MANAGING COMPLEXITY IN HIGH-TECH SYSTEMS

Research at ESI

Wouter Leibbrandt
Science and operations director

7 November 2018
Engineering of complex systems has been done for ages. E.g. in the Netherlands.
Some complex systems today (from The Netherlands)

Semiconductor manufacturing equipment
Medical systems
Agricultural robots
Combat management systems
Electron microscopes
Building control
Robotized warehousing
Industrial printers
Automotive
Some complex systems today (from The Netherlands)

- Semiconductor manufacturing equipment
- Medical systems
- Agricultural robots
- Combat management systems
- Electron microscopes
- Building control
- Robotized warehousing
- Industrial printers
- Automotive
Common challenges

• System complexity
  - Heterogeneity, interdependencies
  - Performance: nm accuracy, strict timing, exact dosage, ...
  - Human factors & Business factors
  - Scale: e.g. 100 Mln lines of code

• Formal systems approach across the lifecycle

• Abstraction
Mastering System Complexity across the abstraction levels

Challenge 1
Abstract but:
- Concrete
- Accurate
- Trustworthy
- Specific

Challenge 2
Seamlessly move from abstract to details for the various aspects

Analysis
Synthesis

Challenge 3
System approach across aspects: relations trade-offs...

Abstractions in engineering are models and feature at all levels of abstraction

Larger Systems
Systems
Multidisciplinary design
Parts, connections, lines of code

Number of details

More details

Abstract
Vague
Detailed
Concrete
A broader view on systems

- market, wider context
- enterprise
- stakeholders
- systems
- multidisciplinary design
- parts, connections, lines of code

Number of details
Models, models, models, ...

Know What You Don’t Know

Know What You Know

Model-Based Systems Engineering

Generic
Descriptive
Explorative

Domain specific
Analytic
Detailed
1459
View of the world
before the age of discovery

1525
View of the world
in the age of discovery

“we know everything”

“so much we do not know yet”

From: Yuval Noah Harari - Sapiens
ESI at a glance

Mission: “To advance industrial innovation and academic excellence in high-tech systems engineering”

Synopsis
- Foundation ESI started in 2002
- ESI acquired by TNO per January 2013
- ~60 staff members, many with extensive industrial experience
- 5 Part-time Professors
- Working at industry locations
- From embedded systems innovation to embedding innovation
- TNO program turnover 2018: 10Mio €

Partner Board

Location
Open Innovation

Full access to industry background

Generic results

Industry specific validation results
From industry challenge to ESI expertise

Industry challenge is addressed from three (inter-related) angles:
1. Business innovation drivers
2. Industry characteristics
3. Continuous increasing complexity
1. Changing business innovation drivers

- Competition in total cost-of-ownership
  - Selling catalogue items
  - Focus on total cost of ownership
  - Customer-specific products/configuration
  - Integration in application context

- Competition in end-user integration

Total life cycle business

Box selling

Solution selling
1. Innovation drivers and associated knowledge needs

**Innovation driver**
- Selling catalogue items
- Focus on total cost of ownership
- Customer-specific products/configuration
- Integration in application context

**Knowledge needs**
- Key Product Qualities
- Quality & dependability
- Variability and ease of change
- Integration and services
2. High-tech industry characteristics

- **Globalization** has introduced a change in customer mix and entries into emerging economies. **Global diversification** has become yet another challenge.

- The success of high-tech companies today increasingly depends on providing services, content and interoperability around their products (open systems, solution selling).

- A significant part of the Dutch (but also European) high-tech industry has (re)positioned itself in the **higher value segment**.

- Competitive advantages gained from innovations are increasingly **short-lived**.

- **Total life-cycle business**, i.e. the sales of new products combined with services to installed products, is an ever more important revenue model.

- Companies are increasingly dependent on **partnering within the industry ecosystem** to develop and introduce new products and solutions. **Effective collaboration** among globally dispersed design teams and with external partners has become a necessity.
### 2. Industry characteristics and associated needs

<table>
<thead>
<tr>
<th>Industry characteristic</th>
<th>Knowledge needs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global product diversification</td>
<td>Variability and ease of change</td>
</tr>
<tr>
<td>Interoperable systems/solution selling</td>
<td>Integration, security, focus on providing services</td>
</tr>
<tr>
<td>Positioning high value segment</td>
<td>Key product qualities</td>
</tr>
<tr>
<td>Short-lived competitive advantages</td>
<td>Development efficiency and reduced lead-time</td>
</tr>
<tr>
<td>Total life-cycle focus</td>
<td>Deal with continuous hardware and software evolution</td>
</tr>
<tr>
<td>Partnering and collaboration</td>
<td>Multi-disciplinary, multi-site, multi-team approaches</td>
</tr>
</tbody>
</table>

*Already covered*
3. Continuous increasing complexity
What is driving complexity in the next decade?

