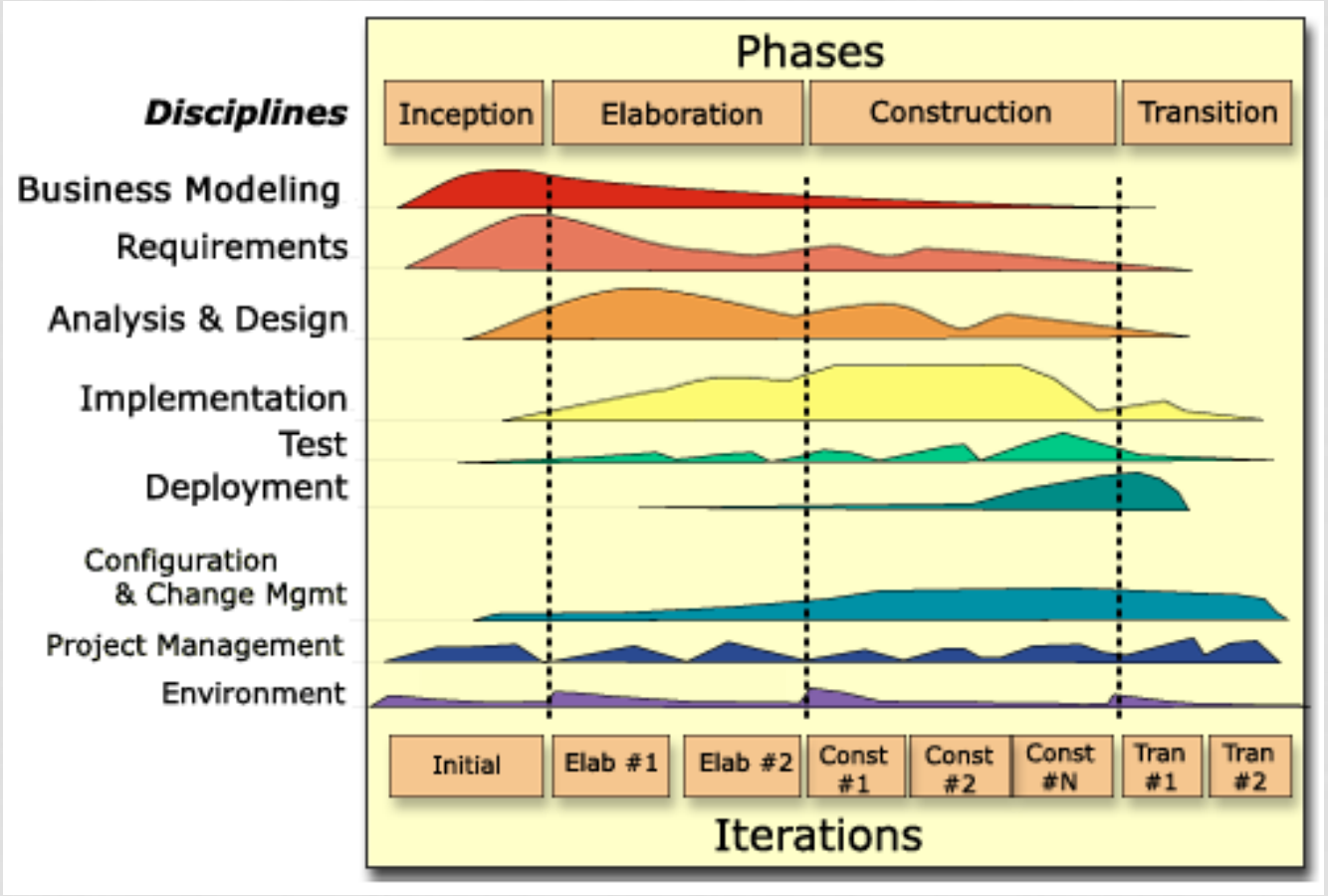
RUP

SOFTWARE PROJECTEN VOLGENS RUP



INHOUD

- **waarom RUP?**
- UP en RUP
- workflows
- use cases
- RUP op maat
- afronding

MEER DAN SCRUM/XP

meer) aandacht voor

in

c

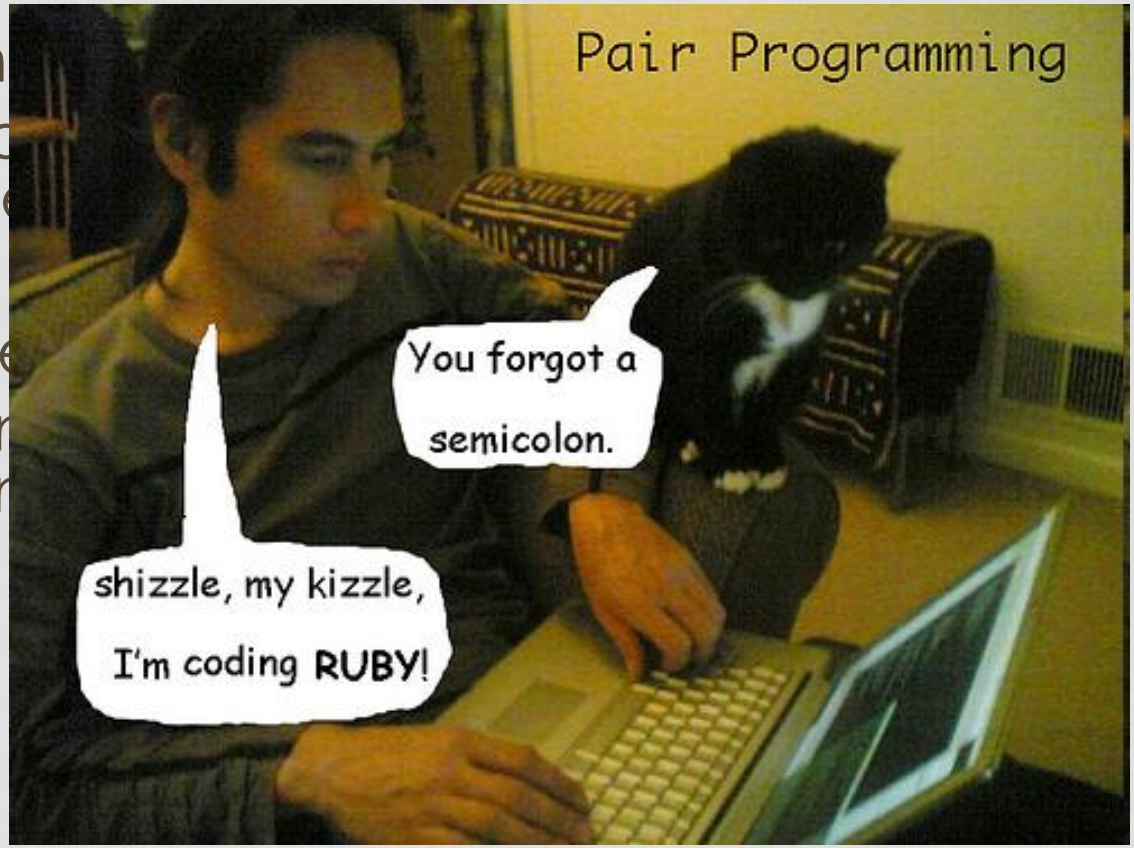
k

k

, e

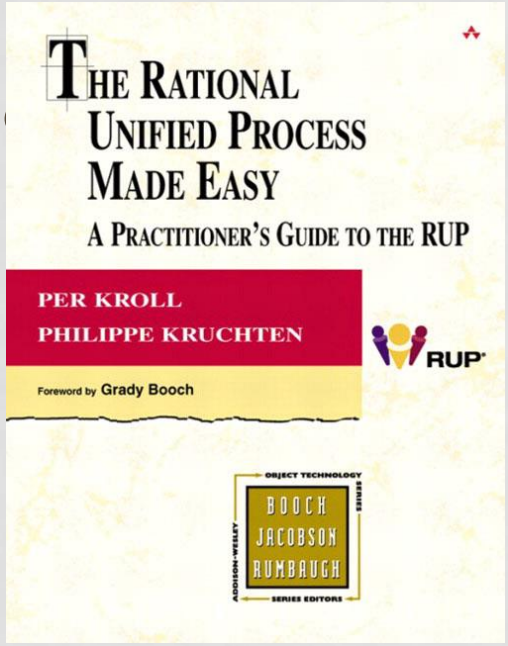
