PDDs and modeling patterns

Session 5
19 February 2019

Prof.dr. Sjaak Brinkkemper
Dr. Sietse Overbeek

Universiteit Utrecht

Agenda

- PDD creation
- Documenting PDDs
- Meta-modeling Patterns

Process-deliverable diagram

- The meta-data model and meta-process model are connected in the so-called Process-Deliverable Diagram, abbreviated PDD.

- Arrows link the deliverable to the activity it has been produced by

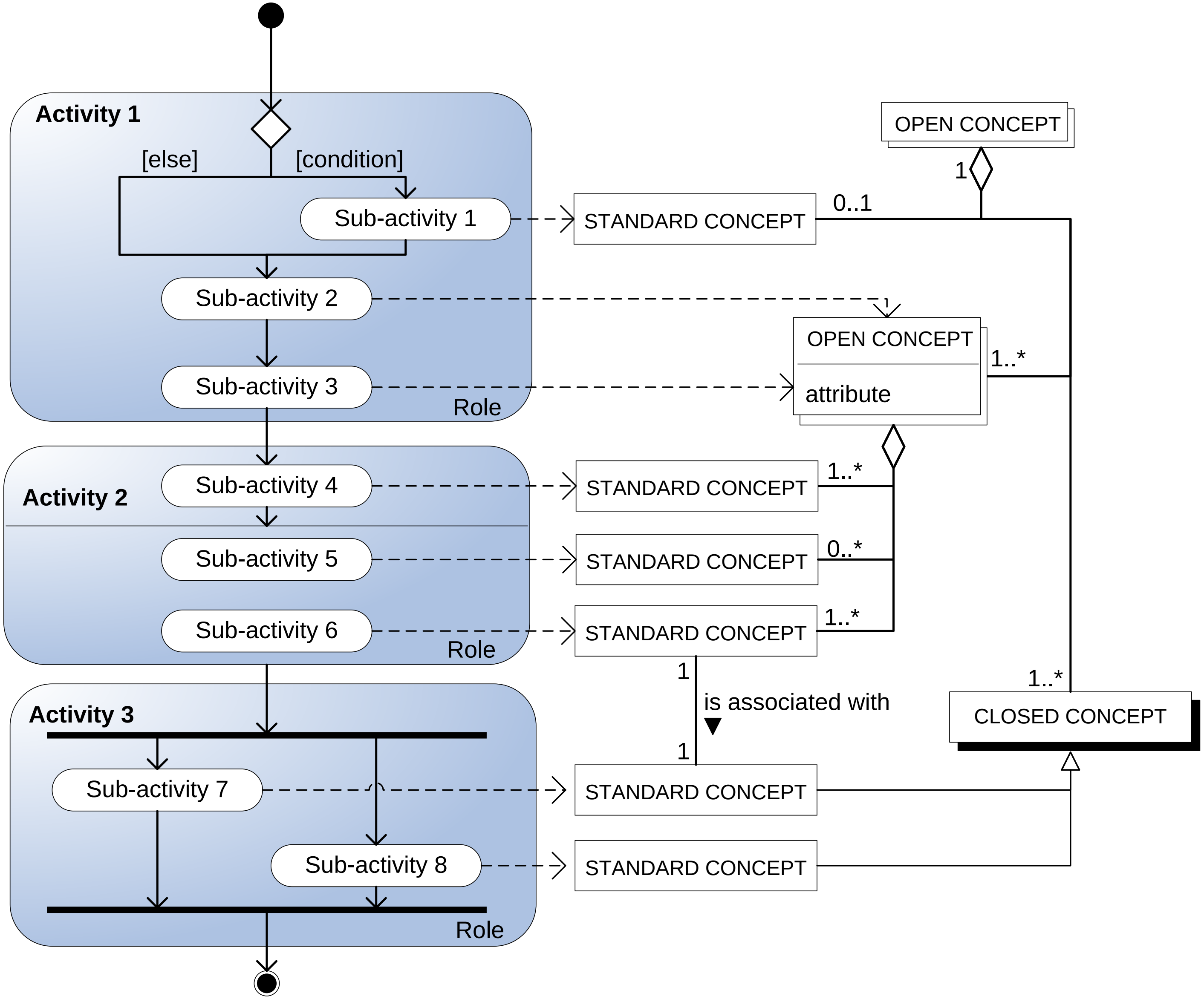
  ----------->

- Input arrows, e.g. which concepts are required by an activity are NOT shown, as all produced deliverables are assumed to be generally available

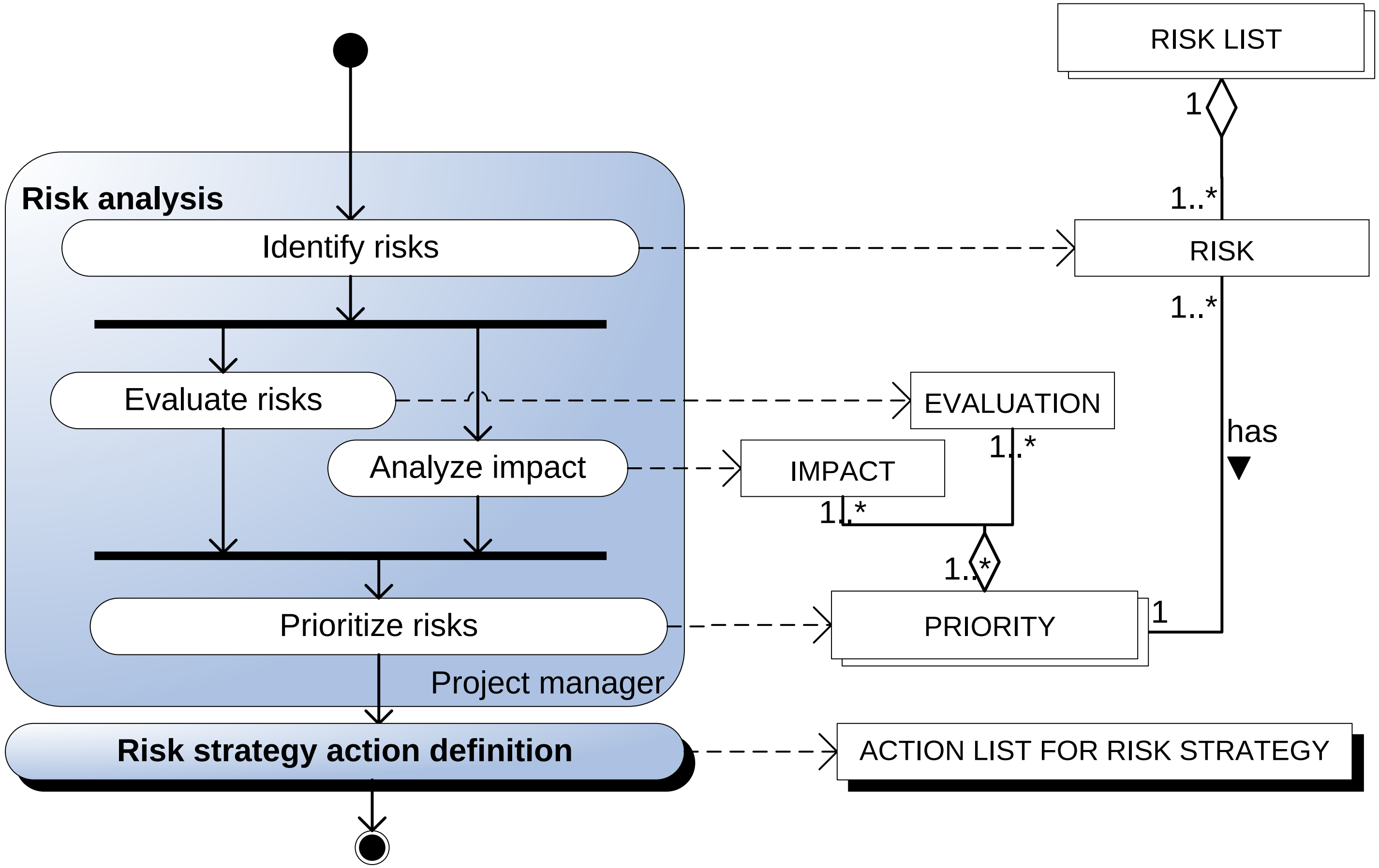
- Levels of complex activities and complex deliverables are aligned.

