh tentang sesuatu yang berhubungan dengan Agile, saya mungkin akan menegur Anda, membimbing Anda, atau bertanya bagaimana menurut Anda kita dapat memperbaikinya. Saya mengubah suhu. Sebagai Scrum Master dan pelatih, saya mengatur suhu di ruangan, bukan hanya memperhatikan apakah ruangan panas atau dingin.

Bersikaplah Fleksibel

Saat Anda Agile, Anda fleksibel. Fleksibel berarti fleksibel, mudah beradaptasi, anggun. Saat saya memikirkan tentang bagaimana kami telah memberikan perangkat lunak selama saya berada di tim Waterfall, kata yang terpikir oleh saya adalah kaku. Kaku kami memberikan kepada pelanggan apa yang kami pikir mereka inginkan. Atau lebih buruk lagi, kami memberikan kepada pelanggan apa yang diinginkan seseorang dalam manajemen, tetapi bukan apa yang diinginkan pelanggan. Scrum tidak kaku. Itu berarti bahwa kita harus mampu bereaksi terhadap masukan pelanggan … agar fleksibel… Itu juga berarti bahwa apa pun yang sedikit pun mirip Waterfall harus mati. Ketika Anda menerapkan prinsip Waterfall dalam lingkungan Agile, Anda tidak lagi melakukan Scrum. Yang Anda dapatkan adalah proyek Waterfall yang dipecah menjadi beberapa bagian bukan Agile. Tim tidak bereaksi terhadap masukan pelanggan.

Pelanggan bukan bagian dari proses pengembangan. Saya akan menjelaskannya seperti ini. Katakanlah Anda menghabiskan waktu di driving range sepanjang hidup Anda. Anda mengayunkan kayu besar dan memukul bola golf. Teman Anda mengundang Anda untuk bermain golf. Ketika Anda sampai di sana, Anda memberi tahu teman Anda bahwa meskipun Anda bermain di lapangan golf, Anda merasa nyaman hanya dengan memukul bola. Anda lebih suka tidak menggunakan iron atau putter. Dalam kasus ini, Anda tidak bermain golf. Kenyataannya, Anda hanya bermain-main. Pada abad ke-21, kita tidak mampu untuk bermain-main. Persyaratan berubah begitu cepat sehingga pelanggan kita bahkan tidak dapat memahami apa saja persyaratan tersebut sehingga mereka dapat memberi kita gambaran tentang apa yang mereka butuhkan.

Sangat penting bagi kita untuk berkolaborasi dengan pelanggan kita sehingga kita dapat memenuhi persyaratan yang muncul. Bahkan jika Anda berkata, "Saya akan bermain sesuai aturan, tetapi saya tidak suka menggunakan putter," dan Anda hanya mencoba untuk menempatkan bola sedekat mungkin dengan pin, Anda tetap tidak benar-benar bermain golf... bukan? Tim Anda tidak Agile jika Anda hanya menambahkan rapat ke apa yang selalu Anda lakukan. Jika Anda mengerjakan desain di Sprint pertama, membuat kode di Sprint kedua, dan menguji di Sprint ketiga, Anda tidak melakukan Scrum. Jika Anda memberikan nilai dalam Sprint, tetapi tidak diuji, didokumentasikan, dan dapat diinstal, Anda tidak melakukan Scrum. Anda juga tidak boleh mengatakan bahwa Anda "melakukan Scrum." Anda tidak hanya "melakukan Scrum." Anda beralih ke pola pikir yang berfokus pada pelanggan.

Pertama-Tama, Lihatlah Diri Anda Sendiri

Dalam situasi apa pun yang muncul, baik dalam tim Scrum, atau konteks pekerjaan atau pribadi lainnya, pertama-tama lihatlah diri Anda sendiri. Bersiaplah untuk mengarahkan jari yang Anda tunjuk ke orang lain ke diri Anda sendiri. Pada akhirnya, orang yang bertanggung jawab adalah Anda. Transformasi Agile tidak selalu indah. Ketika keadaan memburuk, dan itu pasti akan terjadi, bagian dari menjadi seorang pemimpin adalah mengambil tanggung jawab. Tidak ada yang menghentikan "permainan menyalahkan" seperti melangkah maju dan memberi tahu semua orang bahwa Anda bisa melakukannya dengan lebih baik. Scrum adalah kerangka kerja yang hebat, tetapi ia juga berhasil mengungkap disfungsi. Ada banyak peluang untuk perbaikan. Jika rapat Perencanaan Sprint tidak berjalan sesuai rencana, karena cerita tidak disempurnakan dengan benar, jangan biarkan tim menyalahkan Pemilik Produk.