er

(ir



WAAROM RUP?

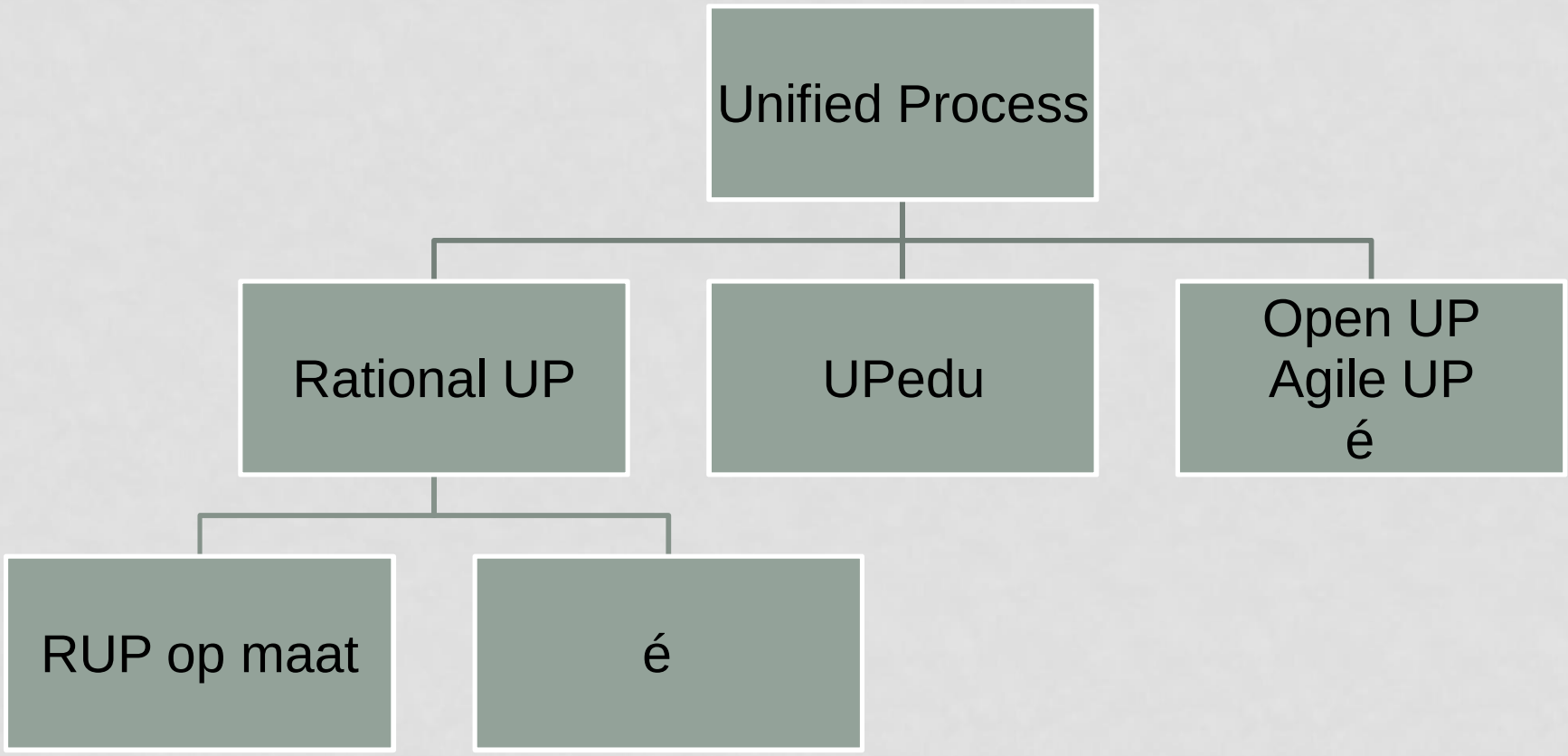
- zit conceptueel tussen Waterval en Agile in
- behandelt meer procesonderdelen dan Scrum
- "Op maat maken" dwingt je tot begrijpen
- heeft eigen sterktes (en zwaktes!)
- RUP is in Nederland erg populair, ve grote bedrijven



