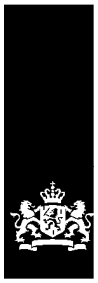

Ministerie van Binnenlandse Zaken en Koninkrijksrelaties

BRP-BZM Use Case Realisations Guidelines

Versie 2.0

02-09-2011
Definitief

Versiehistorie

Datum	Versie	Omschrijving	Auteur
23-12-2010	0.1	Eerste versie	R.F. Schaaf
04-01-2011	1.0	Feedback verwerkt R. Schaaf en D. Geluk rond structuur EA en veschil message / operatie	E. Lopes Cardozo
05-01-2011	1.1	Toelichting interface aangevuld	E. Lopes Cardozo
06-01-2011	1.2	• Naamgeving van KUCR's in EA aangepast • Tekst opgenomen over de wijze waarop delen van flow hergebruikt kunnen worden.	R. Froma
02-09-2011	2.0	Aangeboden aan stuurgroep	D. Geluk

Reviewhistorie

Datum	Versie	Omschrijving	Reviewers

Inhoudsopgave

1. Use Case Realisations Guidelines (keten niveau)

1.1 Algemeen

Stijlgids	Omschrijving
Meer dan diagrammen	Een Realisation bevat (sequence) diagrammen, maar is daar niet tot beperkt. Overweeg dus wat je nodig hebt; is het nodig om een model van de keten bij te leveren? Hoe zit het met (gebruikers- / systeem) interfaces? Vaak heb je meer nodig dan alleen diagrammen.
Neem het goede niveau	Een keten Use Case beschrijft de werking van de keten, niet de werking van de individuele systemen.

1.2 Additionele Documentatie

Stijlgids	Omschrijving
Definieer Keten	Maak een model van de hele keten; welke Actors zijn er voor de keten, welk systeem praat met welk ander systeem?
Definieer externe interfaces	Interfaces met actors dienen gedefinieerd te worden. Dit vereist meestal een apart interface-document. Zeker waar het gaat om externe partijen die een interactie hebben met de keten is het van belang om de interface goed duidelijk te hebben.
Definieer interne interfaces	Interfaces tussen systemen binnen de keten dienen op hoofdlijnen gedefinieerd te worden. Indien het werk uiteindelijk door meerdere partijen gedaan zal worden dan zal ook voor deze interfaces een detail uitwerking plaats moeten vinden (zoals bij externe interfaces), anders is het voldoende om dit te doen op louter logisch niveau. Hiermee wordt bedoeld dat wel degelijk moet worden aangegeven hoe een interface kan worden aangesproken, welke gegevens moeten / kunnen worden meegegeven en terug worden gestuurd. *Voorbeeld* - WEL: definiëren dat BSN wordt meegegeven - NIET: definiëren formaat BSN attribuut - NIET: definiëren verschijningsvorm (XML bericht met schema)

1.3 Sequence Diagrammen

Stijlgids	Omschrijving
Neem het goede niveau	Een keten Use Case beschrijft de werking van de keten, niet de werking van de individuele systemen. De elementen uit het sequence diagram zijn dus de (externe) Actors en de verschillende systemen in de keten. Componenten binnen de systemen worden dus niet opgenomen.
Neem alleen deelnemende elementen op	Beperk je tot de elementen die echt deelnemen aan de interactie, dus Actors die een rol vervullen en de systemen die in de interactie deelnemen.