Saya melangkah maju dan memberi tahu tim bahwa Scrum Master tidak melakukan pekerjaan yang cukup baik dalam membimbing PO. Saya memberi tahu tim bahwa saya akan lebih memperhatikan cerita apa yang ingin disampaikan PO ke Sprint sebelum rapat Perencanaan berikutnya. Jika rapat stand-up harian berlangsung terlalu lama, itu karena saya tidak cukup menekankan batasan waktu. Setiap kali situasi muncul, saya mencari tahu di mana saya bisa melakukan sesuatu yang berbeda dan bertanggung jawab atasnya. Bagi saya, itu bagian dari menjadi pemimpin dan bagian dari bersikap transparan. Pikirkan seperti ini.

Jika saya memutuskan untuk menanam kebun dan melakukan semua pekerjaan seperti memilih tanaman, menggali tanah, dan menanamnya di tanah, tetapi semua tanaman mati apakah itu salah tanamannya? Apakah Anda menyalahkan orang yang membeli tanaman itu, atau tanah tempat Anda menanamnya? Atau apakah Anda melihat cara Anda merawat tanaman? Apakah Anda cukup menyiramnya? Apakah Anda menanamnya di tempat yang mendapat cukup sinar matahari? Apakah Anda memberinya cukup (atau terlalu banyak) pupuk? Pemimpin pelayan perlu bertanggung jawab sehingga mereka dapat menggerakkan tim untuk melakukan inspeksi dan adaptasi. Meskipun itu berarti Anda terkadang harus mengorbankan diri. Pimpin Tim Ada perbedaan antara bos dan pemimpin.

Tim Scrum perlu mengatur diri sendiri; Saya sudah mengatakannya beberapa kali dalam buku ini. Tim yang efektif juga mengelola diri sendiri. Tim tidak membutuhkan bos. Mereka membutuhkan pemimpin. Seseorang yang mendukung tim. Seorang pemimpin yang efektif tahu pertanyaan yang tepat untuk diajukan, dan kapan harus menanyakannya. Pertanyaan yang diajukan pemimpin mendorong tim untuk memikirkan masalah alih-alih memberi tahu mereka apa yang perlu mereka lakukan. Seorang pemimpin memfasilitasi proses pengambilan keputusan dan membantu tim mencapai kesepakatan. Menjadi pengelola mandiri tidak berarti bahwa siapa pun sekarang dapat melakukan apa pun yang mereka inginkan. Masih ada kendali, tetapi kendali itu sekarang terletak pada kepercayaan. Manajemen tidak lagi bertanggung jawab atas semua keputusan.

Mereka sekarang memercayai tim untuk menetapkan harapan dan memenuhinya. Orang-orang sekarang bertanggung jawab atas pekerjaan mereka sendiri yang berarti bahwa mereka sekarang mengatur diri sendiri, mengelola diri sendiri, dan mengambil inisiatif. Hal ini memungkinkan tim untuk berinovasi dan mengambil risiko. Seorang pemimpin yang baik

memungkinkan kepercayaan. Itu mungkin hal terpenting yang dapat dilakukan oleh seorang Scrum Master. Positif, Positif, Positif Buatlah keputusan sadar untuk bersikap positif. Setiap hari, dalam setiap jenis situasi, seorang Scrum Master harus menjadi orang yang "menentukan suasana" di ruangan. Pada akhirnya, kepositifan adalah sebuah pilihan.

Kepositifan itu menular, dan itu benar-benar dapat mengubah hasil. Pada kenyataannya, kita benar-benar mendapatkan apa yang kita harapkan. Sikap positif, "melihat dari sisi positif" dapat memberikan dampak yang nyata. Angka ajaib atau titik manisnya adalah "tiga banding satu," seperti dalam tiga pernyataan positif untuk setiap pernyataan negatif. Jika Anda menginginkan tim yang terdiri dari bintang rock berkinerja tinggi, Anda perlu mendapatkan rasio itu mendekati lima banding satu.