INHOUD

- waarom RUP?
- **UP en RUP**
- workflows
- use cases
- RUP op maat
- afronding

DE UNIFIED PROCESS-FAMILIE



zie verder http://en.wikipedia.org/wiki/Unified_process

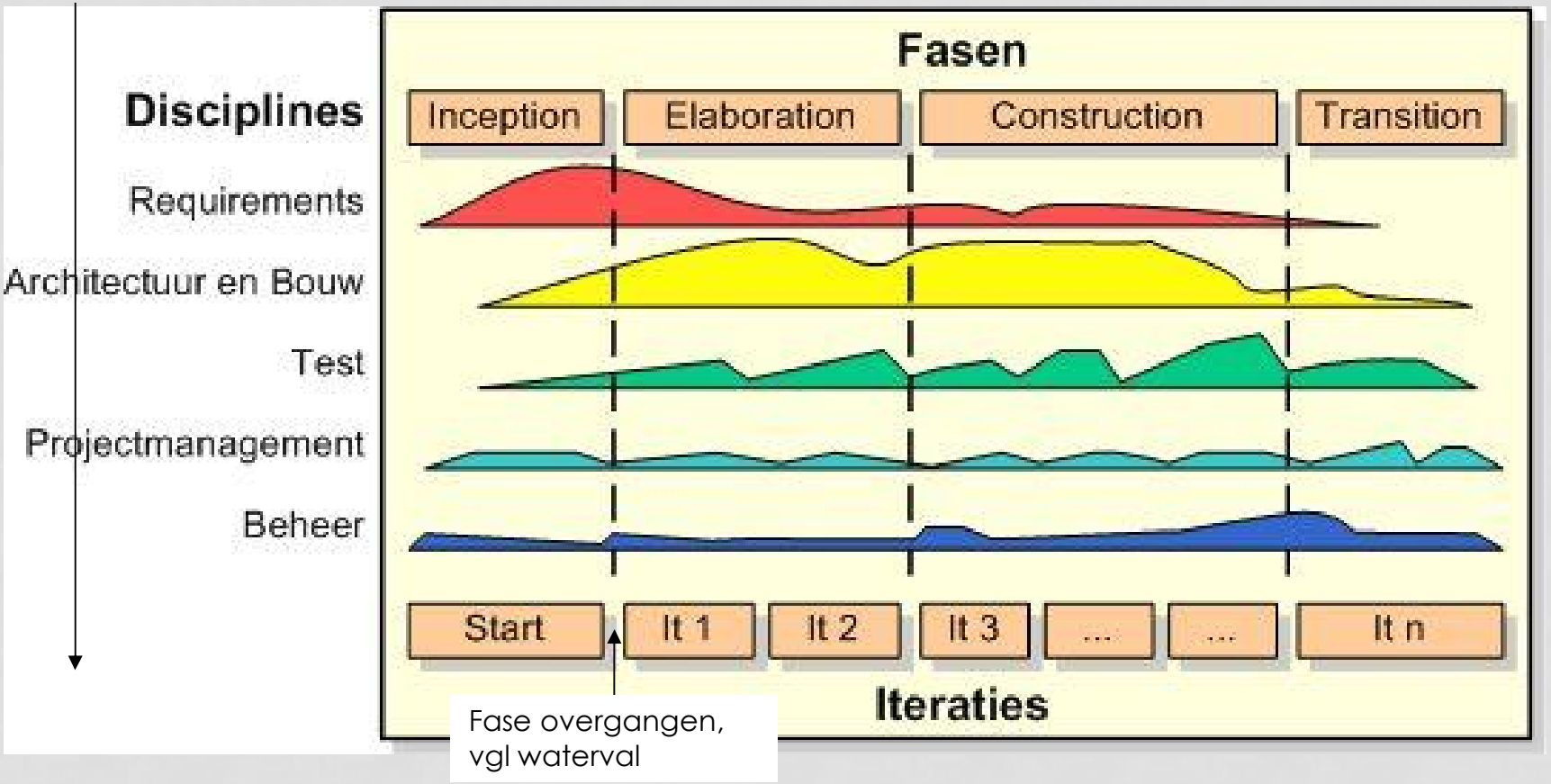
KENMERKEN VAN (R)UP

- iteratief en incrementeel
  - focus op werkend resultaat
  - lever iteratief iets van waarde
  - ingericht op veranderende reqs
- Use Case gedreven
  - geeft stakeholders een stem
- architectuur centrisch
  - stabiliseer de architectuur in werkende code
- risico's aanvallen
  - "attack risks or they will attack you"
- aanpasbaar
- focus op kwaliteit
  - documenteren
  - reviews op alle niveaus
  - testen op alle niveaus
  - specialisten doen het voorwerk

