

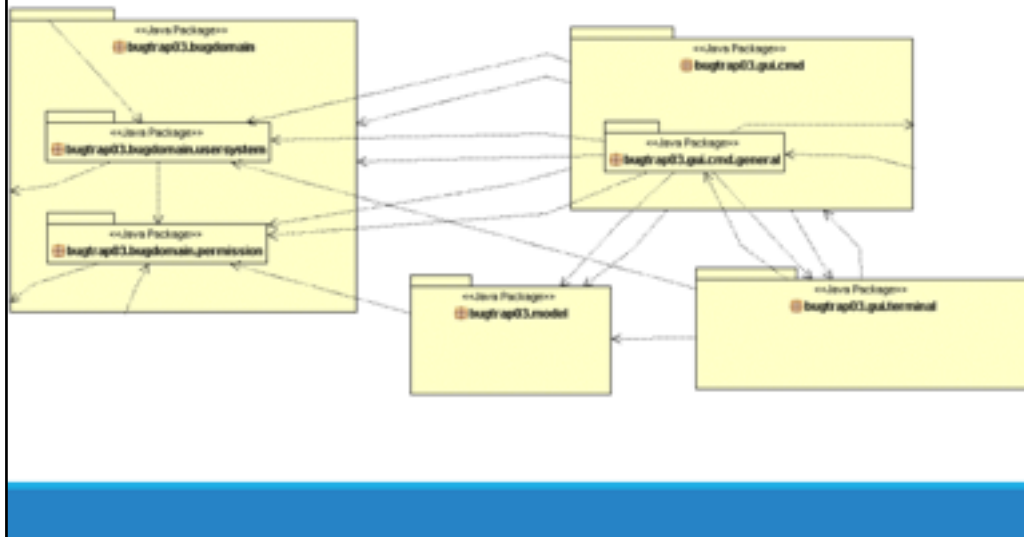
# SWOP Iteratie 1

## Group 03

---

M

## Overview Package level



M

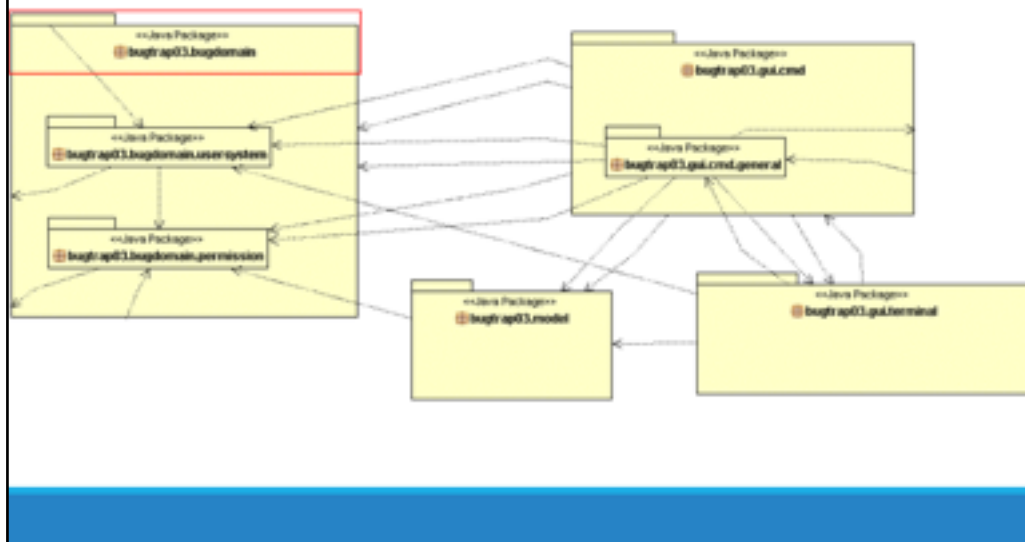
Notice: no dependencies from BugDomain to others.

Notice: No dependencies from Model to UI.