Bagi saya, ini sulit. Sifat saya adalah menanggapi komentar negatif dengan lebih banyak dosis kenegatifan dan sisi sarkasme yang sehat. Faktanya, saya adalah ahli sarkasme sejati. Saya dikenal menggunakan sarkasme sebagai mekanisme pertahanan, senjata ofensif, dan mekanisme koping. Jadi, saya memutuskan untuk mengatasi situasi ini dengan karet gelang yang saya pakai di pergelangan tangan. Setiap kali saya merasa ingin bersikap negatif (atau saat monster sarkasme muncul), saya menarik karet gelang itu dan... thawk! Saya mengingatkan diri sendiri untuk tidak membiarkan itu terjadi. Untuk mengatakan sesuatu yang positif.

Kembangkan Sikap Bersyukur

Rasa syukur adalah sesuatu yang tidak datang secara alami. Sangat mudah untuk terjebak dalam dunia kecil kita sendiri dan tidak berhenti serta menyadari apa yang benar-benar penting. Bahkan, tampaknya semakin tua saya, semakin saya menyadari banyaknya kurangnya kesadaran diri di sekitar saya. Saya sedang keluar dengan salah satu putri saya beberapa hari yang lalu, menjalankan beberapa tugas dan mengurus beberapa barang ketika saya melihat sesuatu. Sebenarnya, saya akan menggolongkannya sebagai semacam pencerahan kecil. Kami sedang berada di sebuah toko ketika saya menyadari bahwa hampir semua orang di sekitar saya merasa kesal dengan sesuatu.

Saya berdiri di sana, di tengah semua kekacauan itu, terkesima oleh keajaiban yang ada di toko fisik. Pikirkanlah ya, lorong-lorongnya terlalu sempit, tempatnya selalu ramai, tetapi Anda dapat menemukan apa pun yang Anda inginkan di sana. Dari sayap ayam hingga pakan ayam. Itu benar-benar keajaiban. Pikirkan apa yang diperlukan untuk mendapatkan satu barang yang tersedia di toko itu dari produksi hingga distribusi. Kemudian pikirkan tentang kemudahannya. Jika saya membutuhkan, saya tidak tahu kaus pada pukul 3 pagi saya dapat berlari ke sana dan mendapatkannya. Saya akan memberikan contoh lain. Kembali pada tahun 2006, saya berada di Jerman selama dua minggu. Kopi Eropa bukanlah pilihan saya (maaf atas permainan kata-katanya).

Saya adalah orang paling bahagia di Eropa ketika saya menemukan kedai kopi berantai Amerika. Saya jarang pergi ke kedai kopi ini ketika saya di rumah. tetapi saya bersyukur ketika saya menemukannya di Jerman. Biarkan saya melukiskan sebuah gambaran untuk Anda. Anda tiba di kantor dan memarkir mobil di tempat parkir. Langkah Anda terasa bersemangat saat memasuki kantor. Anda menyapa semua orang dengan senyuman saat berjalan ke meja untuk memasukkan laptop ke docking station. Anda membutuhkan waktu 10 menit untuk

menyelaraskan lubang-lubang bodoh di bagian bawah laptop dengan tonjolan-tonjolan kecil di docking station. Anda melangkah ke ruang istirahat untuk mengambil secangkir kopi pagi dan disambut oleh tiga teko kopi kosong.

Saat membuat teko kopi, Anda mengutuk orang yang mengambil cangkir terakhir dan tidak memiliki sopan santun untuk membuat teko kopi baru. Hal itu mungkin cukup untuk merusak hari siapa pun. Itu terjadi pada saya hampir setiap pagi, tetapi saya tidak membiarkannya memengaruhi sikap saya. Saat saya mengalami hari yang buruk, saya merasa menyegarkan untuk meluangkan waktu dan membuat daftar semua hal yang saya syukuri. Hal itu membuat saya lebih memahami dan memungkinkan saya untuk fokus pada betapa baiknya hidup saya sebenarnya. Faktanya, saya selalu mencatat tujuh hal yang saya syukuri setidaknya sekali dalam seperempat tahun dan menempelkannya di bilik kerja saya. Ini berfungsi sebagai pengingat terus-menerus bahwa saya tidak punya alasan untuk menjadi pemarah.