- See also: http://en.wikipedia.org/wiki/Process-data_diagram

Generic process-deliverable diagram



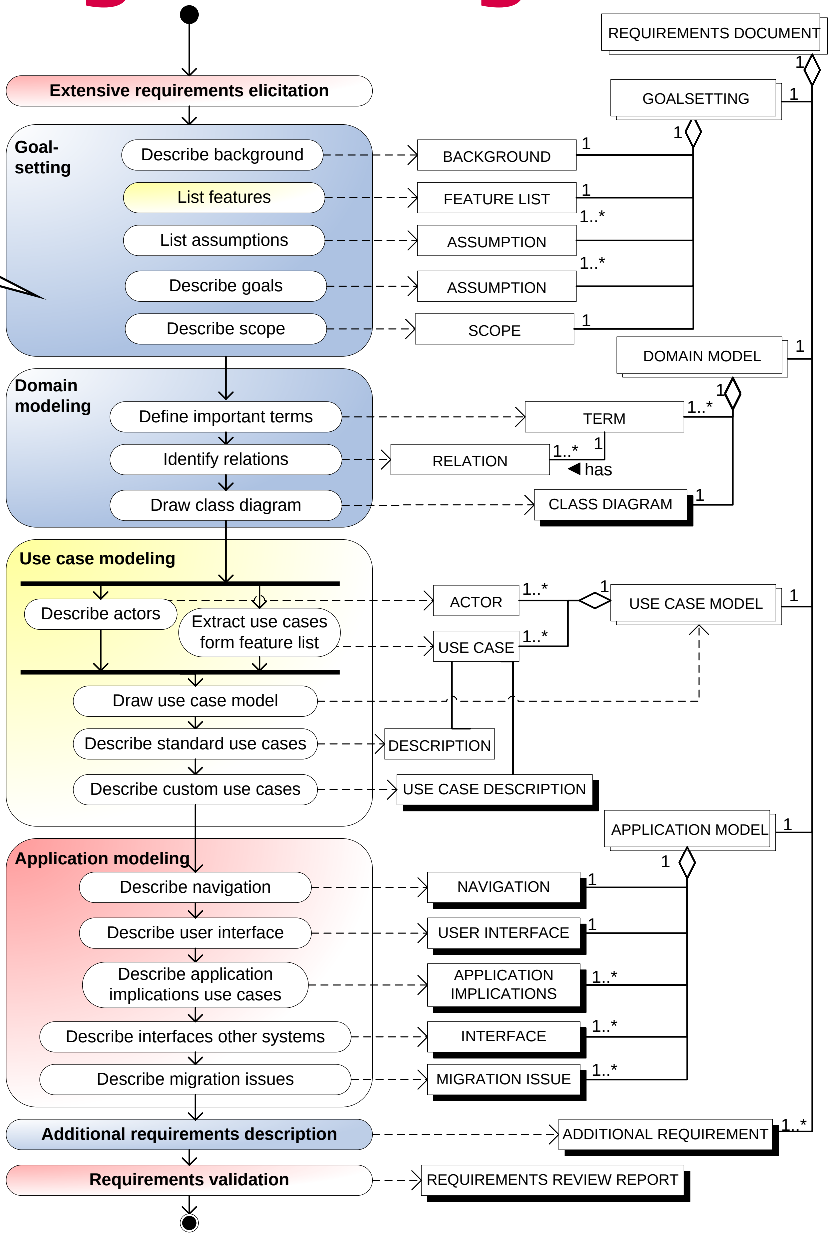
PDD example



Risk workflow in UML-Based Web Engineering

GX web engineering method

Use of color to show origin of the method fragments

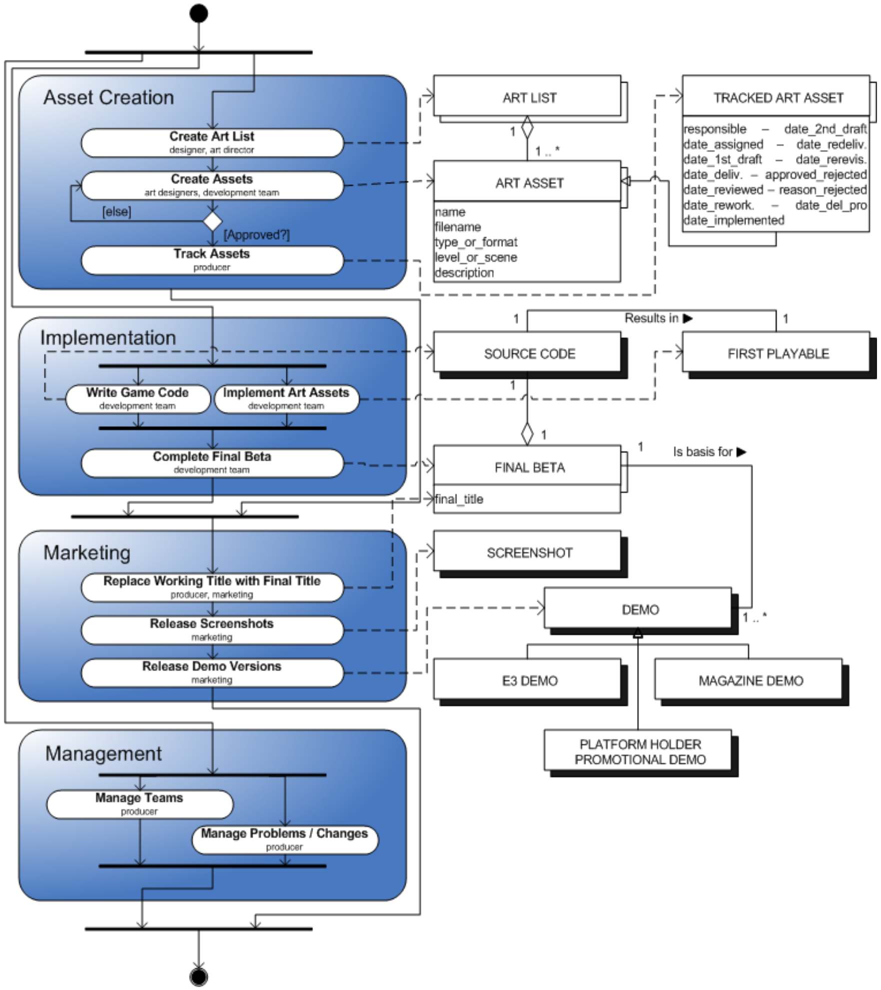


GX method

Unified Process

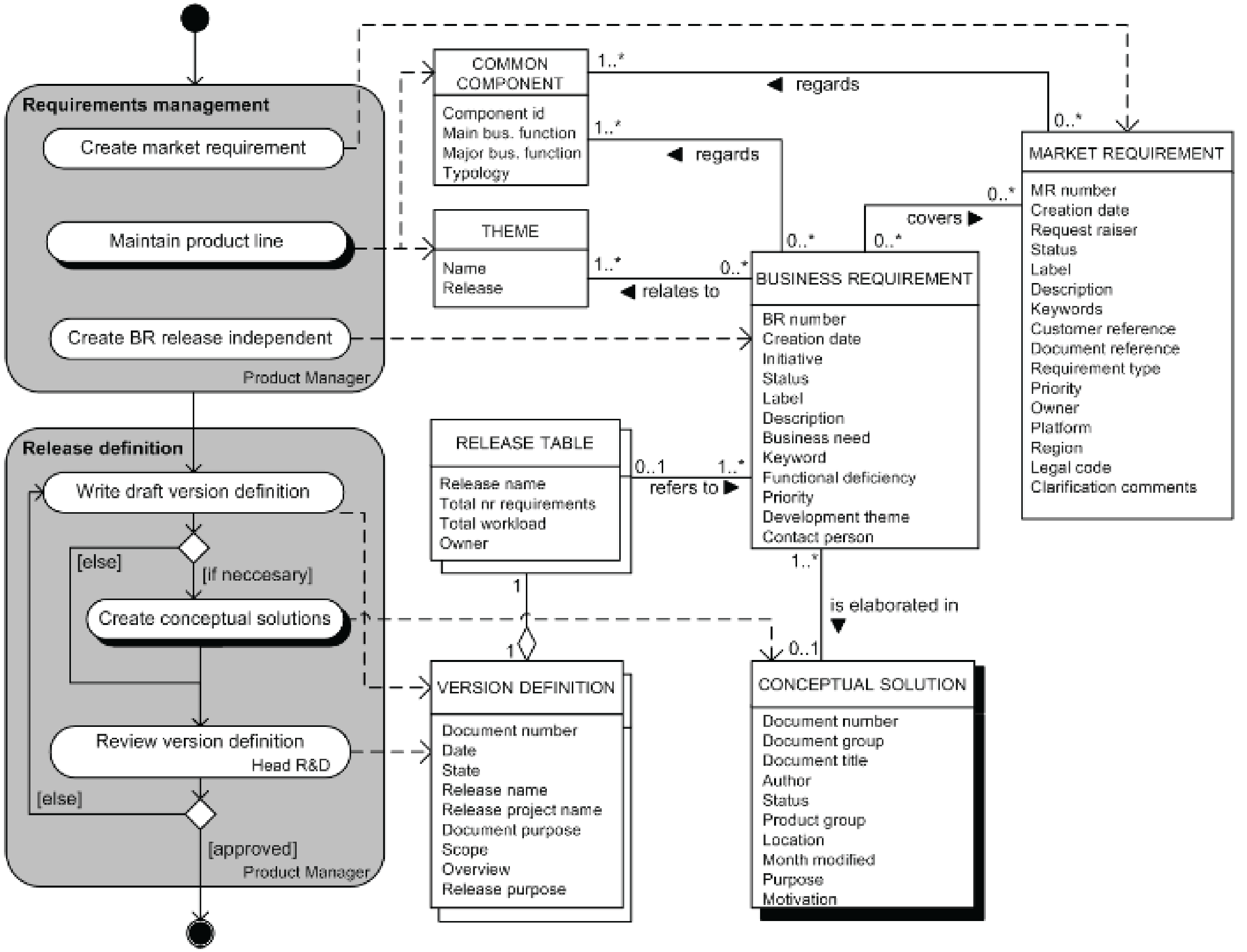
UML-based Web Engineering

PDD Game Production Method



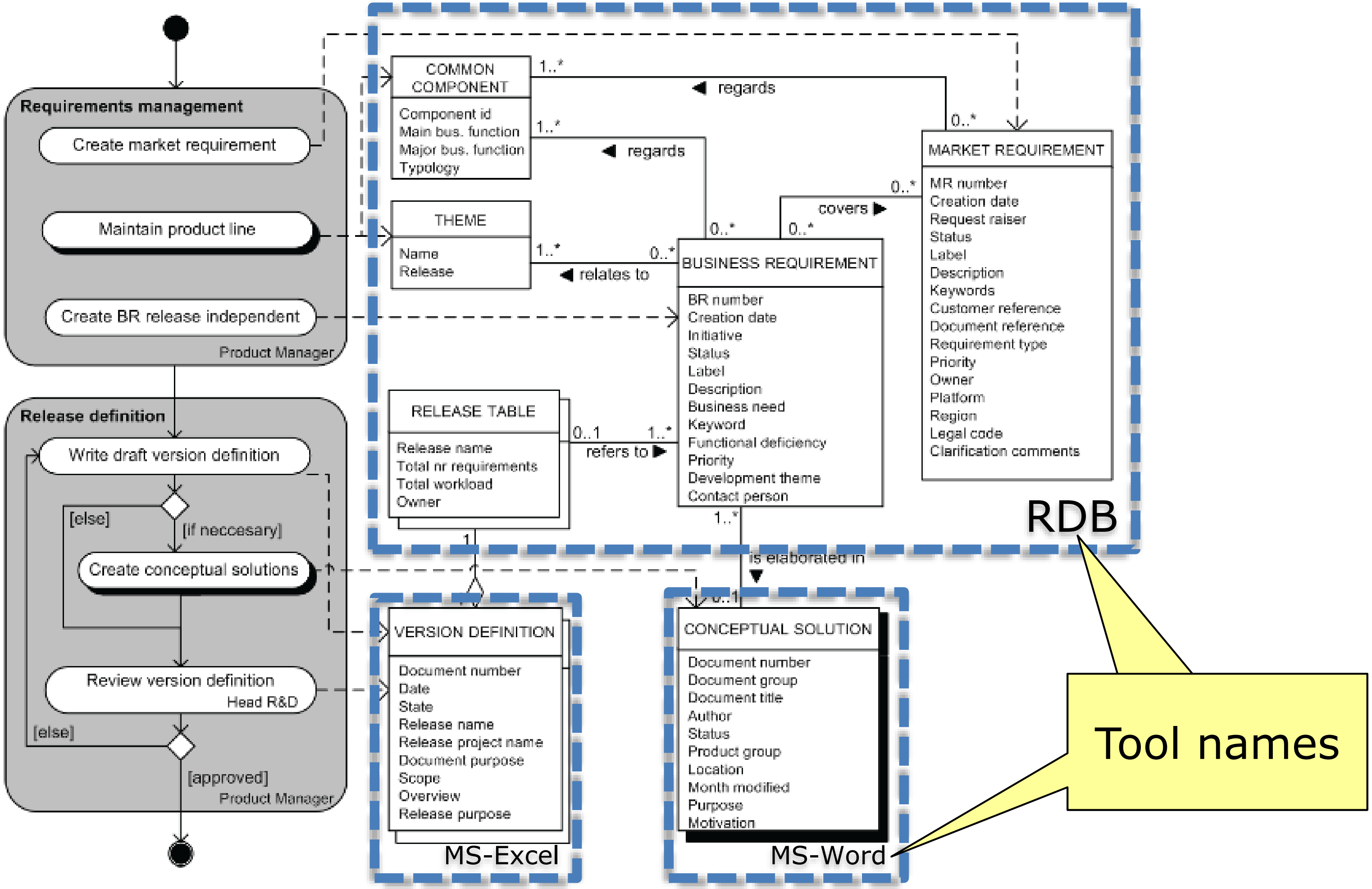
From: Stefan de Weerd: A Reference Method for Game Production
PDD of the method in the book: Introduction to Game Development

