Model Driven Engineering
Philips IGT-Systems

Daan van der Munnik
IGT Systems
July 15, 2019
For over 125 years, we have been improving people’s lives with a steady flow of groundbreaking innovations
We strive to make the world healthier and more sustainable through innovation

We’re aiming to improve the lives of 3 billion people a year by 2030
We are on a journey to transform Philips into a health technology leader

Technology
Health technology
Product
Solution
Transaction
Relationship
Addressing the Quadruple Aim

**Improved patient experience**
Improving the patient experience of care (including quality and satisfaction)

**Better health outcomes**
Improving the health of individuals and populations

**Improved staff experience**
Improving the work life of health professionals

**Lower cost of care**
Reducing the per capita cost of healthcare
## Winning propositions

>60% of sales from leadership positions\(^1\)

<table>
<thead>
<tr>
<th>Diagnosis &amp; Treatment</th>
<th>Ultrasound</th>
<th>Global leader</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Image-guided therapy systems</td>
<td>Global leader</td>
</tr>
<tr>
<td></td>
<td>Image-guided therapy devices</td>
<td>Global leader</td>
</tr>
<tr>
<td></td>
<td>Diagnostic imaging</td>
<td>Global top 3</td>
</tr>
<tr>
<td></td>
<td>High-end radiology and cardiology informatics</td>
<td>#1 in North America</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Connected Care</th>
<th>Patient monitoring</th>
<th>Global leader</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ICU telemedicine</td>
<td>#1 in North America</td>
</tr>
<tr>
<td></td>
<td>Personal emergency response</td>
<td>#1 in North America</td>
</tr>
<tr>
<td></td>
<td>Respiratory care(^2)</td>
<td>Global leader</td>
</tr>
<tr>
<td></td>
<td>Sleep care</td>
<td>Global leader</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Personal Health</th>
<th>Male grooming</th>
<th>Global leader</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Oral healthcare</td>
<td>Global leader</td>
</tr>
<tr>
<td></td>
<td>Mother and child care</td>
<td>Global leader</td>
</tr>
<tr>
<td></td>
<td>Domestic appliances</td>
<td>Air, #1 in China</td>
</tr>
</tbody>
</table>

1 Leadership position refers to #1 or #2 position in Philips addressable market. Effective Q1 2019, Philips’ reporting structure changed as per announcement of January 10, 2019.

2 Based on non-invasive ventilators for the hospital setting

Source: GfK, Nielsen, Euromonitor, Frost and Sullivan, Home Healthcare TBS, PCMS market insight
Philips Azurion
Partially Blocked Coronary Artery
Stent in Coronary Artery
Challenges
In system of systems development
#1: Patient (and operator) safety

- X-ray / dose
- Moving geometry
- Live Imaging functionality
  - interventional procedures
- Hand-eye coordination
  - latency

End-to-end CTQ’s (critical-to-quality)
Medical Device Regulations

• External
  – FDA 21 CFR 820 (1) (Quality System Regulation)
    ▪ Submission
    – IEC 62304 (Medical device software – Software life cycle processes)

• Internal
  – Business Management System

• Audits
  – Say what you do, do what you say
  – Prove it, improve it

(1) https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/CFRSearch.cfm?CFRPart=820&showFR=1
System of systems

- Many interconnected subsystems
  - Developed by Philips
  - Contracted out
  - Off-the-shelf
- Specialist teams, different domains & disciplines

- Subsystems brought together during integration and test phase
- Discovering interface problems at this stage is extremely costly
- Often caused by:
  - Ambiguous specifications
  - Undocumented usage
  - Unintended changes to functionality and/or system-level timing

- Relationship between subsystems should be made clear early in design phase
- Requires an unambiguous definition of the interfaces
  - i.e. protocols, timing, data ranges and functionality
Long lifetime

• Support obligation
  – For each system we sell, we implicitly commit to sustain it for **>12 years**

• Hardware lifecycle
  – Fact:
    ▪ Each component shipped as part of the system, has a certain probability to break during the life time of the system
  – Ideal situation:
    ▪ Form fit function replacements / upgrades
      • Replace a FRU without full install or interface failure

Need for: form-fit-function replacements/upgrades
Code size
Basic laws of software maintenance

\[ \text{errors} = (\text{more code})^2 \quad \text{more code} = \text{more engineers} \]

- Each full-time developer can maintain 50KLoc or create 10KLoc per year \(^1\)
- 75% of the total effort is incurred after the first shipment \(^1\)

\(^1\) Wayne Lobb e.a “Software Development and Maintenance Effort/Cost Models”, Foilage whitepaper
Legacy code
(code older than 2 weeks 😊)

• Software deteriorates over time
  – Over-production
    ▪ It takes skills, time and review to create concise software.
  – Over-abstraction
  – Over-processing
  – Unnecessary glue/interface layers

• The maintenance trap
  – More engineers -> more code -> more maintenance -> more engineers
    ▪ Over time teams will slow down to a halt
  – Maintenance = 75+% of SW development cost
Challenges

summary

• Patient safety & Medical device regulation
  – End to end CTQs
  – Evidence required

• System of systems
  – Complexity, integration issues

• Long lifetime
  – Need for form-fit-function replacements/upgrades

• (Legacy) code size
  – More code = more engineers = more $
Approach

- **Strong focus on SW quality**

- **Managing Code-down**

- **Model Driven Engineering**

  - First time right, (prevents expensive releases)
  - Reducing (legacy) code size
  - Addresses both first time right & code size as well as complexity & integration issues
Modeling Roadmap

Categories

- System Level Modeling
- Unit Level Modeling
- Legacy Software Maintenance
- System Level Testing
- Unit Level Testing

