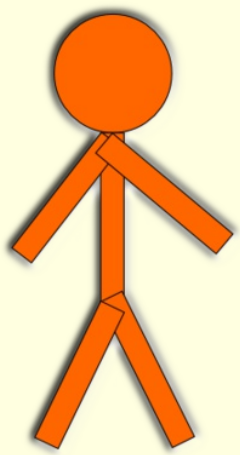
Lichte Use Cases met de kracht van modelleren

Met dank aan hoofdsponsor NS konden we een van de godfathers van Use Cases, Ivar Jacobson, aanschouwen en aanhoren. Met een typisch Zweeds accent vertelde hij zijn verhaal over Use Cases en de ontwikkeling naar Use Case 2.0.

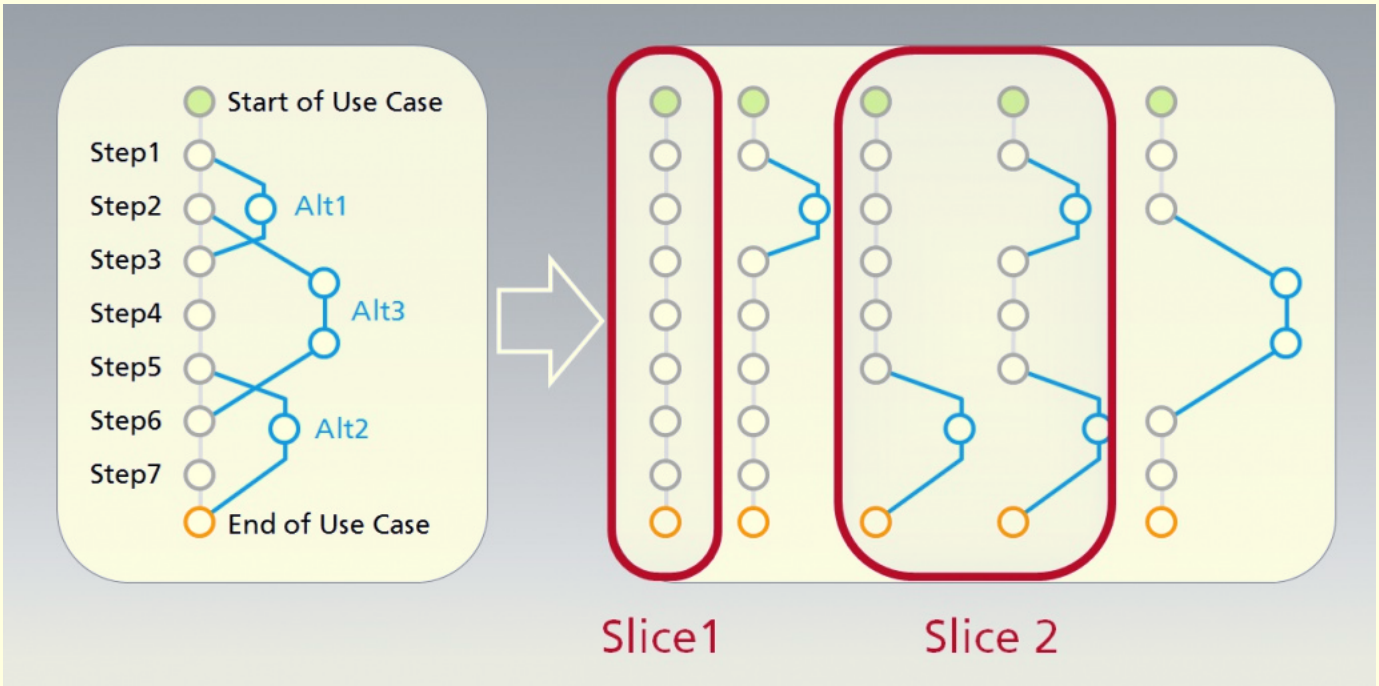
door Johan Olenziel

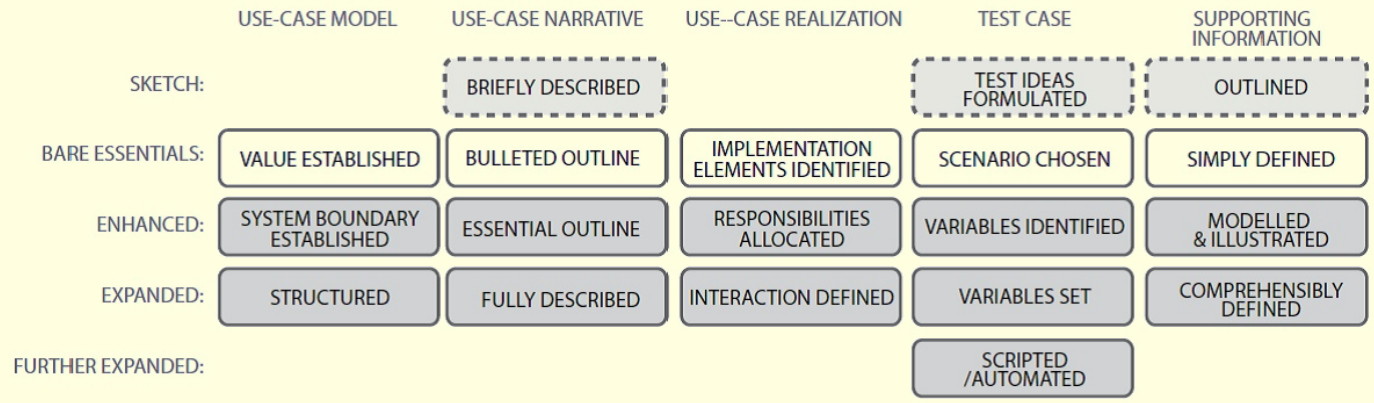


Jacobson schetst hoe hij tot hier gekomen is. Hoe zijn eerste artikel over Object Oriented Development wordt afgewezen als inbreng voor de eerste OOPSLA conferentie in 1986. Gelukkig mag hij het jaar erop dit artikel wel presenteren. In de jaren ’90 worden Use Cases steeds populairder, UML wordt ontwikkeld en Rational komt met RUP – waar Use Cases een integraal onderdeel van zijn.