PDD Product Management



PDD of the requirements and release process at Baan

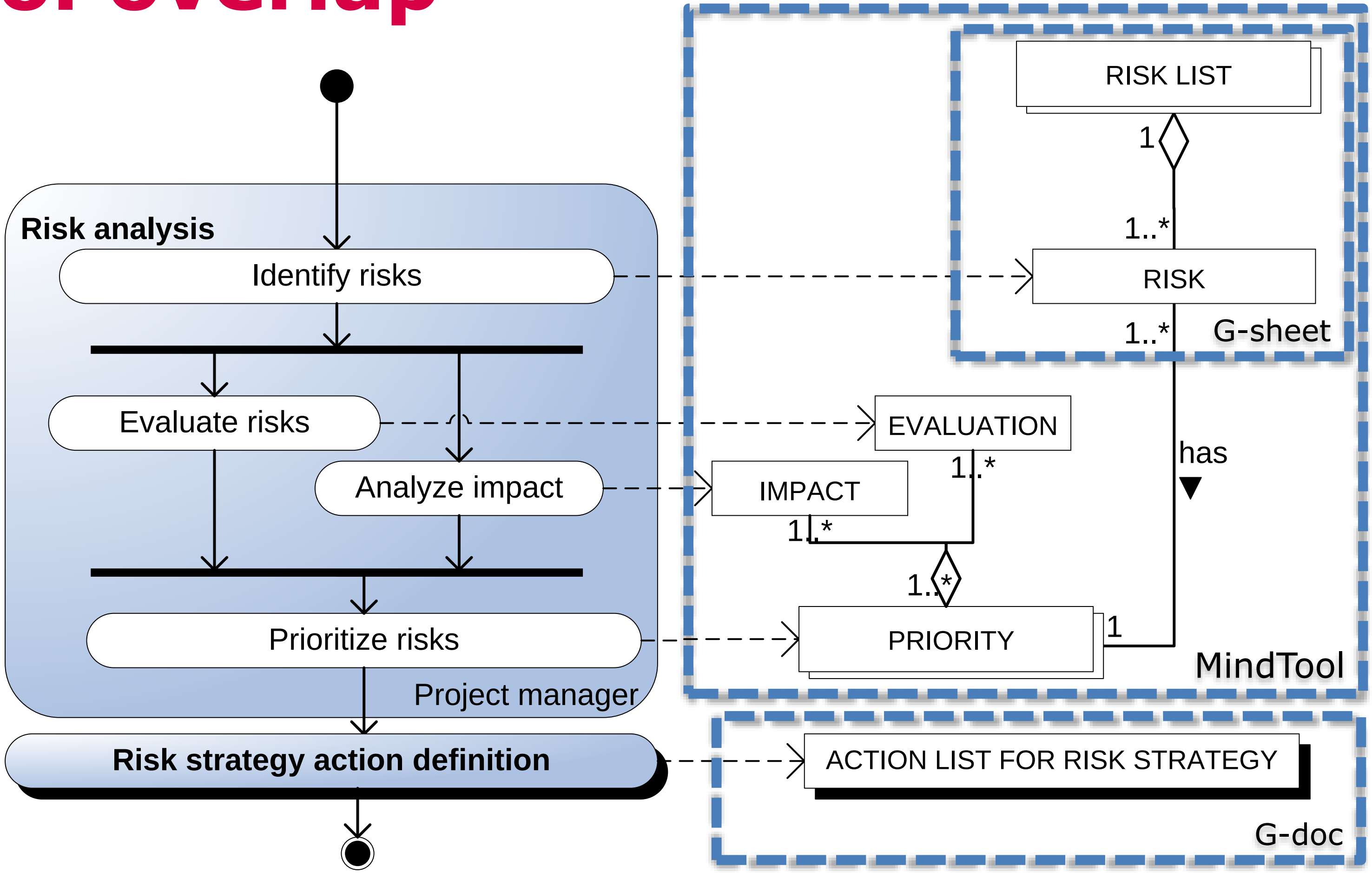
New to PDD: Tools overlay



Tool names

Support tools are listed on the deliverable side

Tool overlap



Tool support for UML-Based Web Engineering

Agenda

- PDD creation
- Documenting PDDs
- Meta-modeling Patterns

Documenting PDDs

- For a PDD an activity table has to be present providing descriptions of (sub-)activities and a concept table with definitions of the concepts.

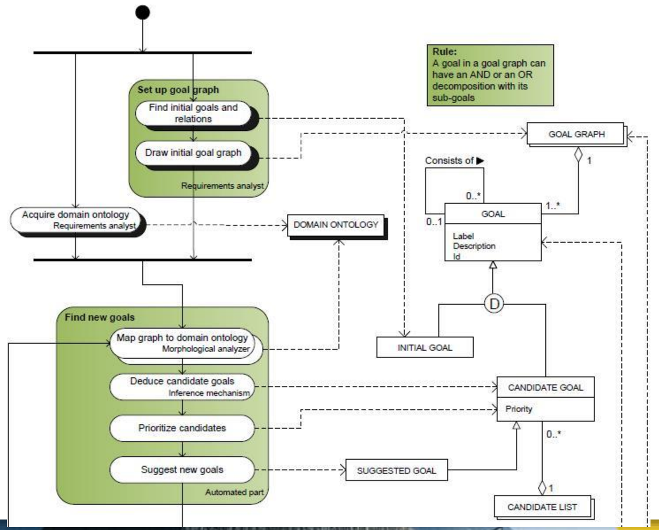
- In case for one study multiple PDDs are produced, then all are documented in one activity table and one concept table.

- Sometimes additional rules or unclarities are documented in an additional statement.

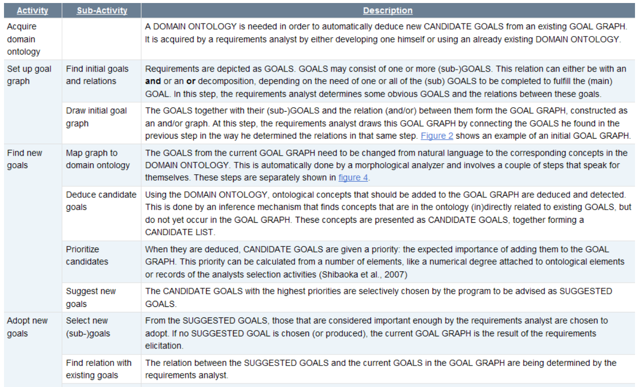
Activity and concept tables

- Example:
  Goal-Oriented And Ontology Driven Requirements Elicitation

process-side



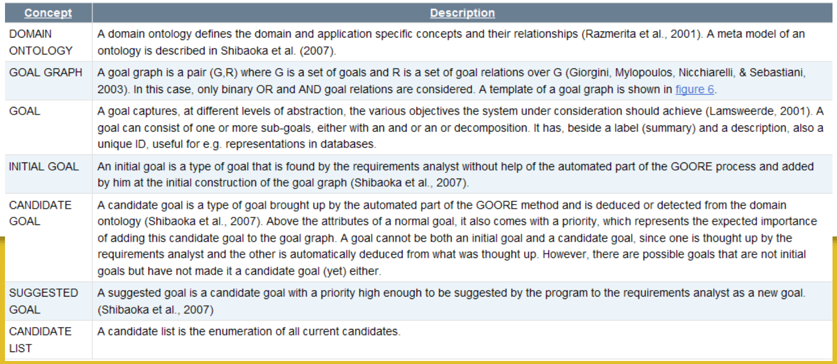
Activity table



Activity	Sub-Activity	Description
Acquire domain ontology		A DOMAIN ONTOLOGY is needed in order to automatically deduce new CANDIDATE GOALS from an existing GOAL GRAPH. It is acquired by a requirements analyst by either developing one himself or using an already existing DOMAIN ONTOLOGY.
Set up goal graph	Find initial goals and relations	Requirements are depicted as GOALS. GOALS may consist of one or more (sub-)GOALS. This relation can either be with an **and** or an **or** decomposition, depending on the need of one or all of the (sub) GOALS to be completed to fulfill the (main) GOAL. In this step, the requirements analyst determines some obvious GOALS and the relations between these goals.
	Draw initial goal graph	The GOALS together with their (sub-)GOALS and the relation (and/or) between them form the GOAL GRAPH, constructed as an and/or graph. At this step, the requirements analyst draws this GOAL GRAPH by connecting the GOALS he found in the previous step in the way he determined the relations in that same step. Figure 2 shows an example of an initial GOAL GRAPH.
Find new goals	Map graph to domain ontology	The GOALS from the current GOAL GRAPH need to be changed from natural language to the corresponding concepts in the DOMAIN ONTOLOGY. This is automatically done by a morphological analyzer and involves a couple of steps that speak for themselves. These steps are separately shown in figure 4.
	Deduce candidate goals	Using the DOMAIN ONTOLOGY, ontological concepts that should be added to the GOAL GRAPH are deduced and detected. This is done by an inference mechanism that finds concepts that are in the ontology (in)directly related to existing GOALS, but do not yet occur in the GOAL GRAPH. These concepts are presented as CANDIDATE GOALS, together forming a CANDIDATE LIST.
	Prioritize candidates	When they are deduced, CANDIDATE GOALS are given a priority: the expected importance of adding them to the GOAL GRAPH. This priority can be calculated from a number of elements, like a numerical degree attached to ontological elements or records of the analysts selection activities (Shibaoka et al., 2007)
	Suggest new goals	The CANDIDATE GOALS with the highest priorities are selectively chosen by the program to be advised as SUGGESTED GOALS.
Adopt new goals	Select new (sub-)goals	From the SUGGESTED GOALS, those that are considered important enough by the requirements analyst are chosen to adopt. If no SUGGESTED GOAL is chosen (or produced), the current GOAL GRAPH is the result of the requirements elicitation.
	Find relation with existing goals	The relation between the SUGGESTED GOALS and the current GOALS in the GOAL GRAPH are being determined by the requirements analyst.