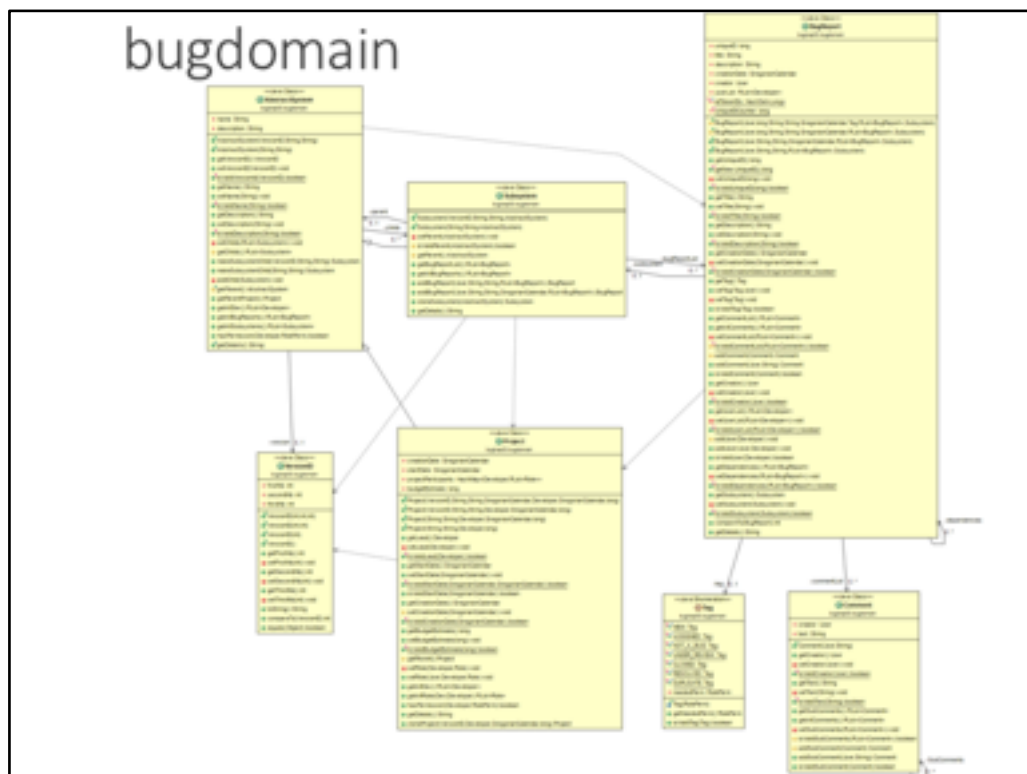
=> Low coupling



# Bug domain



K  
Next we will have a look at the bugdomain package.