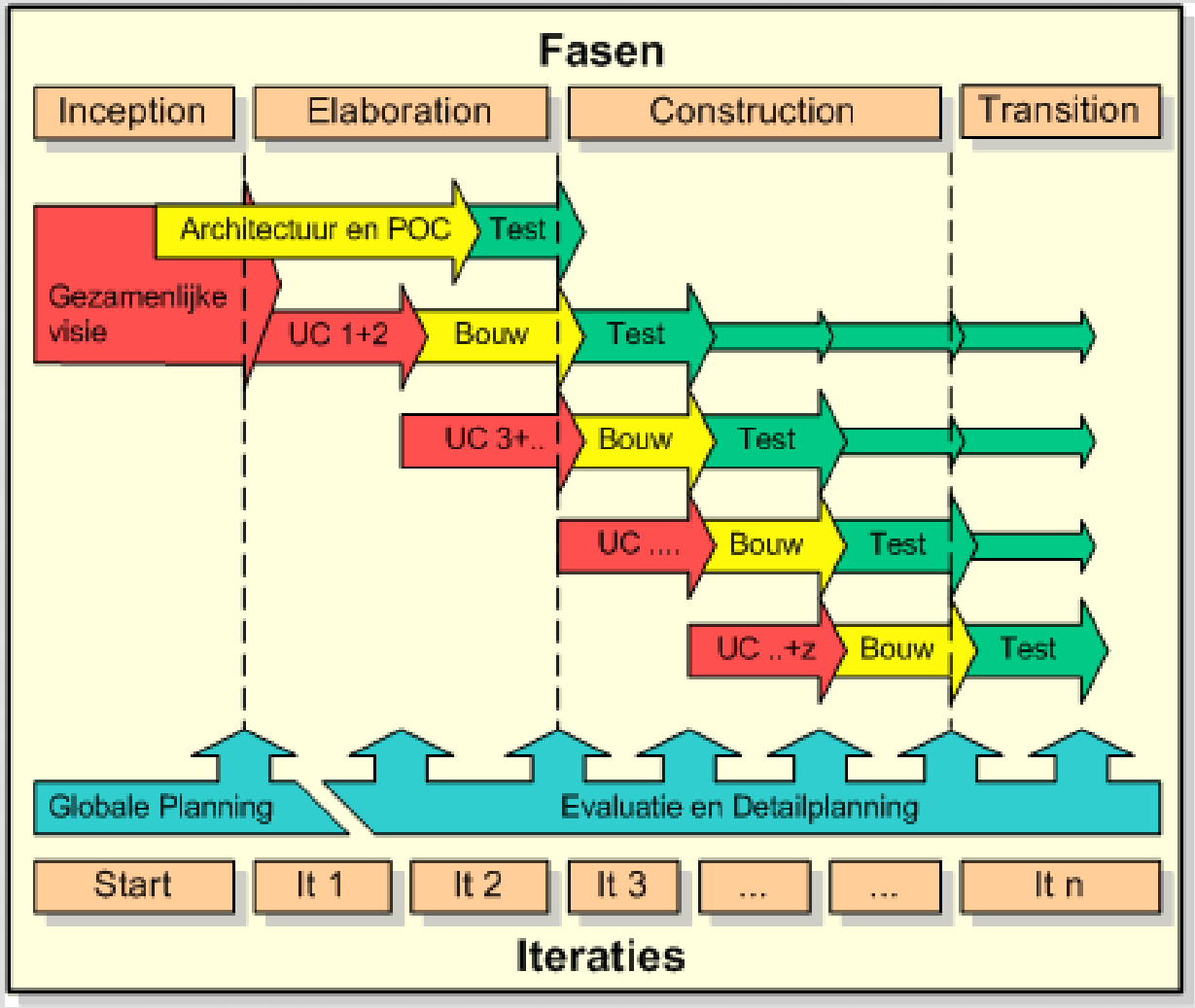
FASEN EN DISCIPLINES

Disciplines, verschilt per UP

Fasen, voor alle UP's gelijk



FASEN EN DISCIPLINES



INCEPTION

- inhoud, scope, risico's en globale planning helder krijgen
- milestones:
  - Vision document (incl. business case!)
  - Use Case Model
  - Glossary
  - Idee van oplossing, tooling
  - Risico's, planning, schatting (Software Development Plan)

ELABORATION

- risico's overwinnen (testen architectuur, kritische UC's uitwerken, proof-of-concepts)
- milestones:
  - beeld van kritische requirements (UC specs uitgewerkt)
  - Software Architecture Document (SAD) + PoC uitgewerkt
  - grootste risico's overwonnen
  - SDP bijgesteld
  - ontwikkelomgeving

CONSTRUCTION

- iteratief ontwerpen, bouwen, testen
- milestones:
  - werken per Use Case
  - alle deelopleveringen voldoen aan functionele eisen en acceptatiecriteria