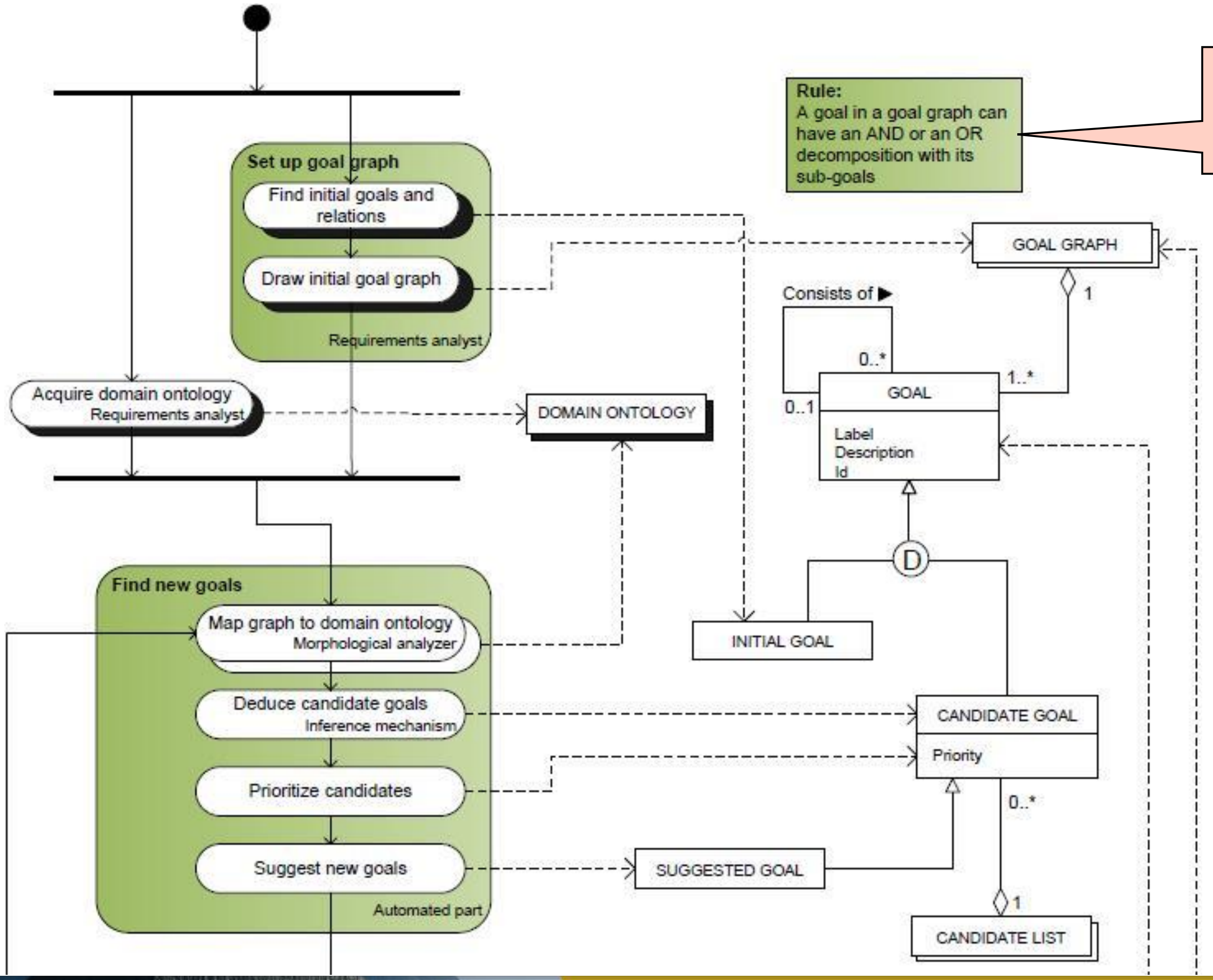
deliverable-side

Concept table



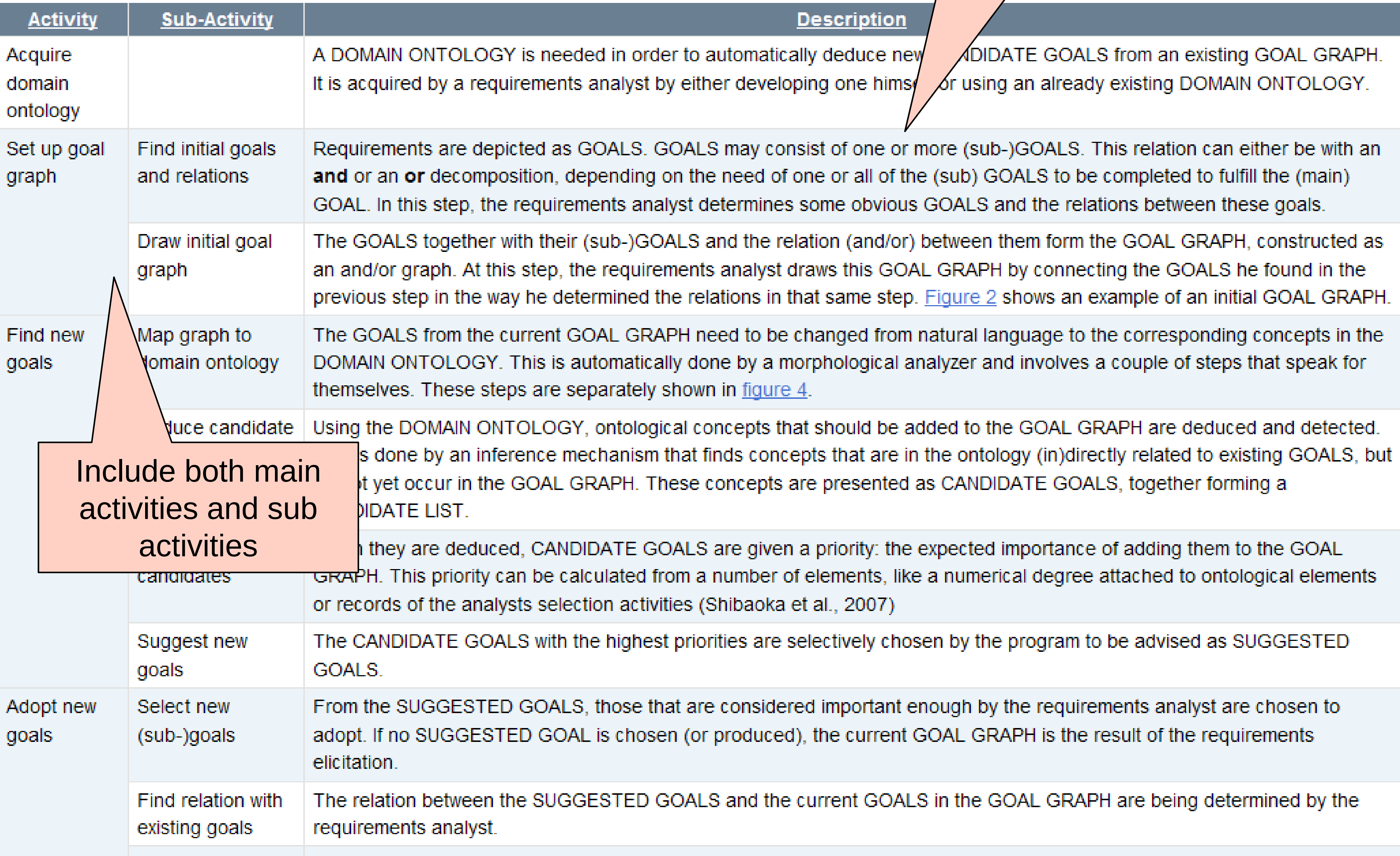
Concept	Description
DOMAIN ONTOLOGY	A domain ontology defines the domain and application specific concepts and their relationships (Razmerita et al., 2001). A meta model of an ontology is described in Shibaoka et al. (2007).
GOAL GRAPH	A goal graph is a pair (G,R) where G is a set of goals and R is a set of goal relations over G (Giorgini, Mylopoulos, Nicchiarelli, & Sebastiani, 2003). In this case, only binary OR and AND goal relations are considered. A template of a goal graph is shown in figure 6.
GOAL	A goal captures, at different levels of abstraction, the various objectives the system under consideration should achieve (Lamsweerde, 2001). A goal can consist of one or more sub-goals, either with an and or an or decomposition. It has, beside a label (summary) and a description, also a unique ID, useful for e.g. representations in databases.
INITIAL GOAL	An initial goal is a type of goal that is found by the requirements analyst without help of the automated part of the GOORE process and added by him at the initial construction of the goal graph (Shibaoka et al., 2007).
CANDIDATE GOAL	A candidate goal is a type of goal brought up by the automated part of the GOORE method and is deduced or detected from the domain ontology (Shibaoka et al., 2007). Above the attributes of a normal goal, it also comes with a priority, which represents the expected importance of adding this candidate goal to the goal graph. A goal cannot be both an initial goal and a candidate goal, since one is thought up by the requirements analyst and the other is automatically deduced from what was thought up. However, there are possible goals that are not initial goals but have not made it a candidate goal (yet) either.
SUGGESTED GOAL	A suggested goal is a candidate goal with a priority high enough to be suggested by the program to the requirements analyst as a new goal. (Shibaoka et al., 2007)
CANDIDATE LIST	A candidate list is the enumeration of all current candidates.