K

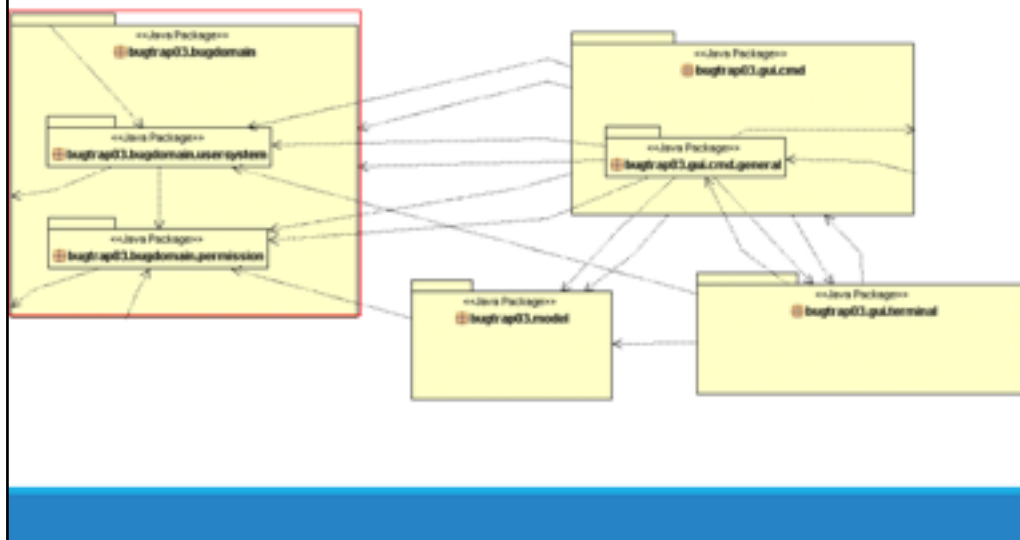
Project + Subsystems

- \* AbstractSystem (name, vID, descip, childs). Proj & Sub inherits.
- + Polymorhispm
- + High Cohesion
- High Coupling
  
- \* Project w attribute SubSystem.
- Code redundancy;
- High Coupling (Lower than first)
  
- \* Project & Subsystem as one class (Composite)
- + Low Coupling
- Low Cohesion (Can't differentiate between Proj & Sub. Code for Bugreports is redundant for Proj)

BugReport + Tags (Duplicate specifiek)

- \* Tag enum w logic inside the Tag enum.
- + Information Expert
- + High Cohesion
  
- \* Tags als objecten w logic inside the Tag
- + Polymorhispm -> separete the logic inside the Tags
- High coupling between Tag Objects
- + Option to add non-static variables.

## Project, BugReport, User & permissions



B



- High Coupling between 2 classes (A lot of coupling w permissions since User does not define methods for each permission, rather there are permissions for every action and it is checked if the user has this.)

- High Coupling between User & Permissions (Adding a permission requires an update for the user hierarchy to make sure certain users have the permissions and others do not.)

- \* User class + Enum userType + Using UserPermas as argument of userType

- Low Cohesion (Hard to enforce User type as parameter etc + instance of hierarchy not possible)

- High coupling

- No Polymorphism

- + Information Expert

- \* PermissionManager holding Permissions for each user with Permissions as Objects.

- High Coupling (need reference to this to check permissions)

