Outline

I Introduction
II Foundations
III The Development Life Cycle
IV Requirements
V Analysis & Design
VI Implementation
VII Verification & Validation
VIII Summary and Outlook
III The Development Life Cycle

III.1 Software Engineering Life Cycle Models

III.2 System Engineering Life Cycle Models

III.3 Embedded System Life Cycle Models

III.4 Advanced Life Cycle Models & MDD

III.5 Process Improvement

III.6 Discussion & Summary

III.7 Bibliography
III.1 Software Engineering Life Cycle Models

- Waterfall Model
- Prototyping
- V Model
- Spiral Model
- RUP
Characteristics of Software Development Methodologies

(Historic) Waterfall Model

Requirements definition → Analysis → Design → Coding (+ Unit Test) → System test → Installation and conversion → Operation and maintenance
The Prototyping Process

- REQUIREMENTS DETERMINATION BY CUSTOMER
- PROTOTYPE DESIGN
- PROTOTYPE IMPLEMENTATION
- PROTOTYPE EVALUATION BY CUSTOMER
- REQUIREMENTS FULFILLED?
  - YES
    - SYSTEM TESTS AND ACCEPTANCE TESTS
    - SYSTEM CONVERSION
    - SYSTEM OPERATION AND MAINTENANCE
  - NO
    - REQUIREMENTS FOR CORRECTIONS, CHANGES AND ADDITIONS

[Gal2004]
“V” Development Process

- Requirement analysis
- Specification
- Coarse design
- Detailed design
- Implementation
- Module test
- Integration test
- System test
- Service

- Requirements documents
- System model, construction
- Quality management
- Tested modules
- Modules
- System, verified system
- Integrated system
- Certified system
- Service
Spiral Model Process

Negotiation
- objectives, alternatives, strategies, constraints

Evaluation
- alternatives: „Make-or-Buy“, risk analysis

Construction & Test
- any SE-Process for partial or full system!

Planning
- Review,
- Plan next phases
Rational Unified Process (RUP)

[ RUP1999 ]

Phases

Inception | Elaboration | Construction | Transition

Workflows

Business Modeling
Requirements
Analysis & Design
Implementation
Test
Deployment
Configuration & Change Mgmt
Project Management
Environment

Iterations

Initial | Elab #1 | Elab #2 | Const #1 | Const #2 | Const #N | Tran #1 | Tran #2

RUP Overview Diagram
III The Development Life Cycle

III.1 Software Engineering Life Cycle Models

III.2 System Engineering Life Cycle Models

III.3 Embedded System Life Cycle Models

III.4 Advanced Life Cycle Models & MDD

III.5 Process Improvement

III.6 Discussion & Summary

III.7 Bibliography
Life Cycle of System Engineering

[Sage&Armstrong2000]
Alternative View

[Sage&Armstrong2000]
III The Development Life Cycle

III.1 Software Engineering Life Cycle Models
III.2 System Engineering Life Cycle Models
III.3 Embedded System Life Cycle Models
III.4 Advanced Life Cycle Models & MDD
III.5 Process Improvement
III.6 Discussion & Summary
III.7 Bibliography
III.3 Embedded System Life Cycle Models

(1) 3V Model

(2) Multiple V Model
(1) 3V Model (1/2)

Model
- design
- test
- build

Prototype(s)
- design
- test
- build

Final product
- design
- test

[Broekman&Notenboom2003]
3V Model (2/2)

- Model: covers the definition and simulation of the overall system functionality
  - Implementation aspects are not considered
- Prototype: is characterized by rapid prototyping
  - hardware specific parameters become important
  - deployment & message scheduling
  - local design addresses the scheduling of tasks on each node
- Final product: addresses the system development for the final target hardware
  - typical problem: limited performance of the target system

http://www.vmars.tuwien.ac.at/projects/setta
(2) Multiple V Model

[Broekman&Notenboom2003]
Multiple V Model (2/2)
III The Development Life Cycle

III.1 Software Engineering Life Cycle Models

III.2 System Engineering Life Cycle Models

III.3 Embedded System Life Cycle Models

III.4 Advanced Life Cycle Models & MDD

III.5 Process Improvement

III.6 Discussion & Summary

III.7 Bibliography
III.4 Advanced Life Cycle Models & MDD

(1) MDA
(2) Y-Model
(3) Platform-Based Design
(1) MDA

- An approach to IT system specification that separates the specification of system functionality from the specification of the implementation of that functionality on a particular technology platform.

- “Design once, build it on any platform”

Diagram:

```
Platform Independent Model (PIM)

Platform Specific Model (PSM)

Code
(+ Platform, ...)
```

- UML, ...
- Platforms (e.g., CORBA Profile, ...)
- C++
  (+ CORBA, ...)