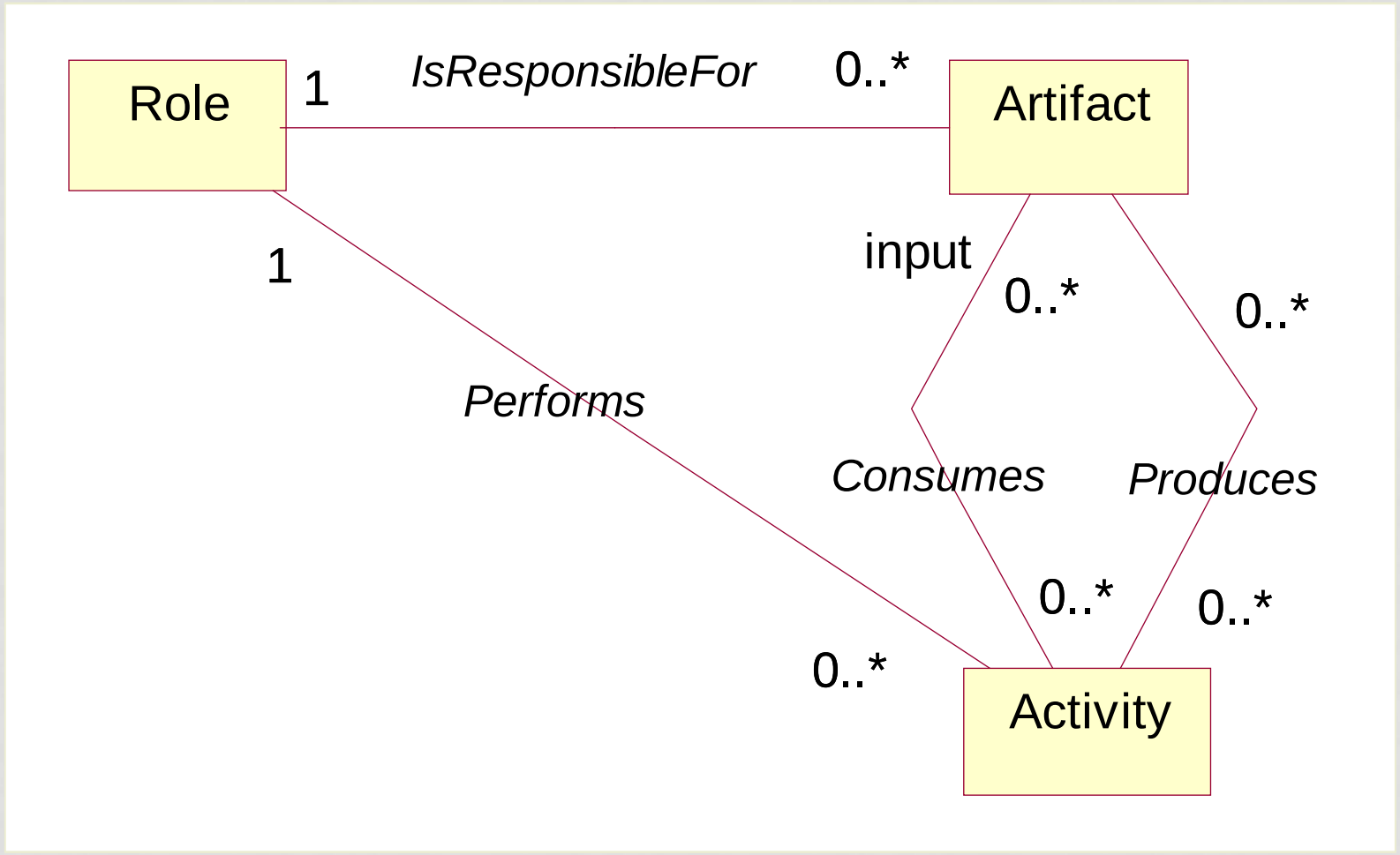
TRANSITION

- bugfixen, trainen, deployment, overdragen aan beheer, afsluiting
- milestones:
  - bugs gefixt
  - getrainde gebruikers en beheerders
  - geaccepteerd product
  - project geëvalueerd (feedbackloop met het proces!)

INHOUD

- waarom RUP?
- UP en RUP
- **workflows**
- use cases
- RUP op maat
- afronding

ROLES, ACTIVITIES & ARTIFACTS



ROLES, ACTIVITIES & ARTIFACTS

Roles

- Gebruiker
- Projectleider
- Informatie-analist
- Architect
- UC-ontwerper
- Testmanager
- Tester
- Programmeur
- Beheerder
- …