Drivers

- Knowledge & Understanding
- Cost & Efficiency
- Quality

Maturity

- Research
- Prototype
- One Off
- Adopted
- Enhance
System Level Modeling

Virtual Cathlab
Clinical workflow modelling and Usability Tests

- Use of partly simulated environment for Usability Testing

- Clinical Workflow modelling to create field based usability test protocols
  - Based on:
    - Machine logging
    - Procedure logging
    - Location data
Unit Level Modeling

• Interface modeling and conformance checking
  – Define, monitor and test interfaces across subsystems
  – “learn” the characteristics of existing interfaces
  – Model-based testing generated from interface models

• Extend the use of MDD using Dezyne
  – Model all new complex state behavior using Dezyne
  – Replace existing manual coded state behavior with Dezyne (where possible)
Interface management

ComMA

- Rigorous interface management
  - Signature
  - Behavior
  - Timing constraints
  - Data constraints
  - Semantics

- Throughout the lifecycle
  - Design
  - Implementation
  - Test
  - LCM
ComMA
Tool suite

• In:
  ➢ Signature
  ➢ Behavior
  ➢ Constraints (timing & data)

• Out:
  ➢ Visualization
  ➢ Documentation
  ➢ Interface proxy code
  ➢ Executable model for simulation
  ➢ Runtime monitoring
  ➢ Model based testing suite
  ➢ Client/server simulator/stub

• Co-owned & created by Philips & TNO-ESI
• **Free** to use and download (for non competitors)
  – We are interested in strengthening the ecosystem
Runtime monitoring

Unit Level Modeling

Technology independent trace language
Executable model for simulation

Interface conformance checking

Model

Constraints

Behavior

generate

Generate

Event

Response

Output

Verdict

Trace

input

Event

Response

Output

Executable model

Unit Level Modeling
Work in progress / Roadmap

- **Test case generation** from interface definitions

- Supporting definition of new interfaces and development
  - Define **Scenarios**
  - **Simulation** of interface
  - Smart **Stubs**

- Modeling existing interfaces - **reverse engineering**

- Generate **formal models** – enable formal analysis

- From interfaces to **components & (sub)systems**
Dezyne
(Verum)

• Model-driven software engineering tool
  – Modeling language
  – Simulation
  – Formal verification
  – Code generation
• For state based, event driven or concurrent software systems
• Used in:
  – ACQ subsystem (ASC/BEA)
  – iEngine (roadmapping, service)
Legacy Software Maintenance

• Software **rejuvenation** using compiler techniques
  – Model extraction from legacy components to improve human understanding
  – **Static** dependency visualization & refactoring tooling

• Re-engineering **dynamics** of legacy software towards MDSE
  – By analyzing source code (model extraction)
  – By analyzing runtime traces (model learning)
Rejuvenation

*Static* dependency visualization & refactoring

- Develop scalable techniques for improving software quality (architecture, code, etc.)

- Three step approach:
  - Extract
  - Analyze
  - Transform
Extract

• The extraction tooling extracts facts from:
  – Build infrastructure related files, e.g., Visual Studio solutions/project files
  – Source files, e.g., C/C++/C# files and COM-related (.idl) files
  – File systems, e.g., the disk layout of the source tree

• The extracted model is stored as a graph database
Analyze

yEd

- Customizable visualizations of dependencies
Analyze Neo4J

- Ad-hoc queries
Analyze
Design Structure Matrix

<table>
<thead>
<tr>
<th>$\text{root}$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>U3</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>13</td>
<td>11</td>
<td>13</td>
<td>13</td>
<td>11</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>U2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
<td>2</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>10</td>
<td>2</td>
<td>2</td>
<td>9</td>
<td>4</td>
<td>13</td>
<td>13</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>P5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1</td>
<td>8</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D2</td>
<td>10</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>11</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A6</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A4</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A3</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>5</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Hierarchy:

- Utility
  - P5
  - P4
  - P3
  - P2
  - P1

Presentation:

- Data
  - D3
  - D2
  - D1

Application:

- Dependencies:
  - U3 to U1
  - U2 to U1
  - P5 to D3
  - P4 to D3
  - P3 to D2
  - P2 to D2
  - P1 to D2

Identity line:

Same rows and columns:

- Identical rows and columns in the matrix.
Transform

- Use-case
  - Remove .cpp “group” files including other .cpp files (faster builds, easier navigation, code-down)

- Results
  - Significant improvement in productivity (4 hours vs. 1 week)
  - Amount of repetitive work is reduced
  - Part that requires creative thinking remains

- Also using queries for architecture guarding
  - Integrated in CI/CD pipeline
Unit & System Level Testing
From document based...

1. Concept
   - Spec.doc
   - Manually

   DSL
   - TorXakis MBT
   - Confidence
   - Evidence
   - SpecFlow
   - Gherkin
   - Feature: ...
   - Scenario: ...
   - Given ...
   - When ...
   - Then ...

   Test Res.doc

   C++

   SUT
Unit & System Level Testing
To model based...

1. Spec .doc

Spec Model DSL

generate

Execute

Generate

Confidence

Evidence

TorXakis MBT

SpecFlow Gherkin

Feature: ...
Scenario: ...
Given ...
When ...
Then ...

Test Res .doc

Concept

C++

SUT

System Level Testing

Unit Level Testing

Spec Model DSL

DSL
Summary

• Model Driven Engineering helps us at all levels to:
  – Increase Quality
  – Manage Complexity & Integration issues
  – Reduce & Manage legacy code

• Addressing the Quadruple Aim:

  Improved patient experience
  Better health outcomes
  Improved staff experience
  Lower cost of care