Fokus pada Batu-batu Besar Terlebih Dahulu

Anda mencoba memasukkan sebanyak mungkin batu ke dalam stoples kaca. Secara logika, Anda harus mulai dengan batu-batu besar, lalu menambahkan batu dan kerikil yang lebih kecil. Alasannya adalah jika Anda mulai dengan hal-hal kecil, Anda mungkin tidak punya ruang untuk batu-batu besar nanti. Batu-batu besar adalah yang paling penting….yang memiliki dampak signifikan. Apa saja batu-batu besar Scrum?

Backlog adalah Segalanya, Jadi Perhatikan Baik-Baik

Menurut pengalaman saya, satu upacara Agile yang paling sering ingin dilewati oleh tim adalah rapat Penyempurnaan Backlog. Ini adalah kesalahan besar. Backlog adalah sumber kehidupan tim Scrum. Backlog harus diperlakukan sebagai sesuatu yang berharga, karena memang begitulah adanya. Backlog adalah daftar "hal-hal" yang diprioritaskan. Secara khusus, Backlog adalah "hal-hal" yang mungkin disertakan dalam produk Anda serta persyaratan untuk perubahan ini. Backlog harus menjadi satu-satunya sumber perubahan yang diminta, dan harus terlihat oleh siapa pun.

Cukup sederhana, bukan? Ini bagian pentingnya: Backlog harus menjadi artefak yang hidup. Item dalam Backlog harus berkembang dan mungkin berubah seiring dengan tim mempelajari lebih lanjut tentang bagaimana pelanggan akan menggunakan fungsionalitas yang diperlukan dan lingkungan tempat tim akan beroperasi. Jika tim dan Pemilik Produk tidak secara teratur menyempurnakan cerita dalam Backlog, Anda tidak akan menghasilkan fitur yang berharga bagi pelanggan Anda sesederhana itu.

Berikan Umpan Balik dengan Tegas

Umpan balik adalah alasan mengapa kita bekerja dalam iterasi singkat. Itulah alasan mengapa kita mengadakan rapat tinjauan Sprint. Kurangnya umpan balik adalah alasan mengapa metodologi Waterfall tidak lagi disukai oleh banyak tim. Umpan balik itu penting, jadi kita harus terus-menerus melakukannya. Tidak cukup hanya mengadakan rapat Tinjauan Sprint. Anda perlu melibatkan pelanggan Anda. Anda memiliki pelanggan dalam rapat Tinjauan Sprint Anda... benar? Anda perlu mendapatkan umpan balik dari sebanyak mungkin pelanggan. Ini berarti Anda perlu melakukan apa pun yang memungkinkan untuk mendapatkan umpan balik yang diinginkan tim Anda. Jika pelanggan tidak memberikan apa

yang Anda cari, mintalah.

Ini tentang Memeriksa dan Beradaptasi, bukan Mengeluh dan Menahan Diri

Agile seharusnya seperti nirwana bagi siapa pun yang terlibat di dalamnya. Mengapa? Memeriksa dan beradaptasi. Anda harus mencari, atau setidaknya berpikir, tentang cara untuk membuat segalanya dan apa pun menjadi lebih baik... terus-menerus. Ketika saya mendengar orang mengeluh, saya selalu bertanya-tanya mengapa. Anda memiliki kesempatan unik untuk melakukan sesuatu tentang hal itu. Saya paham bahwa saya dikenal sebagai orang yang selalu berpikir positif. Sikap positif adalah suatu keharusan, terutama dalam lingkungan Agile. Jika Anda tidak menyukainya, cari tahu cara memperbaikinya.

11.4 TERAPKAN DOD DAN DOR

Seperti yang saya jelaskan di Bab 4, sebelum lepas landas, seorang pilot melakukan sesuatu yang disebut daftar periksa pra-penerbangan. Hal ini dilakukan agar pilot tidak ragu bahwa pesawatnya layak terbang. Tidak ada pilot yang akan turun landasan tanpa melakukan pemeriksaan pra-penerbangan. Hal yang sama berlaku untuk Definisi Selesai (DoD) dan Definisi Siap (DoR).Bagaimana Anda tahu kapan sebuah cerita siap untuk dimasukkan ke dalam Sprint? Anda tahu dari DoR. Yang kami maksud dengan siap adalah siap, bukan siap atau semacam siap.