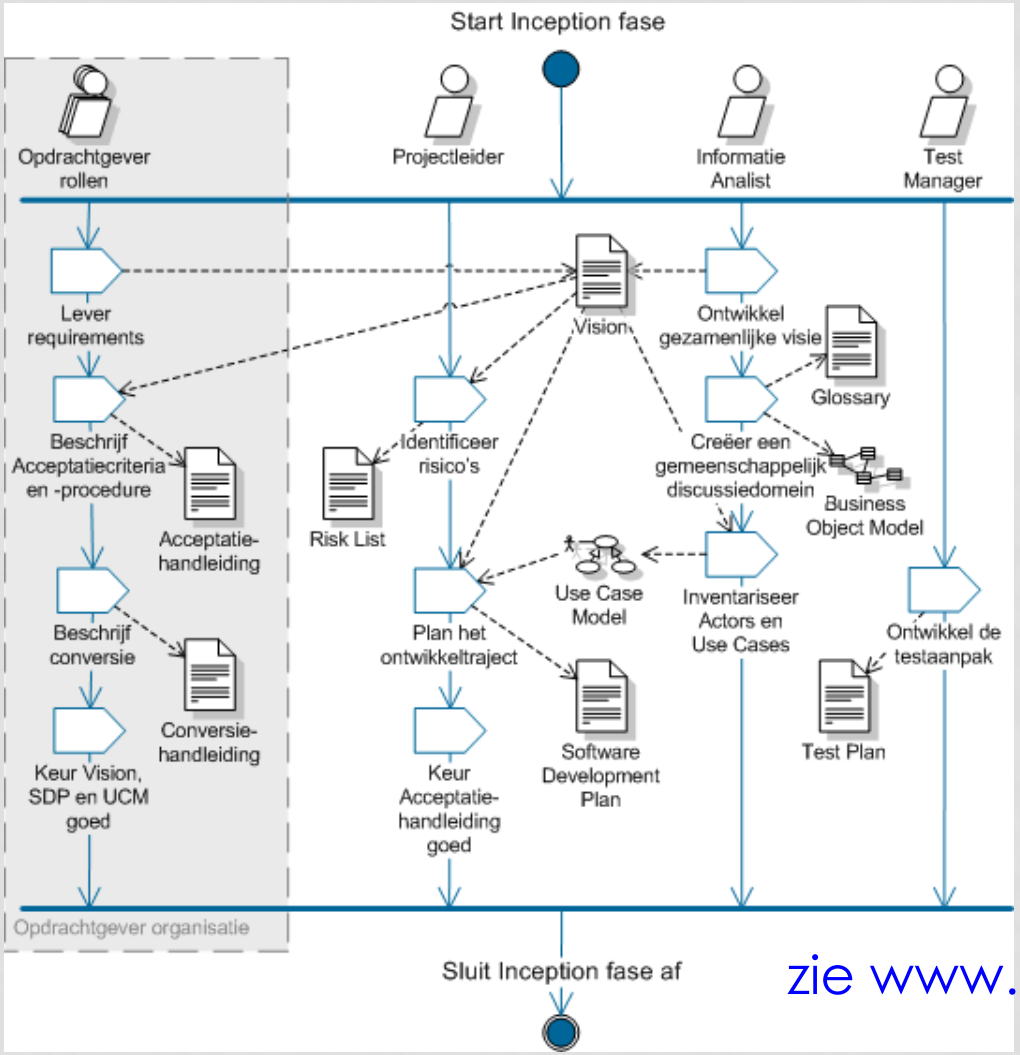
Activities

- Beschrijf acc.criteria
- Richt een helpdesk in
- Identificeer risico's
- Evalueer een iteratie
- Specificeer een use case
- Construeer de SA
- Doe een PoC
- …

Artifacts

- Vision
- Glossary
- Risk list
- SDP
- Test Plan
- SAD
- Navigation map
- OTAP omgeving
- Use Case model
- Class model
- Testrapport
- …

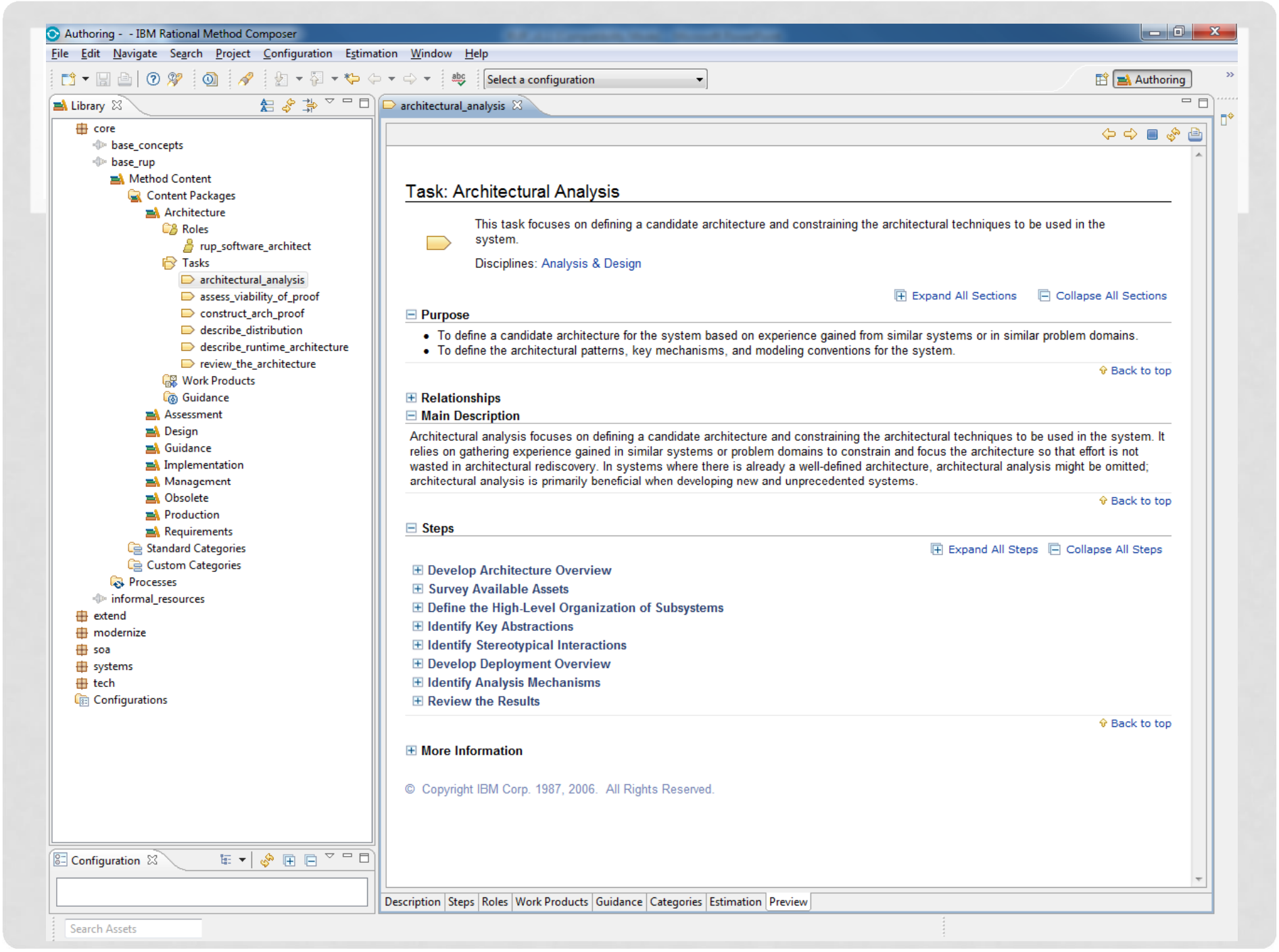
WORKFLOWS



zie www.rupopmaat.nl

JE MAAKT ZELF HET PROCES

- besluit wat je gaat gebruiken
- geen excuus dat RUP te zwaar is
- Rational Method Composer (te zwaar voor de meeste bedrijven)