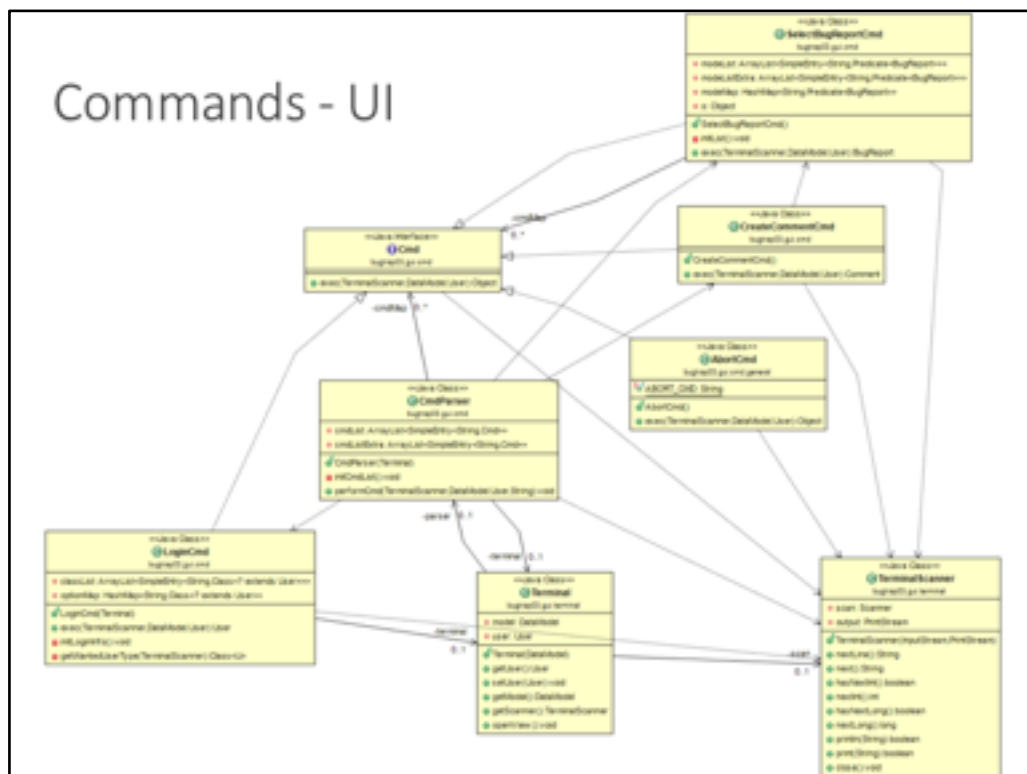
- + High Cohesion

- + Pure Fabrication ('outsource' responsibility)

- + Polymorphism (allow this; but isn't used)

- Polymorphism (Lot of overhead; Big hierarchy of permissions)





V

Explain Scenario's; Interaction from UI and How DataModel fits in.

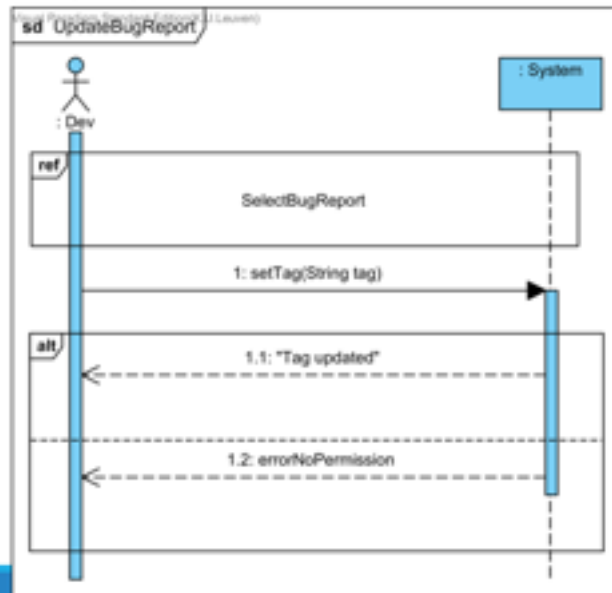
## UI - Model - Cmd

- \* Separate Use case steps into Cmds
- + Low coupling
- High cohesion
- + Controller
- + Polymorphism (little bit - Cmds)

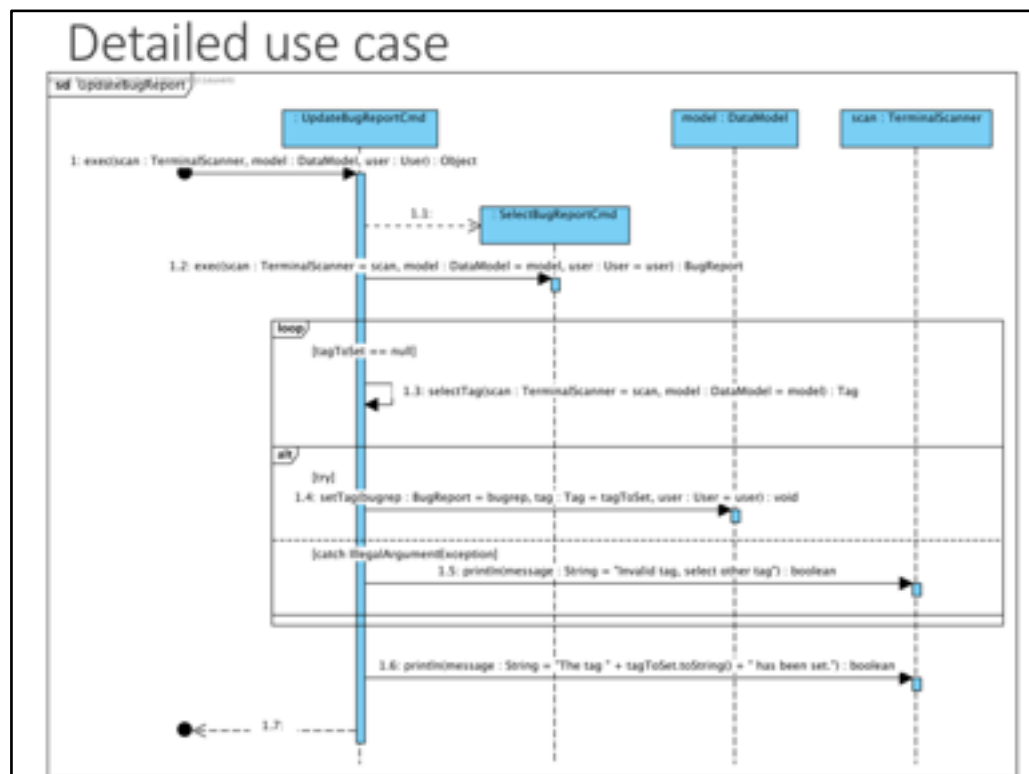
- \* Methods lots of methods
- + High coupling
- Low cohesion