DoR adalah kumpulan semua hal yang diperlukan agar sebuah cerita pengguna dapat dikembangkan. DoR dapat berubah dan berkembang melalui rilis, tetapi seharusnya hanya ada satu yang digunakan untuk semua cerita pengguna. Jika DoR didefinisikan dengan benar, setiap orang dalam tim harus mengetahui kapan sebuah cerita dapat dimasukkan ke dalam Sprint mana pun. DoD adalah hal yang harus dikerjakan oleh tim. DoD memberi tim cara yang bagus untuk melaporkan kemajuan pada sebuah fitur.

Anggap saja ini sebagai daftar periksa yang akan memvalidasi tidak hanya bahwa semua tugas telah diperhitungkan, tetapi juga bahwa Anda tidak lupa melakukan apa pun di tengah tekanan iterasi yang dibatasi waktu. DoD dan DoR harus menjadi dokumen hidup yang tumbuh dan berubah seiring tim mencapai tingkat kematangan Agile yang baru. DoD dan DoR memungkinkan tim untuk tidak pernah memasukkan apa pun ke dalam Sprint yang belum siap, dan tidak pernah mengeluarkan apa pun dari Sprint yang belum selesai.

Pastikan Anda Berbicara dengan Orang yang Tepat

Saat kami mengundang pelanggan ke rapat tinjauan Sprint, kami menginginkan orang-orang yang, seperti yang telah saya katakan sebelumnya, berada di depan layar. Dengan kata lain, orang-orang yang benar-benar menggunakan produk kami. Orang-orang yang duduk di belakang komputer, atau tablet, atau perangkat seluler dan menggunakan perangkat lunak yang kami hasilkan. Di depan layar. Bukankah masuk akal untuk mendapatkan umpan balik dari orang yang benar-benar menggunakan produk? Bagi saya, ini adalah hal yang mudah, tetapi apakah kita melakukannya? Sering kali saya melihat manajemen, atau VP pembelian, atau seseorang di atas rantai komando berinteraksi dengan kita.

Pikirkan seperti ini... Jika saya membuka kedai limun, apakah Anda akan bertanya kepada anak-anak yang minum limun tentang rasanya atau apakah Anda akan bertanya kepada

orang tua yang membeli limun untuk anak-anak mereka? Saya yakin orang tua akan mengatakan hal-hal seperti: "Anda mengenakan biaya terlalu mahal." "Saya tidak suka tampilan stan Anda." Perhatikan, tidak ada yang perlu dikhawatirkan tentang rasa minuman yang Anda jual. Kenyataannya, tidak ada perbedaan dengan perangkat lunak. Kami mencoba membuat orang-orang yang menggunakan perangkat lunak kami lebih seperti penggemar daripada pelanggan.

Namun, bukankah kami ingin fokus pada mereka yang memegang kendali keuangan? Anda tahu, orang yang membuat perjanjian pembelian? Izinkan saya menjawab pertanyaan itu dengan pertanyaan. Apakah menurut Anda orang-orang di pusat data (di atas kaca) senang ketika Anda mengambil perangkat lunak yang mereka suka gunakan? Dari apa yang saya lihat di lokasi pelanggan, ketika pengguna tidak senang, tidak ada yang senang. Tidak ada yang salah dengan menyertakan orang-orang seperti agen pembelian dan wakil presiden dalam demo Sprint, tetapi pastikan orang-orang yang menggunakan perangkat lunak juga ada di sana.

Pemikiran Akhir

Saya ingin menutup buku ini dengan sebuah cerita dan pemikiran. Perjalanan saya menuju Agile, yah, unik, paling tidak. Saya ingin berbagi sedikit wawasan sebelum saya menutup semuanya.

Jangan Pernah Lupa Bahwa Pelangganlah yang Bertanggung Jawab

Teman saya Ken McGaffic menyelenggarakan festival bernama Old Fashioned Christmas in the Woods setiap tahun di bulan Oktober. Saya membantunya, biasanya menjual tiket masuk. Saya yakin Ken tidak keberatan jika saya terlihat seperti Sinterklas. Harus saya akui, itu sangat berguna di festival Natal. Saya orang yang mudah bergaul, jadi saya merasa cocok. Saya bisa berbicara dengan orang-orang dan mendapatkan teman baru. Saya juga bisa melihat langsung bagaimana rasanya memiliki pelanggan yang lebih bersikap seperti penggemar. Ribuan orang datang ke lokasi festival Shaker Woods di Ohio untuk mengunjungi 215 stan kerajinan (dan/atau 25 pedagang makanan), untuk mencari sesuatu yang istimewa untuk liburan. Hal yang paling menakjubkan bagi saya adalah apa yang saya temukan saat berbicara dengan orang-orang. Saya tipe orang yang mau berbicara dengan siapa saja. Seperti yang saya katakan sebelumnya, saya orang yang mudah bergaul.