Authoring - - IBM Rational Method Composer
File Edit Navigate Search Project Configuration Estimation Window Help
Select a configuration
Authoring
Library
core
base_concepts
base_rup
Method Content
Content Packages
Architecture
Roles
rup_software_architect
Tasks
architectural_analysis
assess_viability_of_proof
construct_arch_proof
describe_distribution
describe_runtime_architecture
review_the_architecture
Work Products
Guidance
Assessment
Design
Guidance
Implementation
Management
Obsolete
Production
Requirements
Standard Categories
Custom Categories
Processes
informal_resources
extend
modernize
soa
systems
tech
Configurations
Configuration
Search Assets
architectural_analysis
Task: Architectural Analysis
This task focuses on defining a candidate architecture and constraining the architectural techniques to be used in the system.
Disciplines: Analysis & Design
Expand All Sections
Collapse All Sections
Purpose
To define a candidate architecture for the system based on experience gained from similar systems or in similar problem domains.
To define the architectural patterns, key mechanisms, and modeling conventions for the system.
Back to top
Relationships
Main Description
Architectural analysis focuses on defining a candidate architecture and constraining the architectural techniques to be used in the system. It relies on gathering experience gained in similar systems or problem domains to constrain and focus the architecture so that effort is not wasted in architectural rediscovery. In systems where there is already a well-defined architecture, architectural analysis might be omitted; architectural analysis is primarily beneficial when developing new and unprecedented systems.
Back to top
Steps
Expand All Steps
Collapse All Steps
Develop Architecture Overview
Survey Available Assets
Define the High-Level Organization of Subsystems
Identify Key Abstractions
Identify Stereotypical Interactions
Develop Deployment Overview
Identify Analysis Mechanisms
Review the Results
Back to top
More Information
© Copyright IBM Corp. 1987, 2006. All Rights Reserved.
Description Steps Roles Work Products Guidance Categories Estimation Preview

INHOUD

- waarom RUP?
- UP en RUP
- workflows
- **use cases**
- RUP op maat
- afronding

USE CASES

- centraal in RUP, basis van werk
- Use case model als start, gevolgd door Use Case Specifications (=Descriptions)
- input voor
  - trainingsmateriaal
  - ontwerp
  - testontwerp

USE CASES REVISITED

- Interactie van een *actor* met het systeem
- Synthese: één zinvol doel
- 'Contract' over het gedrag van het systeem

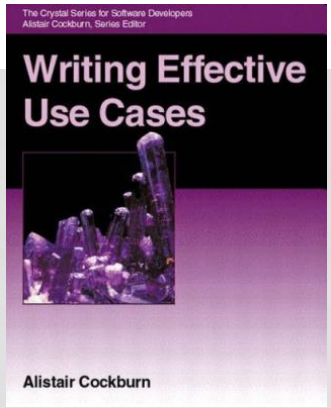
- Gezamenlijke use cases beschrijven de volledige functionaliteit
- Bruikbaar voor requirements en ontwerp

VOORBEELD (CASUAL)

USE CASE 4: BUY SOMETHING (CASUAL VERSION)

The Requestor initiates a request and sends it to her or his Approver. The Approver checks that there is money in the budget, check the price of the goods, completes the request for submission, and sends it to the Buyer. The Buyer checks the contents of storage, finding best vendor for goods. Authorizer: validate Approver's signature. Buyer: complete request for ordering, initiate PO with Vendor. Vendor: deliver goods to Receiving, get receipt for delivery (out of scope of system under design). Receiver: register delivery, send goods to Requestor. Requestor: mark request delivered.

At any time prior to receiving goods, Requestor can change or cancel the request. Canceling it removes it from any active processing. (delete from system?)Reducing the price leaves it intact in process. Raising the price sends it back to Approver.



VOORBEELD (FULLY DRESSED)

USE CASE 5: BUY SOMETHING (FULLY DRESSED VERSION)
Primary Actor: Requestor
Goal in Context: Requestor buys something through the system, gets it. Does not include paying for it.
Scope: Business - The overall purchasing mechanism, electronic and non-electronic, as seen by the people in the company.
Level: Summary
Stakeholders and Interests:
Requestor: wants what he/she ordered, easy way to do that.
Company: wants to control spending but allow needed purchases.
Vendor: wants to get paid for any goods delivered.
Precondition: None
Minimal guarantees: Every order sent out has been approved by a valid authorizer. Order was logged so that company can only be billed for valid goods received.
Success guarantees: Requestor has goods, correct budget ready to be debited.
Trigger: Requestor decides to buy something.
Main success scenario:
1. *Requestor:* initiate a request
2. *Approver:* check money in the budget, check price of goods, complete request for submission
3. *Buyer:* check contents of storage, find best vendor for goods
4. *Authorizer:* validate Approver's signature
5. *Buyer:* complete request for ordering, initiate PO with Vendor
6. *Vendor:* deliver goods to Receiving, get receipt for delivery (out of scope of system under design)
7. *Receiver:* register delivery, send goods to Requestor
8. *Requestor:* mark request delivered.

