Unified Process

Peter Dolog
dolog [at] cs [dot] aau [dot] dk
5.2.47
Information Systems
March 3, 2008

Outline

Model Driven Design

Tutorial on Requirements Eng. and SCRUM reflections (D402a, s601c)

Unified Process

- History
- Principles
- Artifacts
- Practices
- Roles

UP

Unified Process, 1990's
Iterative, originally not agile but can be
Risk-driven development in early iterations focusing on
creation of the core architecture and driving down the high risks
2-6 weeks iterations

History of UP

Some of the roots in "spiral model" of Barry Boehm
Core initial development around 1995-98
Large Canadian Air Traffic Control project as test bed
Phillippe Kruchten chief architect of UP/RUP
Rational Corporation had commercial product in mind (RUP) but also reached out to public domain (UP)

Unified Process (UP)

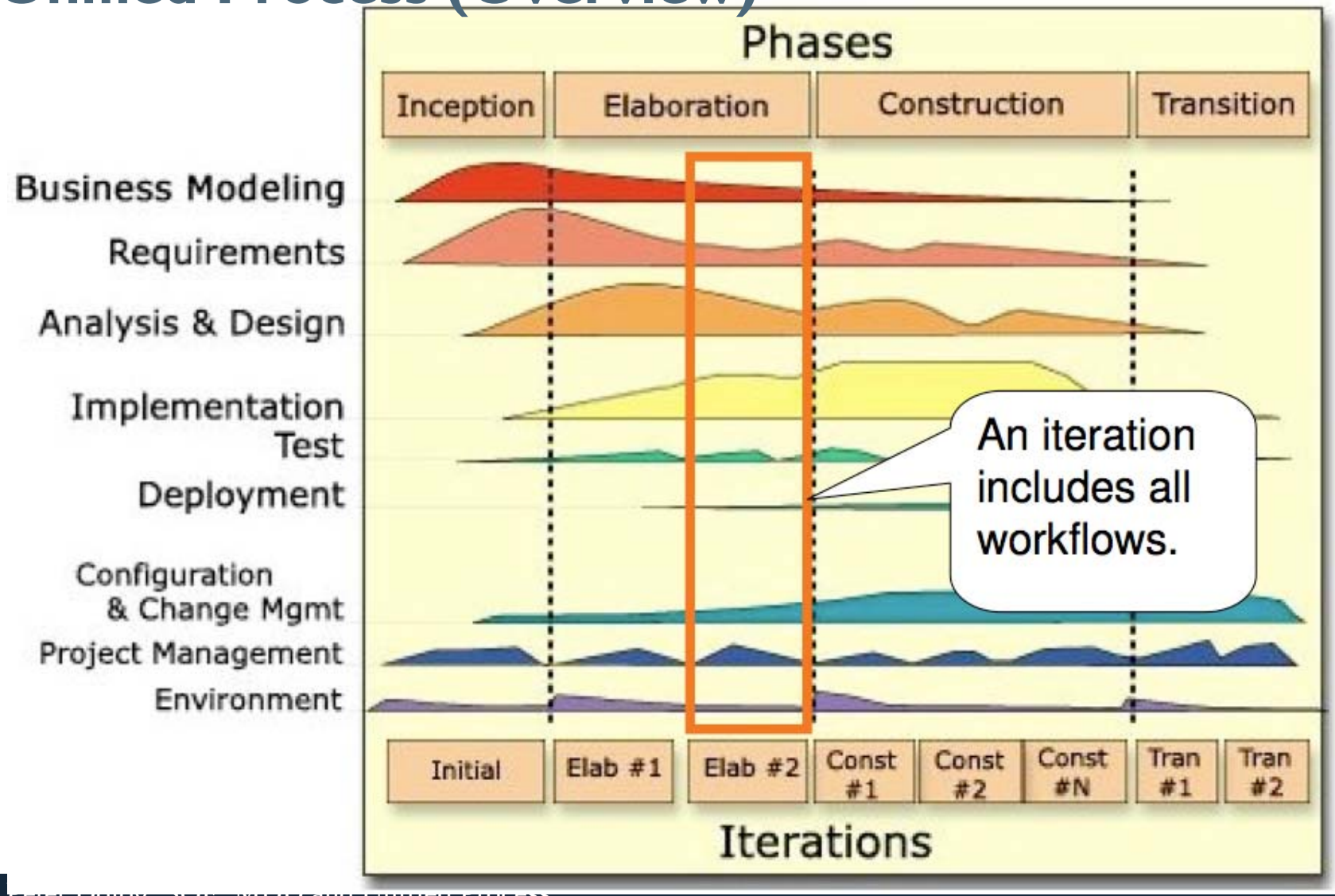
Popular iterative process framework, especially its refinement:

- Rational Unified Process (RUP)

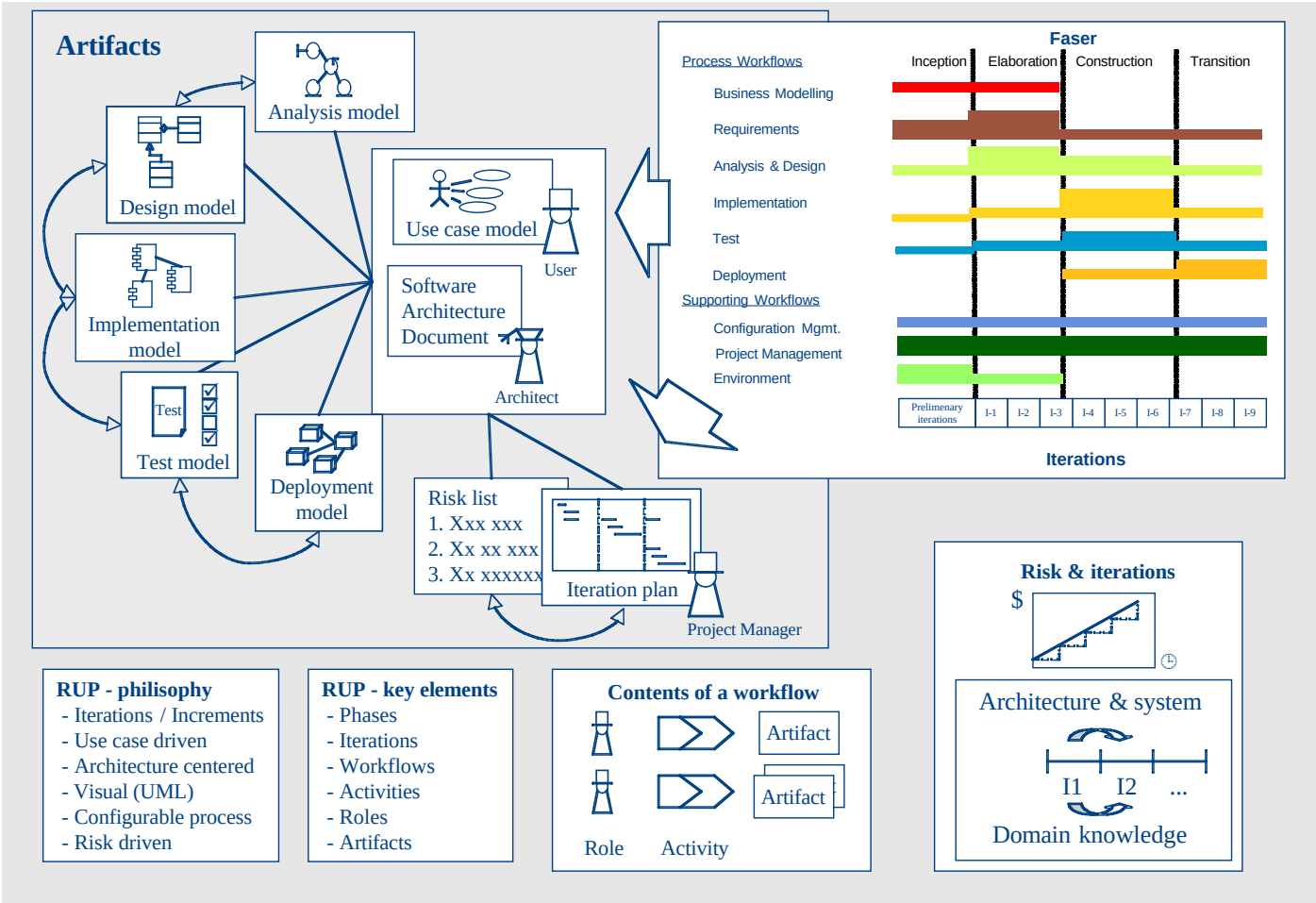
Key practices and guidelines:

- Short time-boxed iterations
- Develop high-risk elements in early iterations
- Deliver value to customer
- Accommodate change early in project
- Work as one team

Unified Process (Overview)