## Use case

e.g UpdateBugReport

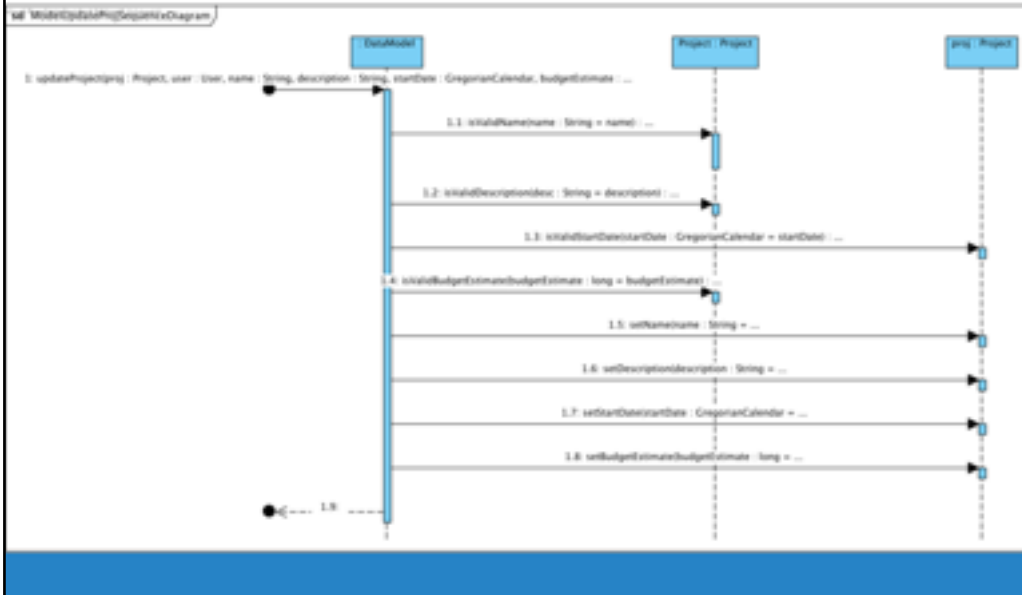


V  
System sequence diagram



V  
Sequence diagram















# DataModel UpdateProjectSequence



# Test Coverage

JaCoCoverage analysis of project "SWOP" (powered by JaCoCo from EcEmma)