Jacobson is geen definitiefreak, een ieder mag Use Cases zo interpreteren als hij zelf wil. 'Iedereen denkt Use Cases te begrijpen en dat is goed zo', volgens de meester. De kracht van Use Cases ligt er volgens Jacobson in dat het focus op waarde legt, samenhang geeft en business en IT samenbrengt. Verder zijn Use Cases volgens hem gemakkelijk te definiëren en te begrijpen.

Maar in deze tijden, met zoveel nieuwe requirements technieken, hebben dan we nog wel Use Cases nodig? Ja, zegt Jacobson, want het is de beste techniek om samenhang tussen requirements te beheren. En hij heeft voldoende kritiek op andere requirements management technieken. Zo zijn volgens hem User Stories alleen maar goed voor kleine systemen en kleine teams. Features zijn heel geschikt voor product management, maar niet voor requirements management. Tevens stelt hij dat declaratieve requirements heel geschikt zijn om atomaire eisen te vangen, maar verder niet. En dat domain modeling alleen geschikt is voor functioneel simpele en informatierijke systemen. Maar ook RUP was niet feilloos, het





uit: Jacobson: Use Case 2.0

was vooral een te zwaar administratief proces geworden. Zo telde de manual 700 pagina's ("RUP is a hooligan" volgens Jacobson). Ook werden er te vaak te veel features vastgelegd.

> Zo bestaat Essential RUP, als onderdeel van Use Case 2.0, uit 23 kaartjes. Dat is heel wat werkbaarder dan de 700 pagina's tellende manual voor RUP 1.0.

Daarom is hij met Use Case 2.0 gekomen. Zo bestaat Essential RUP, als onderdeel van Use Case 2.0, uit 23 kaartjes. Dat is heel wat werkbaarder dan de 700 pagina's tellende manual voor RUP 1.0. Het voordeel van Use Case 2.0 is volgens Jacobson dat je gemakkelijk kunt opschalen van kleine teams naar grote teams. Daarnaast kun je je reikwijdte vergroten, wat Jacobson opschalen noemt. Hiermee bedoelt hij dat je met Use Cases in eerste instantie het requirements proces ondersteunt. Maar je kunt er ook andere processtappen als analyse, bouw, test en business design mee ondersteunen. Verder benoemt Jacobson inzoomen. Hierbij hoef je niet alle Use Cases op detailniveau uit te werken. Soms kun je met een hoofdlijn uit de voeten, terwijl je bij systemen van leven-en-dood (ziekenhuis, defensie) wel alle details uitwerkt.

Met bovengenoemde aanpassingen wil Jacobson bereiken dat Use Case 2.0 zich opnieuw focust op de essentie en recente innovaties en verbeteringen als test-driven development, Kanban en Scrum ondersteunt. Maar wees gerust, Use Cases blijven lichte simpele verhalen, met nog steeds dezelfde betekenis: wat moet een systeem bieden om een gebruiker iets wenselijks te laten bereiken?

In het beeld van Jacobson bestaat een Use Case uit een set van gestructureerde verhalen. Elk verhaal hierin kunnen we op zichzelf beschouwen: dit noemt Jacobson de Use Case slice. Met Use Case slices breek je de Use Case op in kleinere, onafhankelijk van elkaar opleverbare, delen. Ook biedt het de mogelijkheid om binnen een Use Case – waarmee je een set aan requirements oplevert – te kunnen prioriteren. Maar een Use Case is en blijft een groep van verhalen. Met een Scrum werkwijze kun je per sprint een of meerdere Use Case slices opleveren, net zolang tot alle slices van een Use Case opgeleverd zijn (zie afbeelding vorige pagina).

Het doel van een Use Case slice kun je op één kaartje kwijt, vergelijkbaar met een User Story. Het grote verschil is dat een User Story niets meer en minder is dan dat verhaaltje. Daarentegen is een Use Case slice onderdeel van een Use Case en deze is weer onderdeel van het Use Case model. Elke slice heeft zo zijn plaats in het grotere geheel. Dus het Use Case model is nog steeds nodig om het grote plaatje te begrijpen. Kies hieruit de Use Case die je wilt implementeren en deel deze op in slices om je iteraties te sturen. Zo werk je volgens Use Case 2.0.

Hierboven een afbeelding hoe je per Use Case deliverable de diepte in kunt gaan. Als je een Use Case in Scrum wilt gebruiken, kun je op het Kanban bord bij 'To do' de Use Case plakken, met de Slices die je in deze Sprint op wilt leveren. Deze slices schuiven dan gedurende de Sprint door naar 'In Progress' en 'Done'. Maar houd hier ook altijd Use Case Model en scope bij de hand, om vast te houden waar je naartoe werkt.

Zo biedt Ivar Jacobson met Use Case 2.0 het beste van twee werelden: lekker agile werken met Use Case slices en het overzicht over het grotere geheel houden dankzij het Use Case Model. Wij als redactie zijn benieuwd naar praktijkervaringen in het werken met Use Case 2.0. Neem contact met ons op als jij en je collega's hier over willen vertellen: dreamagazine@dreamevent.nl.