RUP - overview



Key Ideas and Practices

Originally developed by Rational

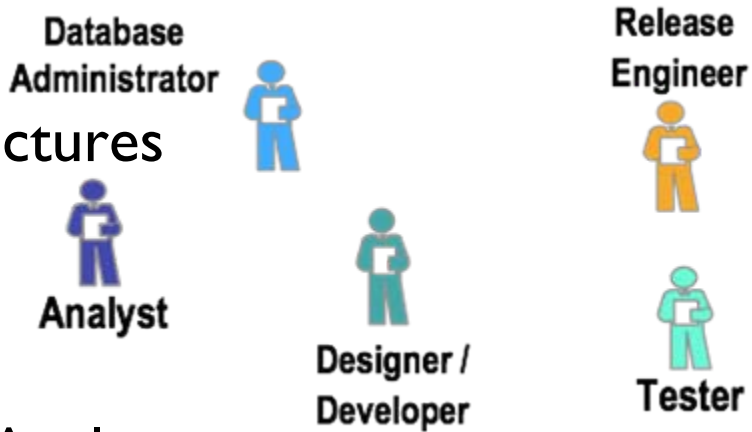
An iterative process framework

Project lifecycle phases

Identifies workers, activities, artifacts

Promotes certain practices:

- Develop software iteratively
- Manage requirements
- Use component-based architectures
- Visually model software
- Verify software quality
- Control changes to software



Six Best "must" Practices

Time-boxed iterations

Avoid attempting large, up-front requirements

Strive for **cohesive architecture** and reuse **existing components**

On large projects: reqs & core architecture developed by small co-located team;
then early team members divide into sub-project leaders

Continuously verify quality

Test early, often, and realistically by integrating all software each iteration

Six Best "must" Practices (2)

Visual modeling

Prior to programming, do at least some visual modeling to explore creative design ideas