Extensions:
1a. Requestor does not know vendor or price: leave those parts blank and continue.
1b. At any time prior to receiving goods, Requestor can change or cancel the request.
Canceling it removes it from any active processing. (delete from system?)
Reducing price leaves it intact in process.
Raising price sends it back to Approver.
2a. Approver does not know vendor or price: leave blank and let Buyer fill in or call back.
2b. Approver is not Requestor's manager: still ok, as long as approver signs
…
Technology and Data Variations List: (none)
Priority- various
Releases - several
Response time - various
Freq of use - 3/day
Channel to primary actor: Internet browser, mail system, or equivalent
Secondary Actors: Vendor
Channels to Secondary Actors: fax, phone, car
Open issues:
What authorization is needed to cancel a request?
…

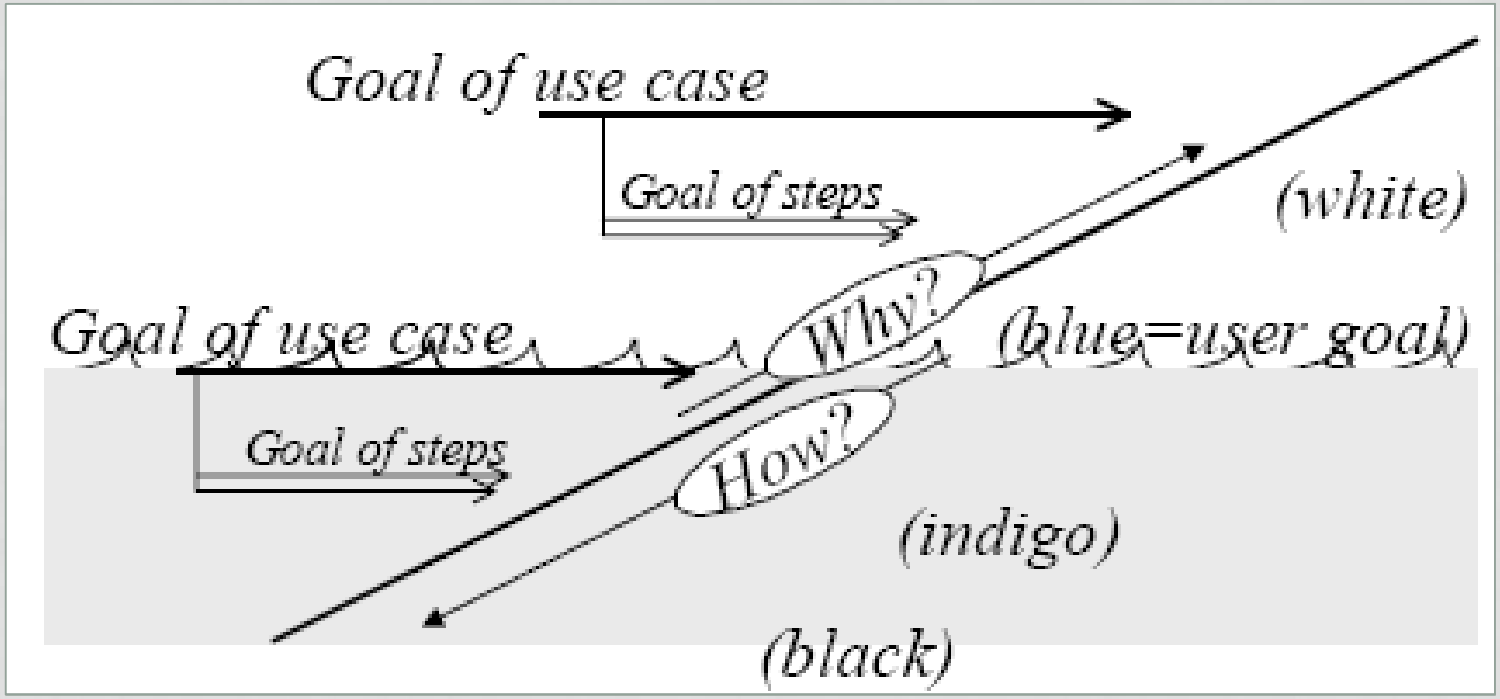
OPSTELLEN USE CASE (COCKBURN)

- schrijf iets leesbaars
- beschrijf top-down, “breadth first”
  1. alleen actor en goal
  2. main succes scenario
  3. extension conditions
  4. extension handling
- gebruik 3 tot 9 stappen per use case
  ‘refactor’ waar nodig.
- data en business rules benoemen maar elders beschrijven

PER SCENARIO STAP

- bereik een doel
- elke stap: <actor> <actie> <object> <…>
- acties, validatie, update
- “grey box”
- benoem geen UI details!
- tijdsvolgorde aangeven m.b.v. commentaar
- zoek juiste niveau met ‘hoe?’ en ‘waarom?’

NIVEAUS OMHOOG & OMLAAG



INHOUD

- waarom RUP?
- UP en RUP
- workflows
- use cases
- **RUP op maat**
- afronding

STANDAARDEN EN HOE ER MEE OM TE GAAN

- veel is al bedacht en vastgelegd in standaarden
- vraag je af:
  - welke standaarden zijn van toepassing?
  - welke onderdelen van de standaard zijn relevant?
- hoe pas je een standaard efficiënt toe?
  - afstemmen op jouw situatie (vb: RUP dev. case)
  - ondersteunen met templates en checklists
  - voorkom “blind volgen” en “dom invullen”

TEMPLATES EN TOOLING

- Templates:
  - www.rupopmaat.nl
  - www.yoopeedoo.org

INHOUD

- waarom RUP?
- UP en RUP
- workflows
- use cases
- RUP op maat
- **afronding**

DE PLUSSEN EN MINNEN VAN RUP

+++

- meer aandacht voor architectuur (itt Scrum)
- meer aandacht voor documentatie, overdraagbaarheid, testen, project mgt (itt Scrum)
- haalt risico's naar voren (vgl. waterval)

- wat documenteer je wel en niet?
- risico op "teveel proces"

HOE VERDER? (1)

- Je oefent op onderdelen en neemt de rest mee naar afstuderen en daarna.

- Week 1: Development case
- Week 2: Software Development Plan
- Week 3: Requirements
- Week 4/5: Testen
- Week 6-9: Volledige set

HOE VERDER? (2)

boek:

- Hfst 1-4: lezen
- Hfst 5-8: bladeren
- Hfst 9-11: naslag

sites:

- blader http://www.rupopmaat.nl
- Blader http://www.yoopeedoo.org → UPEDU → (op upedu.org gekomen) Artifacts sets → Templates… voor nog meer templates én uitgewerkte voorbeelden.