Software Architecture Verification at MR

Architecture Improvement during the Race

René L. Krikhaar
Rene.Krikhaar@philips.com

Philips Medical Systems
Magnetic Resonance
Best, the Netherlands
Overview

• Introducing myself
• Medical System: Magnetic Resonance
• Developing (SW) an MR system
• Software Architecture Verification
• Development Process
• Conclusions
Overview

• Introducing myself
• Medical System: Magnetic Resonance
• Developing (SW) an MR system
• Software Architecture Verification
• Development Process
• Conclusions
Introducing myself

• 1986: MSc. Computer Science KU Nijmegen
• 1994: MSc. Knowledge Engineering Uo Middlesex
• 1999: PhD. Computer Science Uv Amsterdam

Philips:
• 1987: VLSI Testing Software Engineer P-ASIC
• 1991: Logic Synthesis Software Engineer ED&T
• 1994: Research Scientist PRL-Eindhoven
• 1999: Software Architect MR Scan Software
Software Architecture Reconstruction

Framework
- Described Architecture
- Redefined Architecture
- Managed Architecture

Software Architecture Verification
Overview

• Introducing myself
• Medical System: Magnetic Resonance
• Developing (SW) an MR system
• Software Architecture Verification
• Development Process
• Conclusions
Magnetic Resonance system

What?
• (diagnostic) medical images

How?
• Magnetic field
• RF signals (receivers and transmitters)
• Gradient
Functional Areas

Radiology

Cardiology

Interventional
Neurology
Angiography
Functional Brain
Overview

• Introducing myself
• Medical System: Magnetic Resonance
• Developing (SW) an MR system
• Software Architecture Verification
• Development Process
• Conclusions
Product Characteristics

• High Tech Product: on the edge of possibilities in (MR) physics
• Each 0.5-1.0 yr new MR Products / Release
  – new functionality (e.g. SENSE)
  – new hardware (e.g. CPU, RF amplifier)
• Parallel Development
  – Multiple Projects
“Complicating” Factors for Development

• Large System
  – more than 3 MLOC (Lines Of Code)
  – many sw/hw developers (also multi-site)
  – third party software/hardware

• Many products in MR Family
  – deriving variants

• Incremental Development
  – includes code written 20 years ago
Making Life Easier -1-

- *Daily-Build-and-Smoke-Test* (since 1984)
Making Life Easier -2-

- Define Coding Standards (since 1985)
- Enforce Check Coding Standards (since 1990)
- Improve Code for Coding Standards (since 1994)

Code Architecture Verification
Scoping

LOGGING is in the scope of ACQ

Building Block ACQ

File in ACQ can include a file in LOGGING

Building Block LOGGING
Making Life Easier -3-

- Define **Scoping** rules (since 1988)
- Enforce **Scoping** rules (since 1990)
- Improve **Scoping** rules (since 1994)

Module Architecture
Verification
What did we achieve?

• Improvement of code comprehension
  – coding standard
  – scoping
• Reduction of coding errors
  – coding standard
• Incremental Testing
  – scoping
• Easier introduction of an OSAL
  – scoping
Overview

• Introducing myself
• Medical System: Magnetic Resonance
• Developing (SW) an MR system
• Software Architecture Verification
• Development Process
• Conclusions
Software Architecture Verification

- **Software Architecture Verification** is the process of revealing deviations between intended and actual software architecture (achieving *architecture conformance*)

- **Intended Software Architecture**
  - In architect’s mind, architectural documents

- **Actual Software Architecture**
  - Implementation (i.e. source code)
Building Blocks

– A functional unit of the MR system.
– Building blocks are hierarchically organized, meaning that a building block may consist of a number of building blocks.
MR System

Acquisition → Reconstruction → Viewing → Archiving

More Explicit Interfaces Required

Platform

Operating System
Why interfaces?

• Separation of concerns
  – maintenance
  – new employees
  – development (planning & tracking)
  – testing
  – parallel development
  – product variants
  – outsourcing
  – ...

Let’s make things better.
Building Blocks and Interfaces

FileIO

Reconstruction

Viewing

Threading

Platform

Logging
Hierarchy in Interface Usage

Reconstruction

FileIO

Threading

Logging

Platform
Hierarchic Interface Rule

Reconstruction

R

Threading

Platform

Let's make things better
Making Life Easier -4-

- Define Interface Management (2000)
- Enforce Interface Management (2002)
- Improve Interfaces (> 2003)

Module Architecture Verification
Managing the Development Process

- Daily Build and Smoke Test
  - quality & stability of code base
- Coding Standards
  - comprehensability of code base
- Scoping Rules
  - complexity of code base
- Interface Management
  - life cycle independency in code base
Overview

- Introducing myself
- Medical System: Magnetic Resonance
- Developing (SW) an MR system
- Software Architecture Verification
- Development Process
- Conclusions
Interfaces in UML

Platform

provides

FileIO

Threading

Logging

FileIO

Threading

Logging

Platform

<interface>>

FileIO

<<interface>>

Threading

CLLOG

f1();

f2();

CLFR

g10();

g20();

provides

Platform

Interfaces in UML
Interface Usage in UML

Reconstruction

Logging Platform

Reconstruction

<<interface>>

Logging

Platform

provides
Interfaces in Code Archive

- mrsystem
  - platform
    - threading (*.h)
    - logging (*.h)
    - fileio (*.h)
    - source (*.cpp)
  - reconstruction
    - source (*.cpp)
Interface Verification

Rational Rose

- Reconstruction
- Platform

<<interface>>
Logging

Provides
platform/logging

ClearCase

- Reconstruction
  - threading (*.h)
  - logging (*.h)
  - fileio (*.h)
  - source (*.cpp)
  - reconstruction
    - source (recon.cpp)

Compile recon.cpp

CC -Iplatform/logging recon.cpp

+ Coding standards
Development Deliverables

“close the chain”

- For each Building Block:
  - Interface Specification (UML)
  - Dependencies / Usage between sub-Building Blocks (UML)
  - Implementation of Building Block
    - source code
  - mr_build command
Overview

• Introducing myself
• Medical System: Magnetic Resonance
• Developing (SW) an MR system
• Software Architecture Verification
• Development Process
• Conclusions
Introduction in Organisation

Preparation phase
1. Define the required architectural rule
2. Define a way to (automatically) enforce it
3. Measure “status” (get a threshold value)

Execution phase
1. Accept violations < threshold value
2. Decrease continuously threshold value
3. Solve rest of violations
Experience -1-

• Introduction on separate development stream
• Code Base analysis not completely okay
  – missing parts of the code base
  – action: fix in a separate action
• Nested include statement
  – crossing subsystem borders
  – action 1: adapt the mechanism
  – action 2: fix in a separate action
Experience -2-

• Hard to find a project that took the ‘risk’
  – be very early
• Deployment in organisation
  – carefully planned and executed
  – accepted by engineers
• New projects starting to use I/f management
  – project control
Why does it succeed at Philips MR?

- Management & Project Support
- Evolutionary Introduction Strategy
- Verification Mechanism
  - automatic verification tools AND
  - embedded in the organisation’s process