- **Continuously evolving systems**
  - The system has both the capability and the need to evolve over time

- **Series-of-1 Systems**
  - High level of customization of each system delivered, no two systems alike

- **Systems of Systems**
  - The system is an integral part of a larger system without any control of that system

- **X-as-a-service Systems**
  - Not the machine, but the service it provides is the manufacturer’s value proposition

- **Autonomous Systems**
  - System operates without human in the loop, human interaction moves to higher level

- **Parameters-times-10 Systems**
  - All current design parameters (e.g. interfaces, LoC) get one (or more) orders more demanding
3. Continuous increasing complexity – the need

**Complexity drivers**
- Continuously evolving systems
- Series-of-1 Systems
- Systems of Systems
- X-as-a-service Systems
- Autonomous Systems
- Parameters-times-10 Systems

**Knowledge needs**
- Manage cross-disciplinary uncertainty
- Bridge gap between multi-disciplinary architecting & design and mono-disciplinary realization
- Raise abstraction level at which (sub)systems are specified, explored, analyzed and synthesized, integrated and tested
Scope

Multi-disciplinary system architecting and design
- Needs
  - Specification
  - System design
  - Subsystem design

Mono-disciplinary realization
- Component design
- Component test
- Realization
- Subsystem test
- Verification
- Validation

Integration & test

In-context operation

An initiative of industry, academia and TNO
Choice for ESI programme lines

- Selling catalogue items
- Focus on total cost of ownership
- Customer-specific products/configuration
- Integration in application context

Key Product Qualities
- Quality & dependability
- Variability and ease of change
- Integration and services

ESI Programme lines

1. System Performance
2. System Dependability
3. System Evolvability
4. Exploiting System Context
5. System Architecting

Development efficiency and reduced lead-time
Deal with continuous hardware and software evolution
Multi-disciplinary, multi-site, multi-team approaches

Manage cross-disciplinary uncertainty
Bridge gap between multi-disciplinary architecting & design and mono-disciplinary realization
Raise abstraction level at which (sub)systems are specified, explored, analyzed and synthesized, integrated and tested
# ESI expertise teams

## System Overview

<table>
<thead>
<tr>
<th>Architecting</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecting process</td>
<td>Reference architectures</td>
</tr>
<tr>
<td></td>
<td>Architectural patterns</td>
</tr>
</tbody>
</table>

## Aspect focused analysis and synthesis at system and sub-system level, design time and runtime

<table>
<thead>
<tr>
<th>System architecture systematics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
</tr>
<tr>
<td>Quality assurance and testing</td>
</tr>
<tr>
<td>Dependability and robustness</td>
</tr>
<tr>
<td>Adaptivity and evolvability</td>
</tr>
<tr>
<td>Dealing with legacy</td>
</tr>
<tr>
<td>System security</td>
</tr>
</tbody>
</table>

## Model and reasoning based System Engineering

<table>
<thead>
<tr>
<th>Data-driven MBSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data &amp; knowledge integration</td>
</tr>
<tr>
<td>Model forming</td>
</tr>
<tr>
<td>Model languages</td>
</tr>
<tr>
<td>Model management</td>
</tr>
</tbody>
</table>

## ESI Expertise Areas

- **System architecture systematics**
- **System performance**
- **Quality Assurance & testability**
- **Adaptability and evolvability**
- **SW and System rejuvenation**
- **System security**
- **Domain Specific Languages**

---

**Dependability and robustness:** will be a future ESI expertise team once we have build up expertise that can be shared over projects.