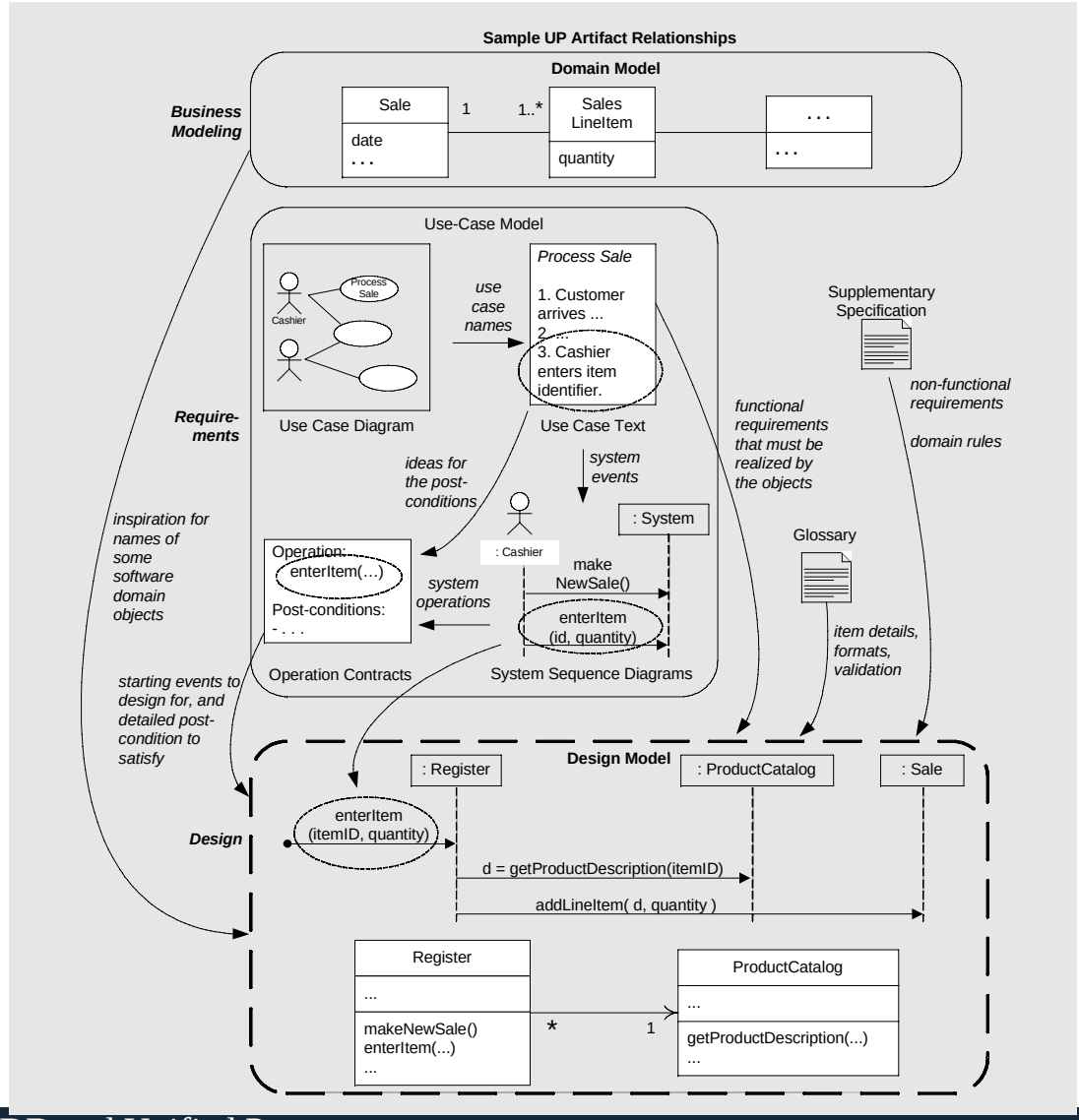
Manage requirements

Find, organize, and track requirements iteratively through skillful means. Use tools.

Manage change

Disciplined configuration management and version control, change request protocol, base-lined releases at the end of each iteration

Larman's Design Process



Life cycle in four phases

Inception

Business case, vision, identify high risks & 10% of key reqs in detail, estimate elaboration effort

Elaboration

Core & architecturally significant parts coded/tested, key risks identified/mitigated, 80% of major reqs evolved/defined

Construction

Builds remaining system in short iterations, efficient and predictable due to solid elaboration

Transition

Exposes release candidate for review/feedback, then deployment

Some prominent work products

Vision: summary of objectives, features, business case
Software Architecture Document: Short learning aid to understand the system
Test Plan: summary of goals and methods of testing
Iteration Plan: detailed plan for the next iteration
Change Request: uniform way to track all requests for work, e.g. defects

Example roles in UP

Stakeholder: customer, product manager, etc
Software Architect: establishes and maintains architectural vision
Process Engineer: leads definition and refinement of Development Case
Graphic Artist: assists in user interface design, etc

Some UP Guidelines

Attack risks early and continuously before they will attack you

Stay focused on developing executable software in early iterations

Prefer component-oriented architectures and reuse of existing components

Baseline an executable architecture early

How to fail with UP

elaboration phase goal is to create a throwaway prototype

- prototypes are acceptable in UP, e.g. during inception, but elaboration goal is creation of subset of final system

iterations too long

- iterations should typically be 2-6 weeks long not months

team should do lots of modelling and UML diagrams, and use a CASE tool

- UP contains optional models with potential use of UML but UP also compatible with agile approach, e.g. whiteboard hand sketches etc

How to fail with UP (2)

Not conforming to official UP work product or phase names

- common vocabulary vital within organization and across global UP-conforming teams

Development Case too complex, too many work products

- "less is better," UP recommends adding work products that really add value

Software Architecture Document "finished" before end of elaboration

- UP SAD is learning aid, so this would imply "up-front design"

UP in "The Real World"

Large: Canadian Air Traffic Control System

Ten years, Ada and C++, test bed for practices RUP, previous failed waterfall attempt 11 years & $ 2.6 billion USD

Medium: Ogre Nextgen Economic Modeling System

2 years, Java technologies, decision support system for oil/gas asset holders

Small: QUICKcheck point-of-sale, 1 year, six people, Java technologies, self-checkout system for grocery stores (main developer: Kyrus)

Rational Unified Process(RUP)

Evolved from Unified Process developed by the "the three Amigos " – Booch, Rumbaugh and Jacobson.

Philippe Krutchen made a detailed refinement on UP and hence came RUP

Main focus on Iterative and Incremental Development

Is RUP agile ?

RUP can be used in a very traditional waterfall style or in an agile manner.

"You can use RUP as a agile process, or as a heavyweight process - it all depends on how you tailor it in your environment. " – Martin Fowler

Craig Larman is a strong proponent of using the RUP in an agile manner

Rational Unified Process

Wide spread methodology championed by Rational Corporation

Combines water-fall and evolutionary development

Plan a little, design a little, code a little.

Aims to minimizes risk of failure

Breaks system into mini-projects, focusing on riskier elements first

Other (claimed) advantages

- Encourages all participants, including testers, integrators, and documenters to be involved earlier on
- Mini-waterfalls centered around UML, a particular OO-methodology
- CASE-TOOL support (of course, from Rational)

Does it work?

- Many positive case studies although benefits difficult to quantify

Cockburn scale

Criticality

Life (L)	L6	L20	L40	L100
Essential Money (E)	E6	E20	E40	E100
Discretionary Money (D)	D6	D20	D40	D100
Comfort (C)	C6	C20	C40	C100

People

Degree of Ceremony and Cycles