PDD



Additional rule

Activity table



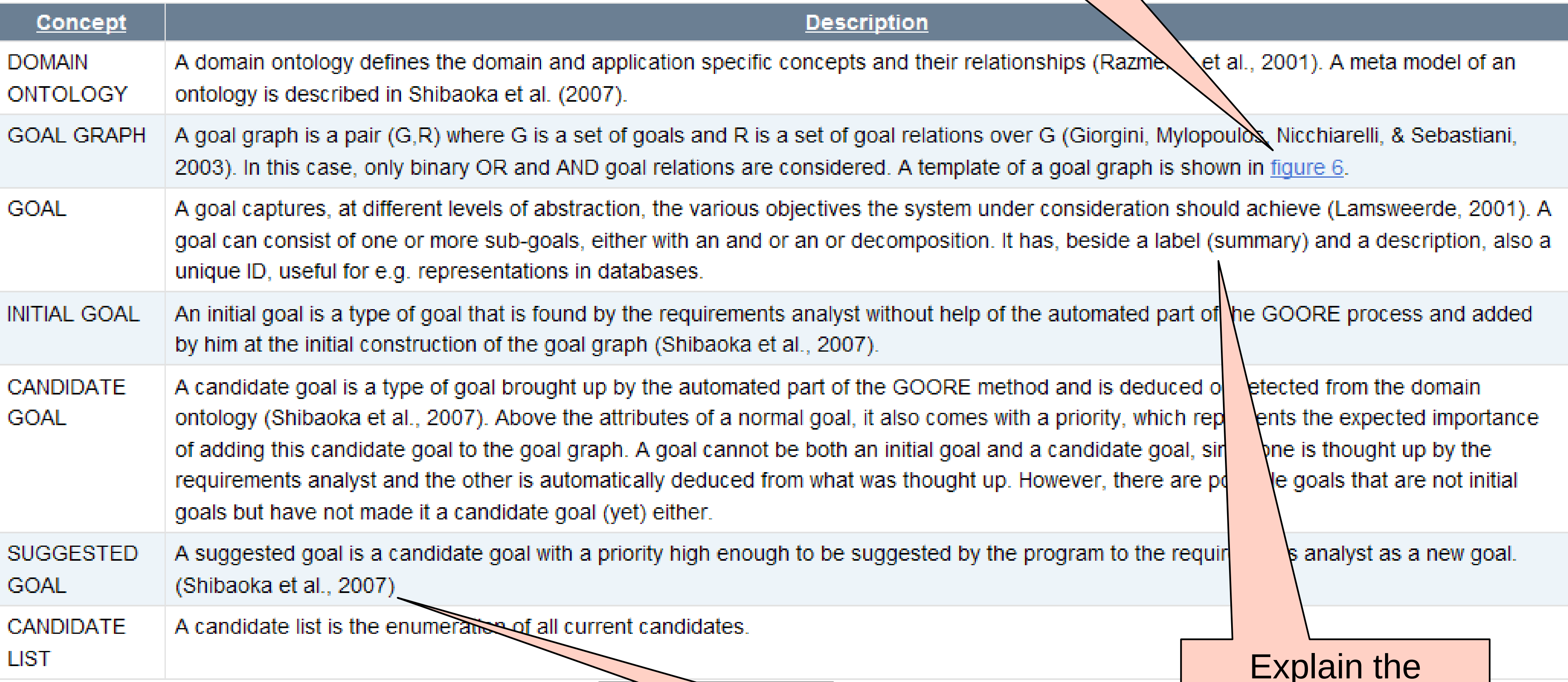
Activity	Sub-Activity	Description
Acquire domain ontology		A DOMAIN ONTOLOGY is needed in order to automatically deduce new NDIDATE GOALS from an existing GOAL GRAPH. It is acquired by a requirements analyst by either developing one himse or using an already existing DOMAIN ONTOLOGY.
Set up goal graph	Find initial goals and relations	Requirements are depicted as GOALS. GOALS may consist of one or more (sub-)GOALS. This relation can either be with an **and** or an **or** decomposition, depending on the need of one or all of the (sub) GOALS to be completed to fulfill the (main) GOAL. In this step, the requirements analyst determines some obvious GOALS and the relations between these goals.
	Draw initial goal graph	The GOALS together with their (sub-)GOALS and the relation (and/or) between them form the GOAL GRAPH, constructed as an and/or graph. At this step, the requirements analyst draws this GOAL GRAPH by connecting the GOALS he found in the previous step in the way he determined the relations in that same step. Figure 2 shows an example of an initial GOAL GRAPH.
Find new goals	Map graph to domain ontology	The GOALS from the current GOAL GRAPH need to be changed from natural language to the corresponding concepts in the DOMAIN ONTOLOGY. This is automatically done by a morphological analyzer and involves a couple of steps that speak for themselves. These steps are separately shown in figure 4.
	duce candidate	Using the DOMAIN ONTOLOGY, ontological concepts that should be added to the GOAL GRAPH are deduced and detected. s done by an inference mechanism that finds concepts that are in the ontology (in)directly related to existing GOALS, but t yet occur in the GOAL GRAPH. These concepts are presented as CANDIDATE GOALS, together forming a IDATE LIST.
	candidates	n they are deduced, CANDIDATE GOALS are given a priority: the expected importance of adding them to the GOAL GRAPH. This priority can be calculated from a number of elements, like a numerical degree attached to ontological elements or records of the analysts selection activities (Shibaoka et al., 2007)
	Suggest new goals	The CANDIDATE GOALS with the highest priorities are selectively chosen by the program to be advised as SUGGESTED GOALS.
Adopt new goals	Select new (sub-)goals	From the SUGGESTED GOALS, those that are considered important enough by the requirements analyst are chosen to adopt. If no SUGGESTED GOAL is chosen (or produced), the current GOAL GRAPH is the result of the requirements elicitation.
	Find relation with existing goals	The relation between the SUGGESTED GOALS and the current GOALS in the GOAL GRAPH are being determined by the requirements analyst.

Provide short but comprehensive descriptions

Include both main activities and sub activities

Concept table

Make cross-references to examples



Concept	Description
DOMAIN ONTOLOGY	A domain ontology defines the domain and application specific concepts and their relationships (Razmerita et al., 2001). A meta model of an ontology is described in Shibaoka et al. (2007).
GOAL GRAPH	A goal graph is a pair (G,R) where G is a set of goals and R is a set of goal relations over G (Giorgini, Mylopoulos, Nicchiarelli, & Sebastiani, 2003). In this case, only binary OR and AND goal relations are considered. A template of a goal graph is shown in figure 6.
GOAL	A goal captures, at different levels of abstraction, the various objectives the system under consideration should achieve (Lamsweerde, 2001). A goal can consist of one or more sub-goals, either with an and or an or decomposition. It has, beside a label (summary) and a description, also a unique ID, useful for e.g. representations in databases.
INITIAL GOAL	An initial goal is a type of goal that is found by the requirements analyst without help of the automated part of the GOORE process and added by him at the initial construction of the goal graph (Shibaoka et al., 2007).
CANDIDATE GOAL	A candidate goal is a type of goal brought up by the automated part of the GOORE method and is deduced or detected from the domain ontology (Shibaoka et al., 2007). Above the attributes of a normal goal, it also comes with a priority, which represents the expected importance of adding this candidate goal to the goal graph. A goal cannot be both an initial goal and a candidate goal, since one is thought up by the requirements analyst and the other is automatically deduced from what was thought up. However, there are possible goals that are not initial goals but have not made it a candidate goal (yet) either.
SUGGESTED GOAL	A suggested goal is a candidate goal with a priority high enough to be suggested by the program to the requirements analyst as a new goal. (Shibaoka et al., 2007)
CANDIDATE LIST	A candidate list is the enumeration of all current candidates.

Include references

Explain the concept's properties

Including additional rules

- Additional rules that cannot be depicted graphically are described next to the meta-data model

- Unclarities and open issues are also described annex to the meta-data model.

Rule:

1. No splitting and joining of FLOWS
2. Naming of FLOWS is optional

Issues:

a. Different types of FLOWS: information, financial, goods
b. Relationship of Goods FLOWS on Supply Chain Diagram to Enterprise Function Level is unclear

Listing of tools

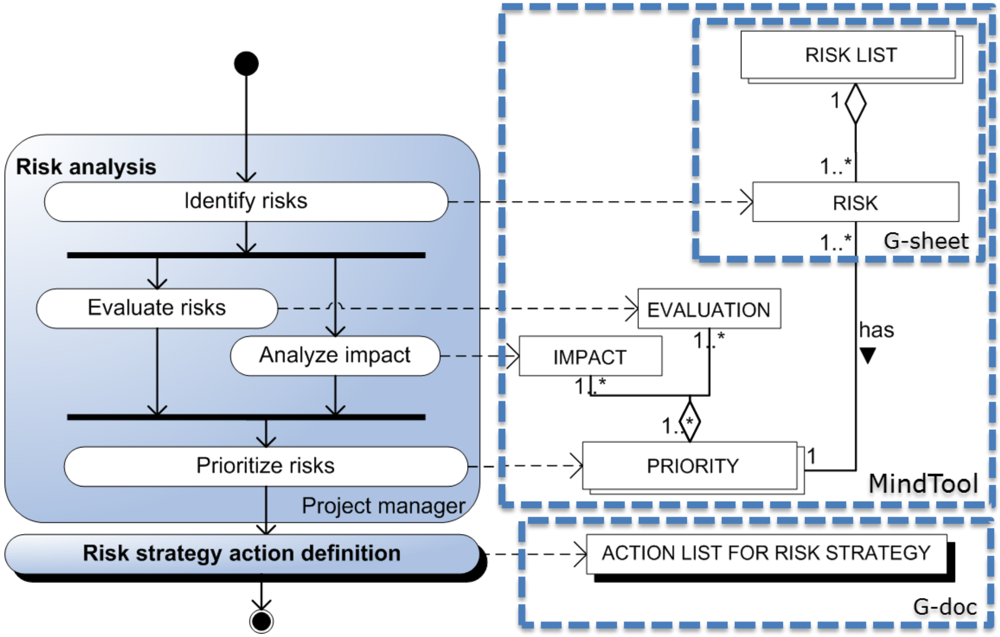
Tools are to be listed in a separate paragraph

Nr.	Tool	Concept
1	MS-Word	Test plan
2	Jira	Issue
3	Github	Software code
…		

- Comments on the quality of support tools can be placed in this paragraph
  - Lack of tools; non-commercial prototypes
  - Overlap in tool support

Another example

Nr	Tool	Concept
1	Google-sheet	RISK LIST RISK
2	MindTool	RISK LIST RISK PRIORITY IMPACT EVALUATION
3	Google-doc	ACTION LIST FOR RISK STRATEGY
…		

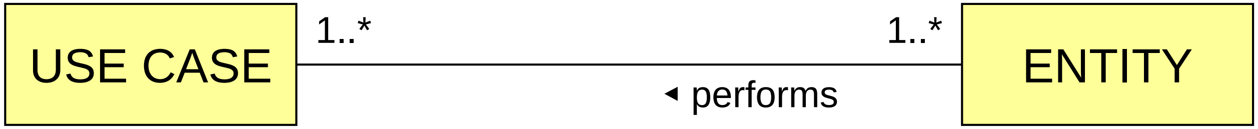


Agenda

- PDD creation
- Documenting PDDs
- Meta-modeling Patterns

Two concept matrix

Entity Use Case	Order	Chemical	Requester	Vendor Catalog
Place Order	√	√	√	√
Change Order	√	-	√	√
Manage Chemical Inventory	-	√	-	-
Report on Orders	√	√	√	-
Edit Requesters	-	-	√	-



USE CASE

1..*

◂ performs

1..*

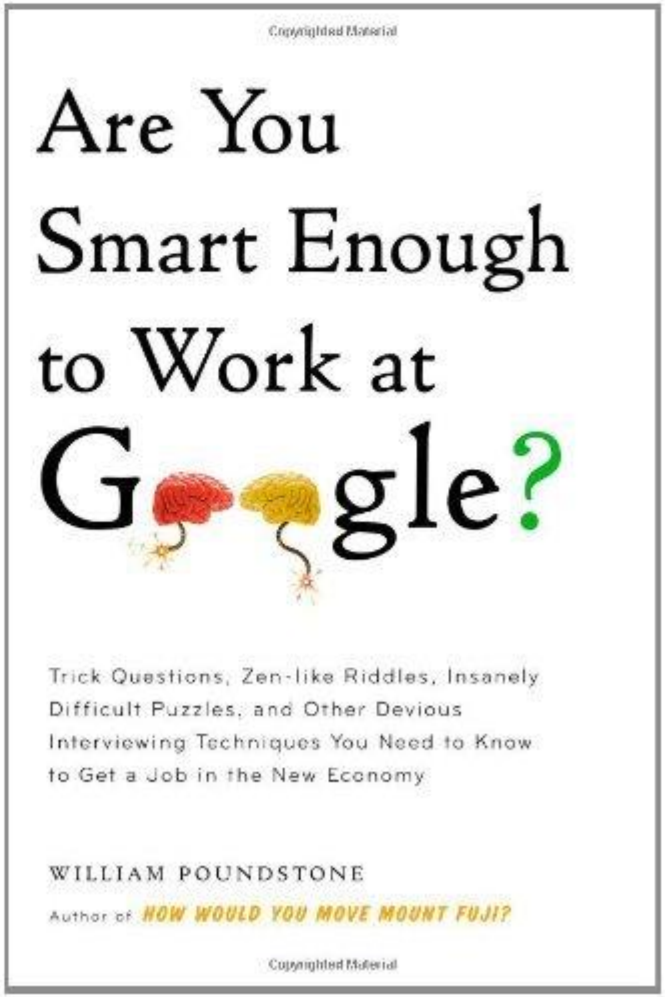
ENTITY

Example taken from: Karl Wiegers, Software Requirements, Microsoft Press

Break

Imagine a country where all the parents want to have a boy. Every family keeps having children until they have a boy; then they stop.