**Model management:** topic is being discussed and no decision has been made yet.
2018 project portfolio

Drivers
- Continuously Evolving Systems
- Series-of-1 Systems
- Autonomous Systems
- Systems of Systems
- X-as-a-Service Systems
- Parameters-times-10 Systems

System Overview
- Architecture
  - Architecting process
    - Reference architectures
    - Architectural patterns
  - Aspect focused analysis and synthesis at system and sub-system level, design time and runtime
    - Performance
    - Quality assurance and testing
    - Dependability and robustness
    - Adaptivity and evolvability
    - Dealing with legacy
    - System security

Model and reasoning based System Engineering
- Data & knowledge integration
- Model forming
- Model languages
- Model management

ESI programme lines
- System performance
- System dependability
- System evolvability
- Exploiting system context
- System architecting
PROJECT 1: IOT DSL

Program: System Evolvability
✓ 6,000 luminaires
✓ 3,000 sensors
✓ 100 motions per day/sensor
✓ 0.1 reboot per day
✓ 5 second boot time
✓ 3/1000 message loss
Domain Specific Language (DSL)

- Easy language to model in a well circumscribed domain
  - Design
  - Generation of code and documentation
  - Testing
  - Commissioning
  - Behaviour and root-cause analysis
  - Reconfiguration
- Originally built for lighting domain
- Generalized to Internet of things:
  - Many connected devices
  - Few different behaviours
IoT DSL Suite: Customizable for domain

Domain DSLs
- Structure DSL: Topology elements and their containment
- Event DSL: Event types with their payloads

System DSLs
- Topology DSL: Physical system topology using domain-specific topology elements
- Behavior DSL: Parameterized behaviors with typed inputs/outputs
- Control DSL: Instantiated control behaviors coupled via their outputs/inputs

Validation DSLs
- Scenario DSL: Control inputs from physical system (sensors)
- Control DSL: Instantiated control behaviors coupled via their outputs/inputs

Multiple controllers
- System topology
- Control network
- System config

Customizable domain
- Stacked control

JCoSim
Interactive sim.

Uppaal
Model checking

An initiative of industry, academia and TNO
PROJECT 2: MODEL BASED SYSTEM ARCHITECTURE
Program System Architecting
System architect role

- Transformation of business strategy and stakeholder purposes to product qualities and technical solutions

Responsibilities of the System Architect

What does the System Architect do?

- think, analyze
- listen, talk, walk around
- design, brainstorm, explain
- assist project leader with work breakdown, schedule, risks
- present, meet, teach, discuss
- test, integrate
- write, consolidate, browse
- read, review
- travel to customer, supplier, conference
- provide vision and leadership
In the squeeze...

• Symptoms

  - Many meetings
  - Long working hours
  - Poor leadership
  - Incomplete
  - Uncertainty
  - Decisions delayed

• How to

  - get system overview? → *"Too much for one brain!"
  - keep a consistent system representation? → *"Many, many stakeholders"
  - understand consequences of decisions? → *"Too many dependencies"
  - architect systematically? → *"No concrete, operational methods"
Model-based system architecting (a methodology)

- Organizing...
  - cooperation in team
  - ownership
  - transparency
  - systematics
  - abstraction
  - formalization
- Change in way-of-working: from informal language to models

Team process

Architecting process

Architectural reasoning

1) Digitized
2) Executable
3) Coupled

Bla bla

~ $10^0$

~ $10^1$

~ $10^2$

~ $10^3$
Organizing the models

C, A
Stakeholder Views

F
System View

C
Architectural Views

R
Realization Views

Product Manager Marketing
Senior Management
Customer

BB1
BB2

Cost
Throughput
Decomposition

Stakeholder Views:
- Throughput
- Cost

System View:
- Customer
- Senior Management
- Product Manager Marketing

Architectural Views:
- BB1
- BB2

Realization Views:
- Cost
- Throughput
- Decomposition

Realization Views:

C, A

Stakeholder Views

~ 10^1

F

System View

~ 10^0

C

Architectural Views

~ 10^1

R

Realization Views

~ 10^2
Wind Farm

Cost

location

Societal
acceptance

Infra-
structure
PROJECT 3: LEGACY CODE REJUVENATION
Program: System Evolvability
Use Case: COM Clean-Up

- Yellow: Visual Studio Project
- Blue: IDL file (COM Interface Definition File)
- Green: Interface Names

Target: IDL file is compiled in 1 Visual Studio project

Pictures have been created with Neo4j Graph Database using graphs created from software using the Renaissance Tools

(≈6MLOC)
SUMMARIZING ...
An initiative of industry, academia and TNO
THANK YOU!