Stijlgids	Omschrijving
Modelleer interactie tussen elementen	Alleen interactie tussen elementen wordt opgenomen. Dus de interactie tussen een actor en een systeem, de interactie tussen systeem A en systeem B, maar niet de interne acties die een systeem onderneemt. De interne functionaliteit is belangrijk, maar pas als je op systeem niveau bezig bent, niet op keten niveau. Een eventuele uitzondering hierop kan hierop gemaakt worden indien de "interne" functionaliteit expliciet beschreven is in de keten use case.
Elementen van een keten leven altijd	De levenslijnen van elk element van een sequence diagram op keten niveau begint bovenaan en loopt door tot het einde. Systemen worden niet halverwege ergens aangemaakt of verwijderd
Let op lengte activations	Een activation van een element duurt tot de hele interactie die hij start is afgelopen. Dus als systeem A een ander systeem, B, aanroept, dan loopt de activation binnen A tenminste tot het punt waar B zijn werk klaar heeft
Resultaten van synchrone aanroepen niet vermelden	In het sequence diagram staan de aanroepen van systemen, de waarden die terug worden gegeven zijn impliciet opgenomen door het einde van activation in het aangeroepen systeem. Er wordt aangenomen dat als je een systeem aanroept deze aan het einde iets terug geeft.
Resultaten van asynchrone aanroepen wel vermelden	Indien de aanroep asynchroon is, dus je roept een systeem aan maar wacht niet op antwoord, dan moet het feit dat het aangeroepen systeem een antwoord geeft ergens opgenomen worden. Dit kan zijn door een gestippelde pijl terug, of door een expliciete aanroep vanuit het andere systeem, maar het moet ergens terug komen.
Werk niet alle gevallen uit	Maak een sequence diagram voor het basis scenario (alles gaat goed) en vul dit aan met diagrammen voor de afwijkingen. In deze extra diagrammen kan gewoon verwezen worden naar het basis scenario (UML2 heeft hier mogelijkheden voor, maar een "note" werkt ook prima). Als je 10 gevallen hebt die ongeveer gelijk zijn werk er dan 1 uit. Het gaat om het begrip, niet om het in detail vastleggen van alles.
Kijk uit met fragments	Het gebruik van een fragment betekent vaak dat je 2 verschillende gevallen in 1 diagram aan het stoppen bent. Dat kan nuttig zijn, maar let er op dat het de leesbaarheid ten goede komt. Het gaat er om dat het systeem begrepen kan worden. Het gaat er niet om om zo veel mogelijk in 1 diagram te zetten. Loops zijn vaak niet te vermijden, maar zijn alt, opt, par, region echt nodig?
Beperk complexiteit	Tools als Enterprise Architect nodigen uit om veel complexiteit in diagrammen toe te voegen. Maar voegt het echt iets toe? Details toevoegen kost tijd, zowel bij het maken van het diagram als in het onderhoud. Vaak is een eenvoudige "note" al genoeg en kan vaak zaken beter beschrijven dan het diagram. Het gaat er om dat we weten hoe het moet werken, we gaan geen code genereren. Gebruik dus steeds die methode die het meeste begrip geeft bij de minste inspanning.
Maak de diagrammen leesbaar	Laat een buitenstaander de use case lezen en dan het diagram zien. Is het dan te begrijpen?

Stijlgids	Omschrijving
Maak alleen diagrammen die iets toevoegen	Vraag je steeds af "voegt dit diagram iets toe", "heeft iemand hier echt iets aan"? Als het antwoord "nee" is, waarom zou je het diagram dan maken? Een diagram dat gemaakt wordt moet ook onderhouden worden…
Verwijder diagrammen die niet (meer) relevant zijn	Als een diagram niet meer gebruikt wordt dient deze te worden verwijderd. Het zou immers alleen maar verwarring op kunnen leveren.
Enterprise Architect	Use case realisations op ketenniveau worden binnen EA ondergebracht binnen package *Use Case Realisations* binnen het Use Case Model. Verder geldt (zie ook configuration management plan): 1. Maak per Keten Use Case een use case specifiek package KUCRxxx - <naam use case> *Voorbeeld* KUCR001 – Aangegeven geboorte 2. Maak binnen een use case specifiek package voor iedere relevant use case module, flow of scenario een package KUCRxxx.<use case module id> - <naam flow> *Voorbeeld* KUCR001.1 – Basic Flow 3. Gebruik altijd objecten in sequence diagrammen en vermijd "links" 4. Wanneer op een gegeven moment dezelfde flow diverse malen terug komt in de diagrammen, overweeg dan om deze flow eenmaal apart te modelleren en deze vervolgens in de diagrammen waar deze flow voor komt op te nemen als link. Hiervoor kun je het (generieke) diagram slepen op de lifeline waar die start. Dat scheelt heel veel werk. Let wel op dat bij wijzigingen van een generieke flow gecontroleerd wordt of die wijziging van toepassing is op alle diagrammen waar die gebruikt wordt.
Messages	Met message wordt hier bedoeld een eerste indicatie of versie van een operatie. Streef naar zoveel mogelijk (her)gebruik van operaties op componenten. Maak inzichtelijk wanneer iets een message is. (toevoeging TEKST voor de message naam) Op het niveau van keten use case worden uiteindelijk alle messages omgezet naar operaties om zo de interfaces van systemen te formaliseren.