Early Problem Detection in MDA

- Models permit to detect some problems early on:
  - Reduced defect detection costs
  - Reduced costs for defect removal

- Traceability and portability

But this is a vision only for software-intensive systems!
(2) Y-Model

Manual coding

Standard automatic code generator

Qualified code generator

Design verifier

Generating code

Programming code

No code test (Automated code test)

Automated design verification

Time

[Camus&Dion2003]
http://www.safeair.org/
Application Example: Airbus

Tool:
- Safety Critical Applications Development Environment (SCADE)

Application:
- A340/600 FCSC (Flight Control Secondary Computer):

Result:
- 70 % automatically generated code
- 50 % reduction in development cost
- reduction in modification cycle time by factor 3

Source: Esterel Technologies
(3) Platform-Based Design

Requirements → Verification
automatic (program analysis/model checking)

Verification → Environment

Environment → Resources
automatic (compilation/synthesis)

Resources → Implementation

Implementation → Model

Model → Platform Design-Space Export

Platform Design-Space Export → Platform Mapping

Platform Mapping → Application Space
Application Space → Application Instance

Application Instance → Platform Instance

Platform Instance → Platform Instance Architectural Space

Platform Instance Architectural Space → System Platform
Idea

- **Platform:**
  - a family of architectures satisfying a set of constraints imposed to allow the reuse of hardware and software components.

- **Platform-based design:**
  - meet-in-the-middle approach: In the top-down design flow, designers map an instance of the upper platform to an instance of the lower, and propagate design constraints.
  - exposing key resource limitations
  - hiding inessential implementation details

[Sangiovanni-Vincentelli2002]
Platform-Based Design

**Top-Down:**
Map an instance of the upper platform onto an lower platform considering appropriate constrains.

**Bottom-Up:**
Find the appropriate platform levels.
Define platform level parameters

Platform instances
Platform abstraction levels
III The Development Life Cycle

III.1 Software Engineering Life Cycle Models
III.2 System Engineering Life Cycle Models
III.3 Embedded System Life Cycle Models
III.4 Advanced Life Cycle Models & MDD

III.5 Process Improvement

III.6 Discussion & Summary

III.7 Bibliography
Systems Product Lifecycle

- Conceptual
- Definition
- Production
- Operational
- Divestment

CONCEPTUAL
DEFINITION
PRODUCTION
OPERATIONAL
DIVESTMENT

ROI
BREAKEVEN POINT
Process Management

Why?
- The quality outcome and timeliness of the system development is highly influenced by the quality of the process used to acquire, develop, and maintain it.

Common Misconceptions
- I don’t need process, I have
  - really good people
  - advanced technology
  - an experienced manager
- Process
  - interferes with creativity
  - equals bureaucracy + regimentation
  - isn’t needed when building prototypes
  - is only useful on large projects
  - hinders agility in fast-moving markets
  - costs too much

http://www.sei.cmu.edu/cmmi/general/general.html
The CMMI Project

The CMM Integration Project was formed to:

- Establish a framework to integrate current and future models
- Build an initial set of integrated models
- CMMI models that cover both systems engineering and software engineering might best be described as "engineering models." They are intended to cover the enterprise and include all the processes that result in products or services.

- The source models for the CMMI include:
  - Software: CMM for Software v2.0 Draft C,
  - Systems Engineering: EIA – 731 Systems Engineering
CMMI Model Representations

Staged

ML5
ML4
ML3
ML2
ML 1

Organization

Continuous

Capable

Process

PA
PA
PA
<table>
<thead>
<tr>
<th>Level</th>
<th>Process Characteristics</th>
<th>Management Visibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimizing 5</td>
<td>Focus is on continuous quantitative improvement</td>
<td><img src="image1" alt="Diagram" /></td>
</tr>
<tr>
<td>Quantitatively Managed 4</td>
<td>Process is measured and controlled</td>
<td><img src="image2" alt="Diagram" /></td>
</tr>
<tr>
<td>Defined 3</td>
<td>Process is characterized for the organization and is proactive</td>
<td><img src="image3" alt="Diagram" /></td>
</tr>
<tr>
<td>Managed 2</td>
<td>Process is characterized for projects and is often reactive</td>
<td><img src="image4" alt="Diagram" /></td>
</tr>
<tr>
<td>Initial 1</td>
<td>Process is unpredictable, poorly controlled, and reactive</td>
<td><img src="image5" alt="Diagram" /></td>
</tr>
<tr>
<td>Level</td>
<td>Process Characteristics</td>
<td>Predicted Performance</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------------------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Optimizing</td>
<td>Focus is on continuous quantitative improvement</td>
<td></td>
</tr>
<tr>
<td>Quantitatively Managed</td>
<td>Process is measured and controlled</td>
<td></td>
</tr>
<tr>
<td>Defined</td>
<td>Process is characterized for the organization and is proactive</td>
<td></td>
</tr>
<tr>
<td>Managed</td>
<td>Process is characterized for projects and is often reactive</td>
<td></td>
</tr>
<tr>
<td>Initial</td>
<td>Process is unpredictable, poorly controlled, and reactive</td>
<td></td>
</tr>
</tbody>
</table>
III The Development Life Cycle

III.1 Software Engineering Life Cycle Models
III.2 System Engineering Life Cycle Models
III.3 Embedded System Life Cycle Models
III.4 Advanced Life Cycle Models & MDD
III.5 Process Improvement

III.6 Discussion & Summary

III.7 Bibliography
III.6 Discussion & Summary

- We have nearly the same life cycle models in the different disciplines.

- Advanced life cycle models and model-driven approaches try to increase the degree of automation and decrease time-to-market.

- Especially for organizations which develop large-scale software-intensive systems process improvement is crucial.
III.7 Bibliography (Additional ones)