Salah satu tanggung jawab saya di festival adalah "menjaga kawat" sebelum pertunjukan dimulai. Yang saya lakukan adalah memasang pita peringatan di bagian festival tempat orang-orang masuk dan menjaga agar pelanggan tetap di tempat sampai para pedagang siap memulai hari. Sekali lagi, tampil seperti Sinterklas sangat berguna. Ukuran tubuh saya juga cocok untuk situasi ini. Sambil berdiri di dekat kawat, saya berbicara dengan orang-orang. Ini adalah cara yang bagus untuk menghabiskan waktu bagi pelanggan dan saya. Saat kami mengobrol, saya terus mendapat komentar seperti ini: “Saya menantikan ini setiap tahun.

Selama bertahun-tahun, kami telah menjadikannya sebagai acara keluarga.” “Ini adalah pameran kerajinan terbaik di Pantai Timur.” Apa hubungannya pameran kerajinan dengan Agile? Singkatnya, semuanya. Tempat pameran Shaker Woods benar-benar berada di antah berantah. Salah satu tempat yang seperti “belok ke kanan di ladang jagung ketiga”, tetapi

orang-orang datang dari lebih dari 30 negara bagian setiap tahun. Sebagai bagian dari Scrum, kami mencoba untuk melibatkan pelanggan lebih banyak dalam proses pengembangan. Kami meminta mereka untuk bergabung dengan kami di akhir Sprint dan berbicara tentang apa yang kami lakukan dan mendapatkan umpan balik tentang apa yang mereka suka dan tidak suka.

Idenya adalah jika kami memberi pelanggan apa yang mereka inginkan dan apa yang mereka butuhkan, atau membuat barang yang mereka lihat dan berkata, “Saya harus memilikinya!” maka proses penjualan akan berjalan dengan sendirinya. Dengan kata lain, kami tidak perlu menjual produk kami kepada siapa pun. Pelanggan kami akan berebut untuk mendapatkannya. Itulah yang saya inginkan. Pelanggan yang akan berdiri di tengah hutan pada pagi Oktober yang dingin karena mereka menginginkan apa yang dijual. Jadi, bagaimana teman saya Ken menciptakan jenis pelanggan yang sama dengan yang kami cari? Dia menyesuaikan pameran dengan target pelanggannya.

Dia merasa bahwa wanita berusia di atas 25 tahun menyukai suasana luar ruangan yang unik. Seperti yang dia katakan, "Mereka suka berjalan-jalan di hutan tanpa sepatu mereka terkena lumpur." Dia tahu apa yang diinginkan pelanggannya dan berusaha menyenangkan mereka. Semua kerajinan yang dijual di festival ini dibuat dengan tangan oleh para perajin. Kualitasnya sangat tinggi. Untuk bisa mengikuti festival, seorang perajin tidak hanya harus memiliki kerajinan buatan tangan mereka juga harus mengenakan kostum dan melakukan demonstrasi tentang cara membuat kerajinan mereka selama pameran. Kami berdua mengejar hal yang sama pelanggan yang menyukai apa yang kami hasilkan dan terus datang lagi untuk mendapatkan lebih banyak lagi.

Kami berdua mencoba melakukannya dengan cara yang sama. Dengarkan pelanggan, berikan apa yang mereka inginkan, dan pastikan kualitasnya tinggi. Jangan pernah lupa bahwa ini tentang pelanggan. Ketika kita berpikir bahwa kita adalah "suara pelanggan", ketika kita lebih mementingkan penulisan kode daripada memberikan nilai, kita tidak lagi melayani kepentingan terbaik pelanggan. Mereka akan pergi ke tempat lain ke seseorang yang melayani mereka. Kapan terakhir kali pelanggan mengucapkan terima kasih kepada Anda? Salah satu hal yang paling saya nikmati dari waktu saya di Christmas in the Woods adalah ketika orang-orang meninggalkan festival, mereka mengucapkan terima kasih kepada saya. Bukan ucapan terima kasih setengah hati saat mereka melewati saya dalam perjalanan menuju mobil mereka, tetapi ucapan terima kasih yang tulus karena telah membantu menjadikan festival ini seperti sekarang.

