


1  
2 Feedback *it1 groep 8*  
3 =====  
4 17/20  
5 + presentatie is duidelijk moeite gedaan om het ontwerp uit te leggen,  
6 dat is goed.  
7 + Gebruik van klasse diagramma's was goed  
8 - Mix system sequence diagramma en sequence diagramma  
9 - SD hoeft niet altijd vanaf "begin" te beginnen, mag vanaf eender welke  
10 methode in eender welke klasse.  
11 *K* - Ui biedt geen "logout" aan, je kan dus geen twee usecases na elkaar  
12 uitvoeren als ze voor een verschillende soort user zijn.  
13 - Paar plaatsen waar consistency niet wordt afgedwongen:  
14 - Comment.addComment: circularity  
15 nb: Ik zou dit fixen in Comment.addComment en niet in zijn gebruikers, is  
16 veel robuuster. Dit is iets algemeen: voor package-visible methodes doe  
17 je de consistency-checks zo veel mogelijk in de methode zelf  
18 (defensief programmeren) ipv in de gebruikers. Voor public methodes  
19 heb je geen keuze en moeten alle checks in de methode zelf  
20 nb: ik zou de instantiatievariable "comments" hernoemen naar "children",  
21 veel duidelijker.  
22 *Tom* - BugReport.addAssignee: zou niet toegelaten mogen zijn om vb een "NotABug"  
23 bug ineens iemand te assignen.  
24 *Tri* - SubSystem.addBugReport: laat toe dat een BugReport in twee subsystems  
25 tegelijk is.  
26 *Tom* - BugReport.addAssignee heeft een koppeling naar New - dat is er maar  
27 1tje maar in de toekomst (it2 wss) zal dat wss snel meer worden,  
28 hint: hier is een design pattern voor.  
29 *Tom* - Role.assignmentPermission is wel erg een hack.  
30 Een nullable field voor een enum met 1 instantie dat op non-null  
31 gezet wordt in sommige subklassen... Dat kan beter.  
32 - DeveloperController.initializeUseCasesDeveloper:  
33 *K* Een methode opzoeken dmv reflection, is een beetje reflection abuse.  
34 De standaard manier om een lijst van zaken te houden, is een  
35 lijst van objecten, dat lijkt me properder. Heeft wel een beetje  
36 designaanpassing nodig, maar is denk ik geen slechte zaak om te doen.  
37 *L* - Op verdediging is gezegd dat ModelException een soort IllegalArgumentException  
38 is die checked moet zijn, maar bij assignDeveloperToBugReport  
39 wordt het gebruikt als "The user doesn't have the permission to add the developer."  
40 - "throws EenException" met documentatie "gebeurt eigenlijk nooit": laat dan  
41 *Tom* die methode geen exception throwen  
42 (in DeveloperAssignmentService.canUserAssignDeveloperToBugReport)  
43 *Tri* - TheDate.copy: throwt een ModelException wanneer TheDate in een inconsistente  
44 state is... zou geen concern voor de gebruiker van die methode mogen zijn!  
45 *L* - BugReport.addAssignee: specs zeggen niets over de tags. De methode  
46 breekt dus zijn contract ("alles waar niets over gezegd wordt blijft  
47 hetzelfde" wordt meestal niet expliciet in de specs geschreven)  
48 - Klasse moeten specs hebben die uitlegt hun verantwoordelijkheid is }  
49 - Tip: ipv een private isValid, en dan duplicatie in de specs van  
50 methoden die die isValid/canHaveAs gebruiken, is het praktischer  
51 *L* om de isValid public te maken zodat je er naar kan refereren van  
52 andere specs en geen specduplicatie te hebben  
53 *Tri* - Parser klasse: doet niets parsing, dus niet echt praktische naam.  
54 Beter: Formatter.  
55

 *Paasvariantie*

*TODO: elke maandag  
verslag doorsturen  
met verloop project*

21/03/16

## SWOP: week 7

opmerkingen bij evaluatie

25 presentatie (✓)

27 gebruik van class diagrams goed

sequence diagrams minder

SSD = man - systeem interactie

gewoon diagram moet niet beginnen bij het begin,  
beter voor diepgang (bvb. voor services)

211 UI heeft geen logout → TODO implementeren

213 consistentie wordt niet overal afgedwongen  
API moet dat voorzien

214 niet het geval bij • add comment  
→ beter werken met child-parent  
→ nu package visible  
→ zo veel mogelijk constraints in  
methode zelf, niet in gebruiker zelf.

222 • add assignee herbeijken iin tags

• add bugreport → consistentie

226 • add assignee heeft koppeling naar klasse "new"

→ moet verholpen worden = design pattern

• assignment Permission is te licht (fancy boolean)

OF lijst ipv  
variabele

→ het veld is zwaar

OF meer documentatie bij code

→ er is al dynamische binding → gebruik ze!

Algemeen in een superklasse mogen velden open blijven

→ mogelijk: "canDo"-methode = specs

als x dan OK  
anders exception



232 lijst in de Controller van use cases  
opgebouwd door reflection (niet zo goed)  
\* moet beter = objecten in lijst ipv methoden  
opt: klasse per use case (controller) (V)

237 ModelException van naam veranderen  
wordt door te veel methoden gebruikt  
~~ReportErrorToUserException~~

240 throws exception moet conform de documentatie zijn.  
(in the Doc) — exception moet in juiste methode  
behandeld ~~worden~~ worden  
mogelijk met printstacktrace en system exit  
evt. devlog?

245 programmering = addAssigner kan tag veranderen,  
maar specs vermelden er niets over  
als spec iets niet vermeldt, dan  
gebeurt er niets.

248 OOP-ding bij klasse doc van verantwoordelijkheden.  
invarianten documenteren indien deze kritiek voor programma's

249 bij checkers: als ze private gemaakt worden, problemen  
met specs  
→ beter gewoon public maken.

253 klasse Parser parseert niet.  
→ andere naam kiezen (Formatter of zo)

### Opmerkingen bij design

\* FSM beter niet doen omdat er nog andere factoren  
kunnen zijn buiten vorige tags

alternatief: tag verandering policy in TAS (tag assignment service)

\* Facade gebruiken voor delegatie  
voor ontwerpen van API

functies nodig voor UI verzamelen in Facade per package

store op bugreports, niet op patch individueel