## JaCoCoverage analysis of project "SWOP" (powered by JaCoCo from EcEmma)

Element	Missed Instructions	Cov	Missed Branches	Cov	Missed	Cov	Missed	Lines	Missed	Methods	Missed	Classes
bugtap01.bugdomain		98%		94%	17	269	7	515	1	138	0	7
bugtap01.bugdomain.permission		92%		n/a	4	9	0	24	4	9	0	3
bugtap01.bugdomain.usersystem		99%		92%	5	62	0	80	1	38	0	5
bugtap01.gui.cmd		98%		95%	10	156	14	480	1	54	0	14
bugtap01.gui.cmd.general		89%		76%	16	66	14	133	0	33	0	12
bugtap01.gui.terminal		67%		89%	2	25	16	63	1	16	0	2
bugtap01.model		97%		79%	7	53	6	112	0	36	0	1
Total	300 of 6,867	96%	56 of 632	91%	61	640	57	1457	8	324	0	44

K /

TODO? Tekstvak met uitleg Test strategy? (White box testing + mocking)

# TimeTable

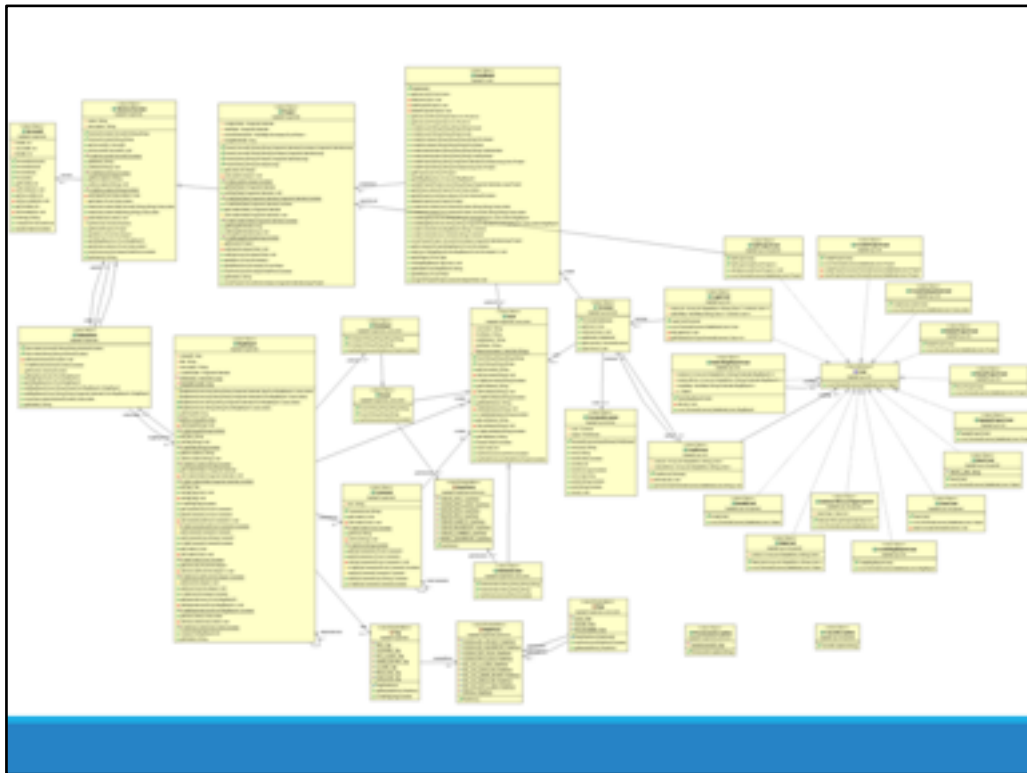
TODO Add Uurbestuding hier!!

---

Enkel totaal overzicht. NIET UITLEGGEN tijdens presentatie.

## Roles

- Design Coordinator - Derkinderen Vincent
- Testing Coordinator - Buytaert Kwinten
- Domain Coordinator - Dekempeneer Mathias



Hide – Backup/Just in case

Side Information GRASP:

High cohesion

Low coupling

Information expert

Don't talk to strangers

Polymorphism

Controller

Pure Fabrication (Not in conceptual model, specifically made for code)

Protected Variation (Assigning responsibility in a way that variations do not have undesirable effects. The way to build a stable 'interface' in a context of variations.)

Indirection (outsource responsibility)