Mereka Adalah Manusia

Tim Scrum adalah kumpulan orang. Jangan pernah lupakan itu. Hargai "sifat manusiawi" orang-orang dalam tim. Bahkan, hargai sifat manusiawi setiap orang yang Anda temui. Menurut pengalaman saya, kebanyakan orang tidaklah tidak masuk akal; tekanan hiduplah yang membuat mereka seperti itu. Saat menulis buku ini, ayah saya meninggal, ibu mertua saya didiagnosis menderita kanker payudara, dan seorang teman baik saya yang bermain musik dengan saya selama bertahun-tahun juga meninggal secara tiba-tiba. Semua itu bahkan membuat saya menjadi sedikit kurang positif dari biasanya. Saya benar-benar percaya bahwa tidak ada orang yang bangun di pagi hari dan merencanakan bagaimana ia akan

bersikap sulit saat mandi.

Orang yang memiliki sikap seperti itu mungkin berada di bawah tekanan yang lebih besar daripada yang dapat Anda bayangkan karena satu dan lain alasan. Bahkan mungkin bukan sesuatu di tempat kerja yang memengaruhi mereka. Penguasaan dan pembinaan Scrum lebih banyak tentang keterampilan orang daripada Manifesto Agile. Kerangka kerja itu penting, tetapi orang-orangnya lebih penting. Salah satu hal yang selalu saya tanyakan kepada orang-orang adalah apakah mereka bahagia. Seorang Scrum Master bertanggung jawab atas kebahagiaan tim yang dipimpinnya. Saya akan sampaikan ini. Jika Anda menunjukkan kepada orang-orang bahwa Anda benar-benar peduli dengan kesejahteraan mereka, mereka akan menunjukkannya. Tidak hanya kepada Anda tetapi juga kepada orang-orang di sekitar mereka. Itulah yang membuat Scrum berhasil.

DAFTAR PUSTAKA

Anderson, D. (2012). *Kanban: Successful evolutionary change for your technology business*. Blue Hole Press.

Anderson, D. J. (2010). *Agile management for software engineering: Applying the theory of constraints for business results*. Pearson Education.

Babich, A., & Chen, L. (2017). *Implementing Scrum in non-software teams: A global approach*. Pearson Education.

Beck, K., & Cunningham, W. (1995). *The story of eXtreme Programming*. ACM SIGPLAN Notices, 30(7), 18-20. https://doi.org/10.1145/211937.211943

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., … & Thomas, D. (2001). *Manifesto for Agile software development*. Agile Alliance.

Bock, S. (2015). *Agile in the real world: A complete guide to agile project management*. McGraw-Hill.

Boehm, B., & Turner, R. (2004). *Balancing agility and discipline: A guide for the perplexed*. Addison-Wesley.

Charette, R. (2014). *Agile and scrum in the enterprise: Software development methodologies in practice*. Wiley.

Cohn, M. (2005). *Agile estimating and planning*. Prentice Hall.

Cohn, M. (2009). *Succeeding with agile: Software development using Scrum*. Addison-Wesley Professional.

Dingsøyr, T., & Moe, N. B. (2014). *Agile software development: A decade of research*. Springer.

Fitzgerald, B., & Stol, K. (2017). *The role of agile methodologies in modern software development: A comprehensive review*. International Journal of Information Management, 37(5), 477-490. https://doi.org/10.1016/j.ijinfomgt.2017.06.002

Fitzpatrick, S., & Bond, P. (2017). *The agile product manager's guide to success*. McGraw-Hill.

Fowler, M. (2004). *The new methodology*. IEEE Software, 21(1), 13-14. https://doi.org/10.1109/MS.2004.1250634

Gaim, W. D., & Pylkka, A. (2018). *Agile practices in software development*. McGraw-Hill.

Highsmith, J. (2002). *Agile software development ecosystems*. Addison-Wesley.