What is the proportion of boys to girls in this country?



Copyrighted Material

Are You Smart Enough to Work at Google?

Trick Questions, Zen-like Riddles, Insanely Difficult Puzzles, and Other Devious Interviewing Techniques You Need to Know to Get a Job in the New Economy

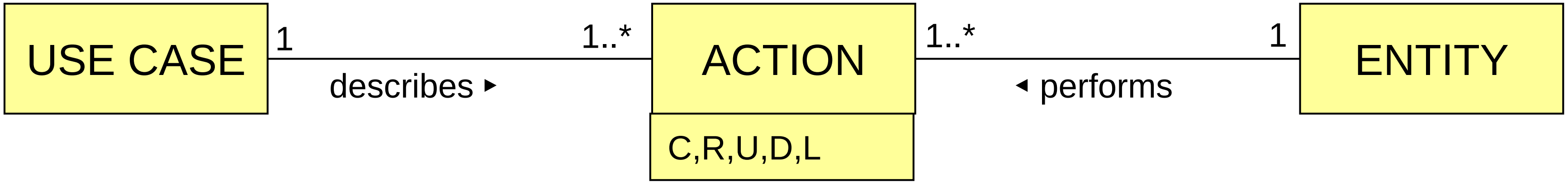
WILLIAM POUNDSTONE

Author of *HOW WOULD YOU MOVE MOUNT FUJI?*

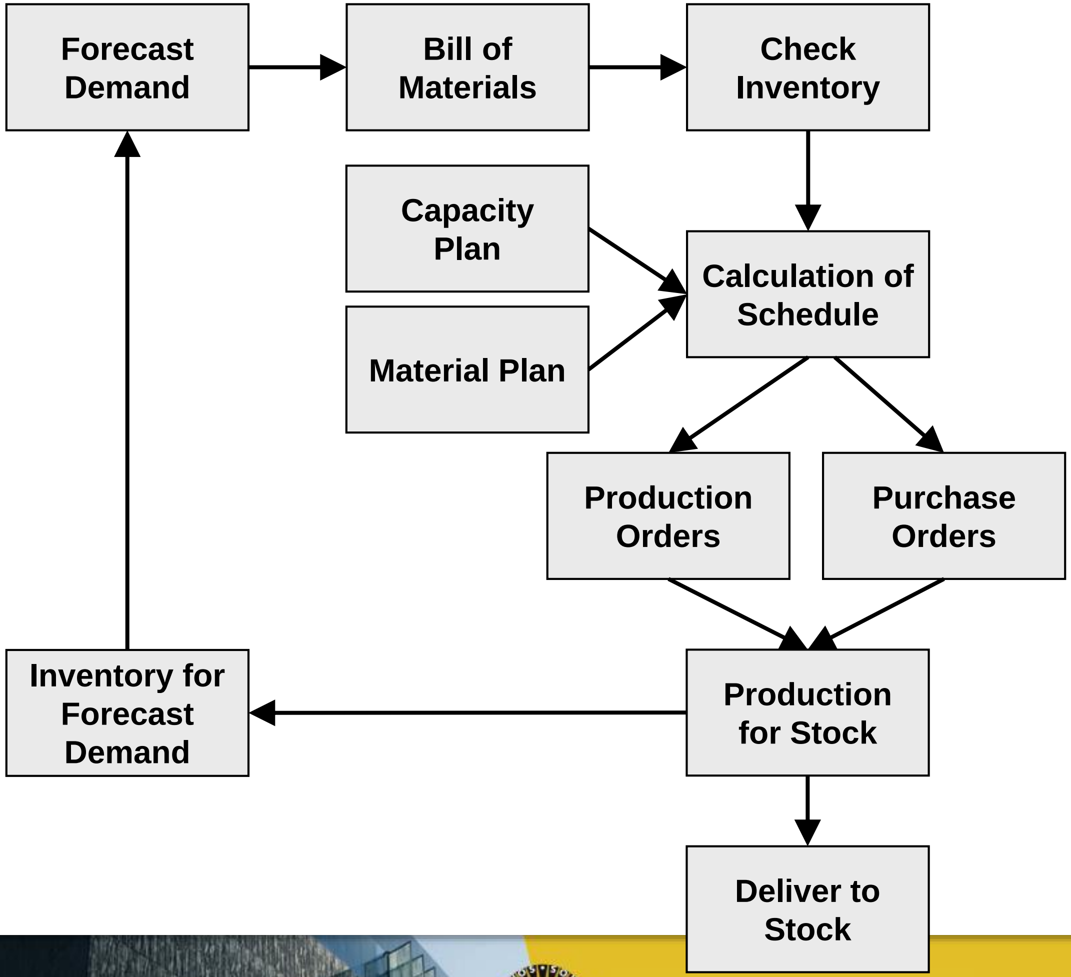
Copyrighted Material

Two concept matrix

Entity Use Case	Order	Chemical	Requester	Vendor Catalog
Place Order	C	R	R	R,L
Change Order	U, D		R	R,L
Manage Chemical Inventory		C,U,D		
Report on Orders	R	R,L	R,L	
Edit Requesters			C,U,L	



One concept diagram



Forecast Demand

Bill of Materials

Check Inventory

Capacity Plan

Material Plan

Calculation of Schedule

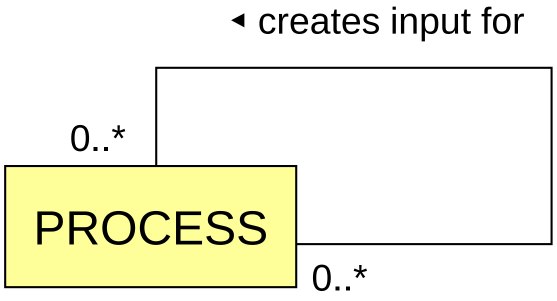
Production Orders

Purchase Orders

Production for Stock

Inventory for Forecast Demand

Deliver to Stock



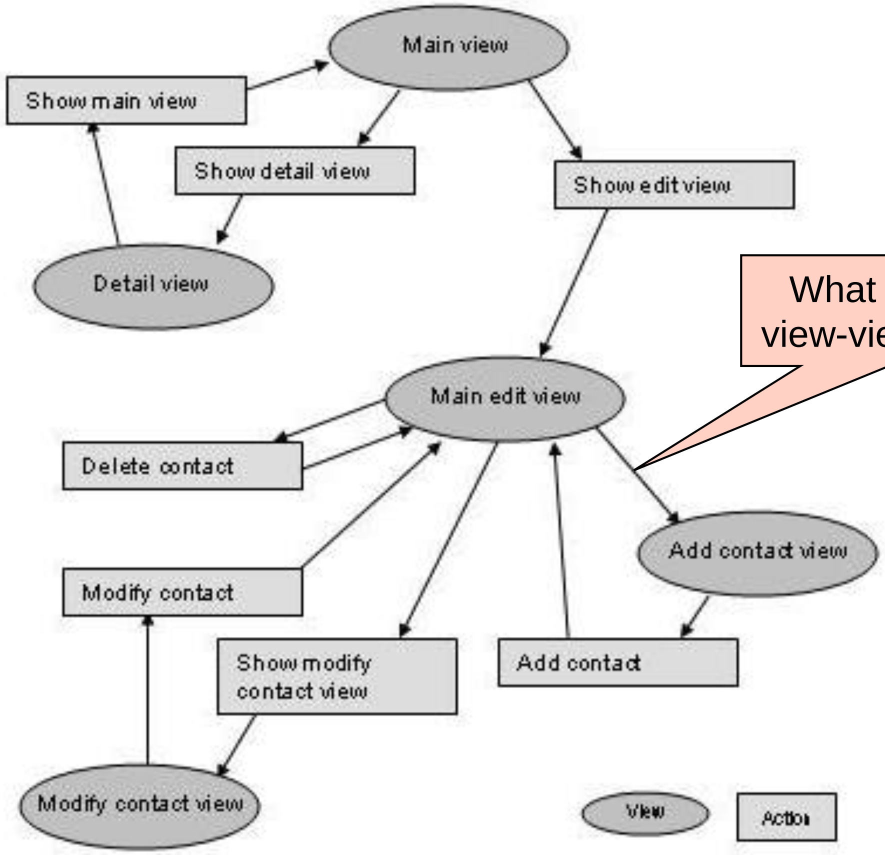
◂ creates input for

0..*

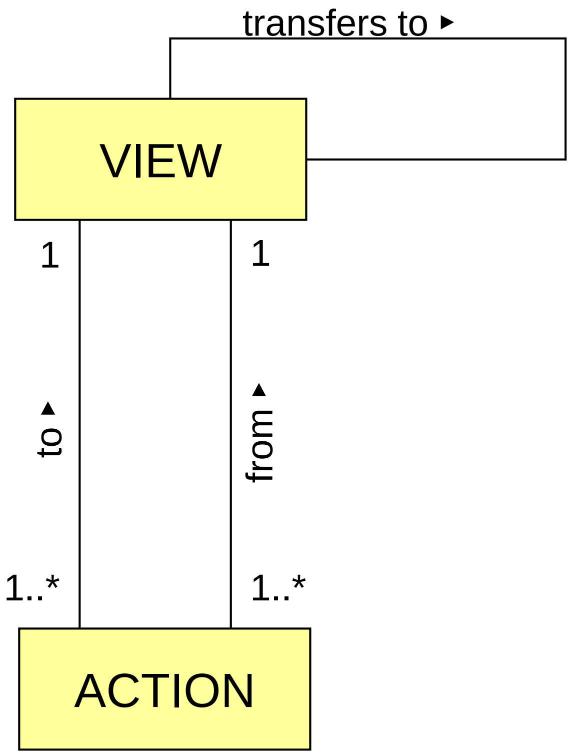
PROCESS

0..*

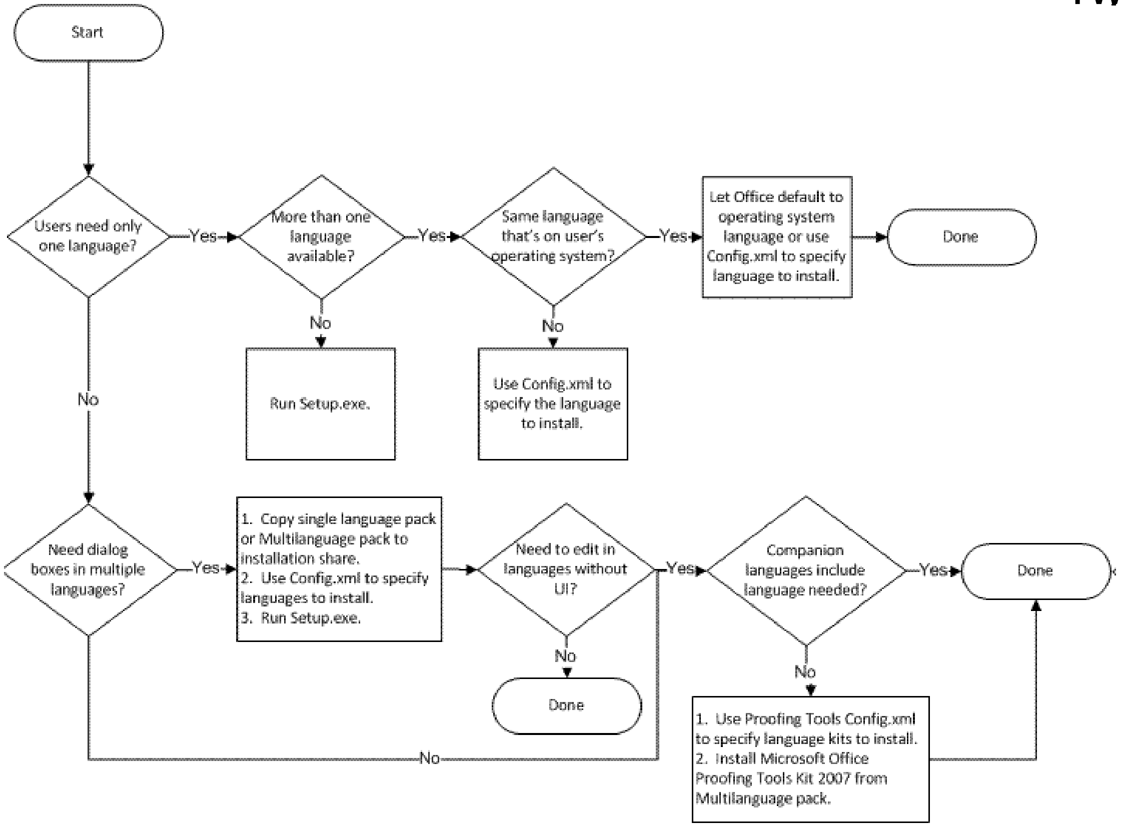
Two concept diagram



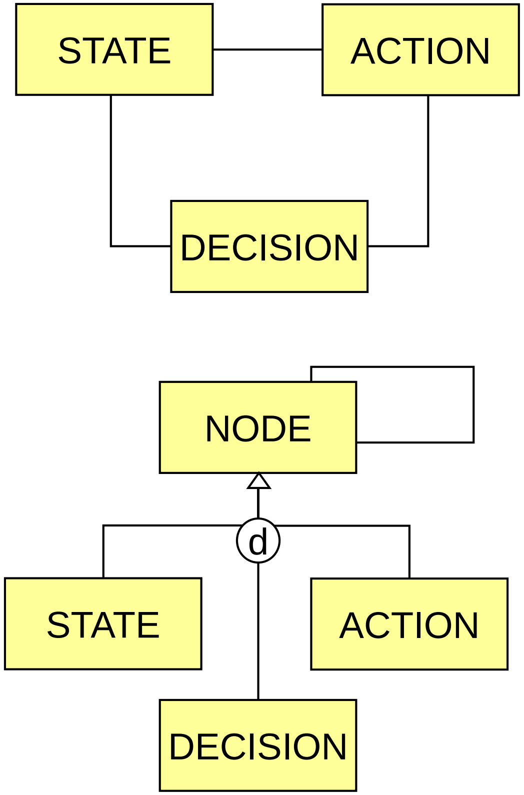
What about this view-view relation?



Multi-concept diagram


Start
Users need only one language?
Yes
More than one language available?
Yes
Same language that's on user's operating system?
Yes
Let Office default to operating system language or use Config.xml to specify language to install.
Done
No
Run Setup.exe.
No
Use Config.xml to specify the language to install.
No
Need dialog boxes in multiple languages?
Yes
1. Copy single language pack or Multilanguage pack to installation share.
2. Use Config.xml to specify languages to install.
3. Run Setup.exe.
Need to edit in languages without UI?
Yes
Companion languages include language needed?
Yes
Done
No
Done
No
1. Use Proofing Tools Config.xml to specify language kits to install.
2. Install Microsoft Office Proofing Tools Kit 2007 from Multilanguage pack.
No

Two possible meta-models:

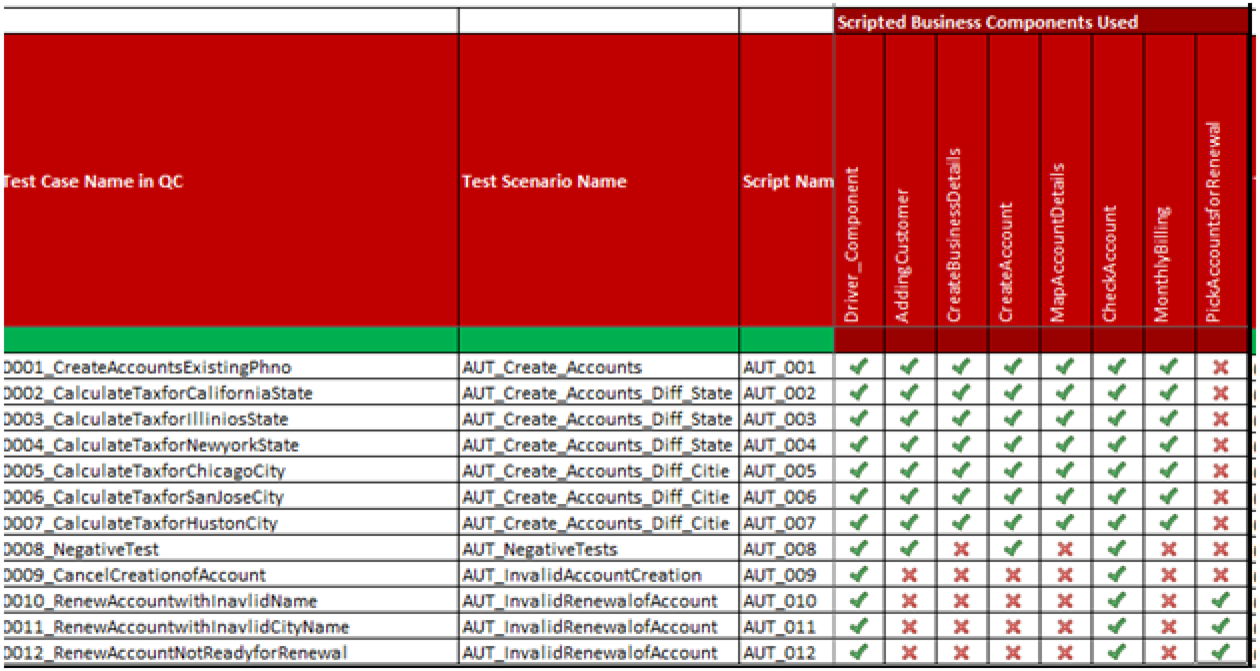


Choice depends on relations with other concepts in the domain

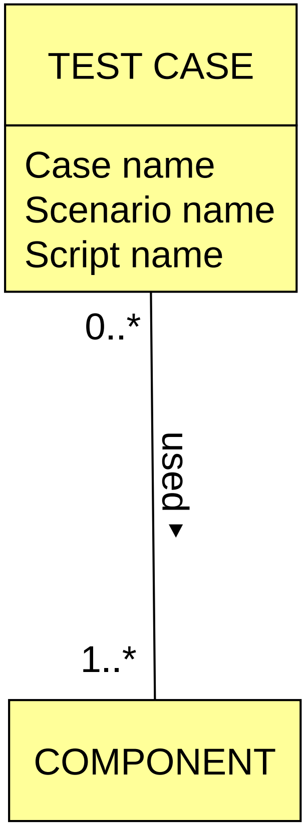
Example from: Microsoft Office Technet

Tables

Test Coverage sheet



			Scripted Business Components Used							
Test Case Name in QC	Test Scenario Name	Script Nam	Driver_Component	AddingCustomer	CreateBusinessDetails	CreateAccount	MapAccountDetails	CheckAccount	MonthlyBilling	PickAccountsforRenewal
0001_CreateAccountsExistingPhno	AUT_Create_Accounts	AUT_001	✓	✓	✓	✓	✓	✓	✓	✗
0002_CalculateTaxforCaliforniaState	AUT_Create_Accounts_Diff_State	AUT_002	✓	✓	✓	✓	✓	✓	✓	✗
0003_CalculateTaxforIlliniosState	AUT_Create_Accounts_Diff_State	AUT_003	✓	✓	✓	✓	✓	✓	✓	✗
0004_CalculateTaxforNewyorkState	AUT_Create_Accounts_Diff_State	AUT_004	✓	✓	✓	✓	✓	✓	✓	✗
0005_CalculateTaxforChicagoCity	AUT_Create_Accounts_Diff_Citie	AUT_005	✓	✓	✓	✓	✓	✓	✓	✗
0006_CalculateTaxforSanJoseCity	AUT_Create_Accounts_Diff_Citie	AUT_006	✓	✓	✓	✓	✓	✓	✓	✗
0007_CalculateTaxforHustonCity	AUT_Create_Accounts_Diff_Citie	AUT_007	✓	✓	✓	✓	✓	✓	✓	✗
0008_NegativeTest	AUT_NegativeTests	AUT_008	✓	✓	✗	✓	✗	✓	✗	✗
0009_CancelCreationofAccount	AUT_InvalidAccountCreation	AUT_009	✓	✗	✗	✗	✗	✓	✗	✗
0010_RenewAccountwithInavlidName	AUT_InvalidRenewalofAccount	AUT_010	✓	✗	✗	✗	✗	✓	✗	✓
0011_RenewAccountwithInavlidCityName	AUT_InvalidRenewalofAccount	AUT_011	✓	✗	✗	✗	✗	✓	✗	✓
0012_RenewAccountNotReadyforRenewal	AUT_InvalidRenewalofAccount	AUT_012	✓	✗	✗	✗	✗	✓	✗	✓