Hoda, R., & Nissen, K. (2016). *The impact of agile practices in large-scale software development*. IEEE Transactions on Software Engineering, 42(8), 753-768. https://doi.org/10.1109/TSE.2016.2548980

Hossain, L., & Babar, M. A. (2009). *Agile methodologies in the software industry: Challenges and solutions*. Springer.

Jarvis, P. (2016). *Scrum: A framework for software engineering*. Wiley.

Ken Schwaber (2004). *Agile Project Management with Scrum*. Microsoft Press.

Khan, M. A. (2017). *Implementing agile methods: From waterfall to scrum*. Pearson Education.

Larman, C. (2004). *Agile and iterative development: A manager's guide*. Addison-Wesley.

Larman, C., & Vodde, B. (2009). *Practice agile: Software development for the real world*. Addison-Wesley Professional.

Larman, C., & Vodde, B. (2016). *Lean agile adoption and transformation: A guide for Scrum teams and organizations*. Addison-Wesley.

Leffingwell, D. (2011). *Agile software requirements: Lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley.

Madsen, O. S., & Østerlie, T. (2019). *Scrum and agile methodologies in the service sector*. International Journal of Software Engineering, 44(3), 121-134. https://doi.org/10.1016/j.ijse.2018.12.003

Martin, J. (2006). *Agile project management: The complete guide*. Pearson Education.

Martin, R. C. (2008). *Clean code: A handbook of agile software craftsmanship*. Prentice Hall.

Martinez, A., & Rios, M. (2017). *Scaling agile with Scrum: A comprehensive guide*. Addison-Wesley.

McHugh, J., & Jovanovic, M. (2016). *Scrum for non-technical managers*. Wiley.

Milani, P., & Forrester, J. (2018). *Building effective teams with agile development*. Wiley.

Orsborn, S., & Williams, A. (2015). *Scrum principles for product and service management*. Springer.

Pichler, R. (2010). *Agile product management with Scrum: Creating products that customers love*. Addison-Wesley Professional.

Pichler, R. (2015). *Agile product management with Scrum*. Addison-Wesley Professional.

Poppendieck, M., & Poppendieck, T. (2003). *Lean software development: An agile toolkit*. Addison-Wesley.

Rubin, K. S. (2012). *Essential Scrum: A practical guide to the most popular agile process*. Addison-Wesley Professional.

Schreiner, P. (2015). *Agile project management with Scrum: Tools and techniques for success*. Wiley.

Schwaber, K., & Sutherland, J. (2017). *The Scrum guide: The definitive guide to Scrum: The rules of the game*. Scrum.org. https://www.scrumguides.org

Smits, P. (2014). *Agile project management with Scrum: How to deliver high-value software in less time*. Wiley.

Steinberg, L., & Harris, M. (2013). *Using Scrum in large-scale product development*. Lean Methods.

Stober, T., & Hansmann, L. (2016). *Agile processes in software engineering and extreme programming*. Springer.

Sutherland, J. (2010). *Scrum and the Agile framework: A case study approach*. Wiley.

Sutherland, J. (2014). *Scrum: The art of doing twice the work in half the time*. Crown Business.

Sutherland, J., & Schwaber, K. (2011). *Scrum: A revolutionary approach to building teams, beating deadlines, and boosting productivity*. Crown Business.

Sutherland, J., & Schwaber, K. (2016). *The Scrum guide: The definitive guide to Scrum: The rules of the game*. Scrum.org. https://www.scrumguides.org

Thomas, M., & Geiger, A. (2014). *Agile retrospectives: Making good teams great*. Pragmatic Bookshelf.

Vaidyanathan, G. (2014). *Scrum essentials: A short guide to the agile framework*. McGraw-Hill.

Widman, D., & Williams, A. (2010). *Scrum in action: Applying the Scrum framework to real-world projects*. Wiley.

Williams, L. (2012). *Agile software development: A complete guide to Scrum methodologies*. Wiley.

Williams, L., & Kessler, R. R. (2000). *All together now: A case study of an agile team*. IEEE Software, 17(6), 36-45. https://doi.org/10.1109/52.888533

Dr. Agus Wibowo, M.Kom, M.Si, MM.

Manajemen Proyek Scrum

Pada Pengembangan Perangkat Lunak,Produk dan Layanan

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id