TEST CASE

Case name
Scenario name
Script name

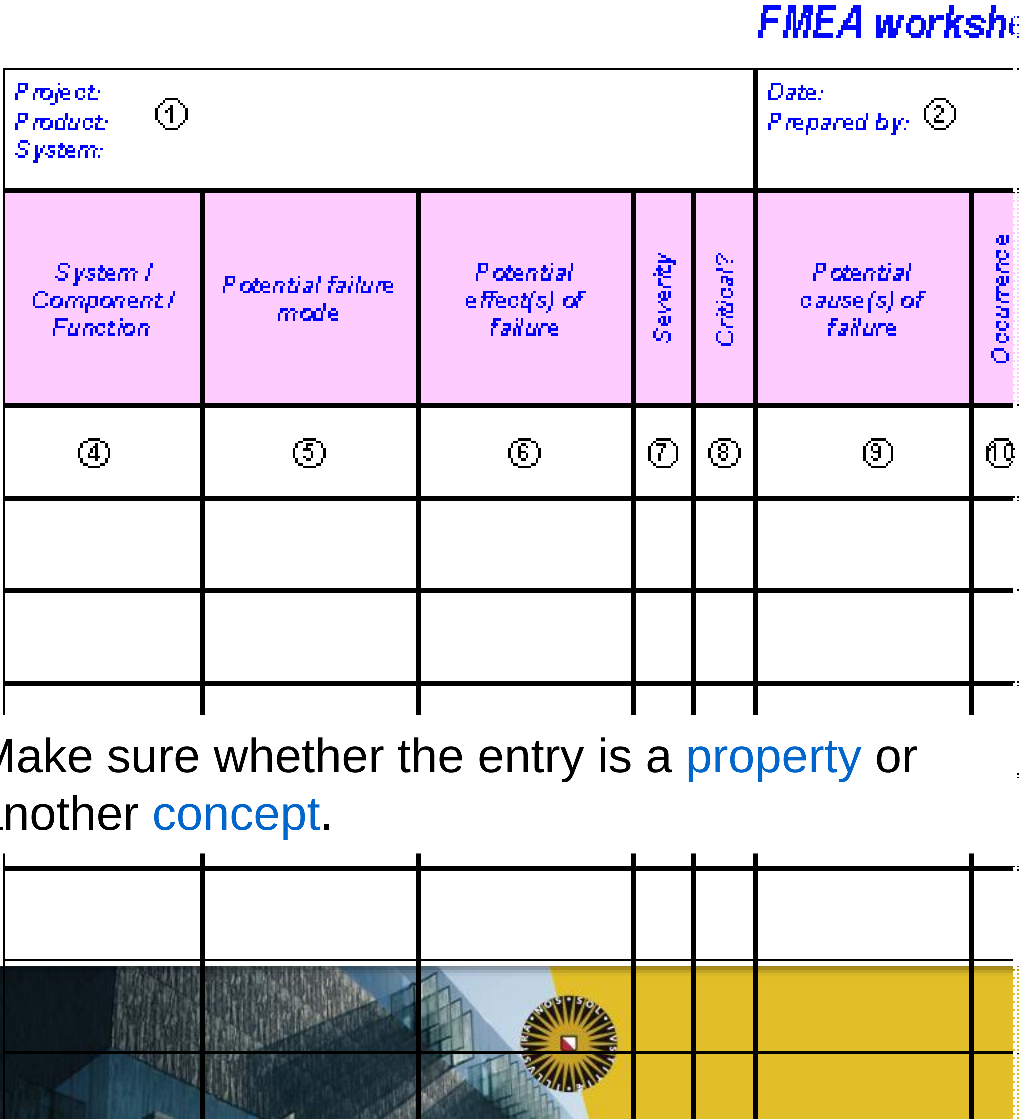
0..*

used ▸

1..*

COMPONENT

Complex tables



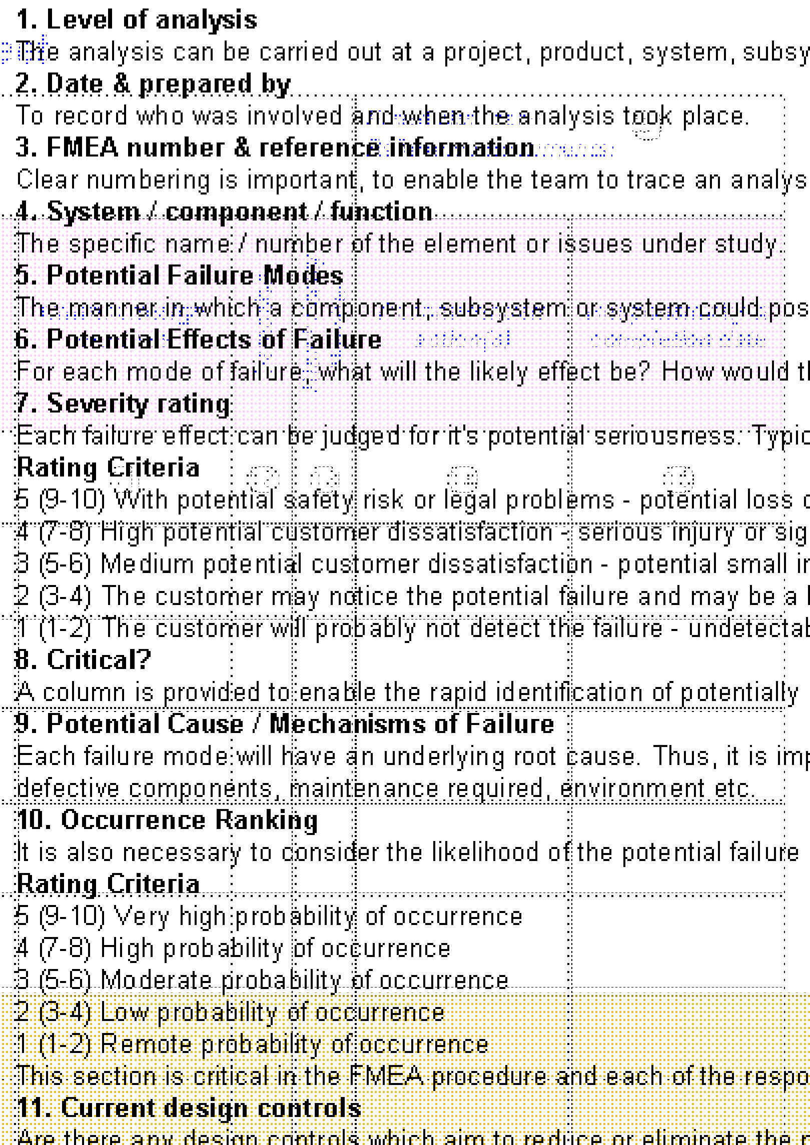
FMEA workshe

Project:
Product:
System: ①

Date:
Prepared by: ②

System / Component / Function	Potential failure mode	Potential effect(s) of failure	Severity	Critical?	Potential cause(s) of failure	Occurrence
④	⑤	⑥	⑦	⑧	⑨	⑩

Make sure whether the entry is a property or another concept.



1. Level of analysis

The analysis can be carried out at a project, product, system, subsy

2. Date & prepared by

To record who was involved and when the analysis took place.

3. FMEA number & reference information

Clear numbering is important, to enable the team to trace an analys

4. System / component / function

The specific name / number of the element or issues under study.

5. Potential Failure Modes

The manner in which a component, subsystem or system could pos

6. Potential Effects of Failure

For each mode of failure, what will the likely effect be? How would t

7. Severity rating

Each failure effect can be judged for it's potential seriousness. Typic

Rating Criteria

5 (9-10) With potential safety risk or legal problems - potential loss c

4 (7-8) High potential customer dissatisfaction - serious injury or sig

3 (5-6) Medium potential customer dissatisfaction - potential small in

2 (3-4) The customer may notice the potential failure and may be a

1 (1-2) The customer will probably not detect the failure - undetectab

8. Critical?

A column is provided to enable the rapid identification of potentially

9. Potential Cause / Mechanisms of Failure

Each failure mode will have an underlying root cause. Thus, it is imp
defective components, maintenance required, environment etc.

10. Occurrence Ranking

It is also necessary to consider the likelihood of the potential failure

Rating Criteria

5 (9-10) Very high probability of occurrence

4 (7-8) High probability of occurrence

3 (5-6) Moderate probability of occurrence

2 (3-4) Low probability of occurrence

1 (1-2) Remote probability of occurrence

This section is critical in the FMEA procedure and each of the respo

11. Current design controls

Are there any design controls which aim to reduce or eliminate the r

Version-dependent data (1)

- Products have different versions that vary over time, so the meta-data model should accommodate that in case it is essential.

- Examples:
  - Different versions of a Release Definition, Functional Design
  - Tracing and tracking of requirements in the release history

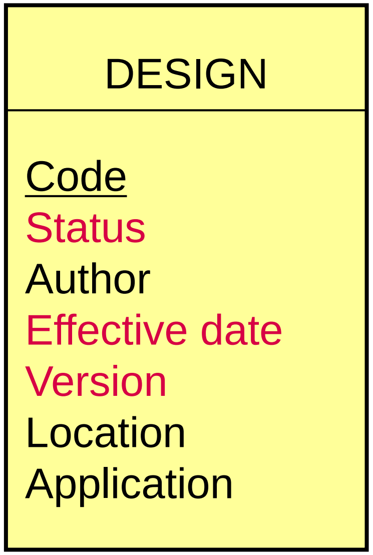
Version dependent data (2)

How to keep track of the history of the data of a deliverable?

Example: DESIGN has multiple versions

Properties

- in black: do not change across versions
- in red: change across versions



DESIGN

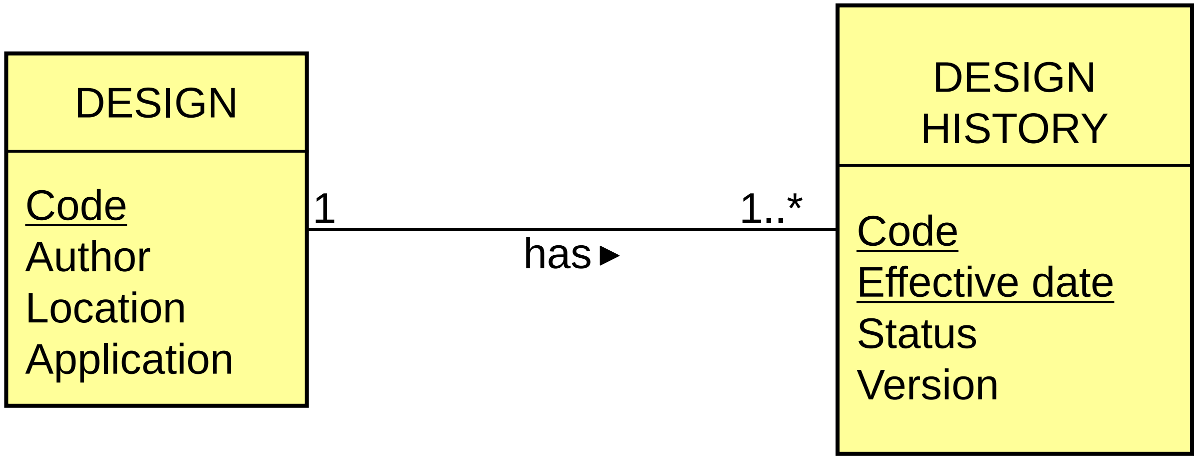
Code
Status
Author
Effective date
Version
Location
Application

Version stamping

- All time/version related properties are copied into a new Concept, called XX HISTORY and a history relationship is added in between

Example: DESIGN has DESIGN HISTORY

- The key is copied and the effective date is added.



DESIGN

Code
Author
Location
Application

1 has ▸ 1..*

DESIGN HISTORY

Code
Effective date
Status
Version

Concluding

- Meta-data modeling and meta-process modeling provide simple means for the documentation and communication of methodical processes and deliverables.

- Various applications of meta-modeling:
  - Situational methods and Method assembly (week 11)
  - Formalization of methods (week 11)
  - Method rationale (week 12)
  - Incremental method engineering (week 12)
  - Method association for product implementations (week 13)

QUESTIONS?