ITEA Smart Systems Engineering workshop

Session I - Complexity of the applications
ITEA Smart Systems Engineering workshop

7 April 2022 | online
Harald Schöning, Software AG
Systems engineering vs. increasing complexity
Just an illustration

- IoT (lot of heterogeneous devices)
- Cloud
- Edge Computing
- Bandwidth?
- Connectivity?
- Energy efficiency?
- Data Space?
- AI
## Functional and non-functional complexity

<table>
<thead>
<tr>
<th>Functional</th>
<th>Non-Functional</th>
</tr>
</thead>
<tbody>
<tr>
<td>E.g. Distribution optimization</td>
<td>Energy efficiency</td>
</tr>
<tr>
<td>Computing Capacity</td>
<td>Ethics</td>
</tr>
<tr>
<td>Latency</td>
<td>Security</td>
</tr>
<tr>
<td>Dealing with load peaks</td>
<td>Easiness of operation</td>
</tr>
<tr>
<td>Bandwidth limitations</td>
<td>...</td>
</tr>
<tr>
<td>Distributed Learning</td>
<td></td>
</tr>
</tbody>
</table>

### Combining IT and physical devices

**Digital Twins**

**Safety**
What do we need?

**Tools**
- Design
- Build/Generate?
- Test
- Monitor&Operate

**Education**
Covering complex system handling
- academic
- On the job

**Interdisciplinarity**
Within computer science disciplines
But also with many other disciplines
ITEA Smart Systems Engineering workshop

7 April 2022 | online
Philippe Dobbelaeere, Nokia Bell Labs
Introduction slide

Background experience

- Msc engineering and physics
- Bell Labs 3y – 5y research on HW, SW and systems problems
- 15 y of HW and SW product development (DSL, core routers)
- 15 y of applied research in IoT cooperative projects
  - ITEA DiY Smart experiences
  - ITEA M2MGrids
  - ITEA MOS2S
Complexity has always been a topic in science / industry…

- “There are two ways of constructing a software design. One way is to make it so simple that there are **obviously no deficiencies**. And the other way is to make it so **complicated** that there are **no obvious deficiencies.**” C.A.R. Hoare
- KISS (airforce engineering)
- Bjarne Stroustrup (C++) "Make Simple Tasks Simple!“
- Einstein paraphrased: "Everything should be made **as simple as possible**, but **not simpler**"

\[ \text{design complexity spawns mistakes } \leftrightarrow \text{ if the problem is complex, a simple solution is probably not going to work (but maybe the subcomponents can be “simple”) } \]
trends...

nomadic computing → need to migrate functionality (VM concept helps)
clouds make it cheap/easy to "install" applications that behave as long running "services"
AI/deep learning makes systems autonomous

cyber-physical systems
actuation is the enabler
Concerns in contemporary IoT design and engineering
heterogeneous environment

nature of heterogeneity is changing
  - used to be single computer → cluster of computers
  - ("pets" with different capabilities and roles)
  - but this has migrated to IaaS cloud based on hypervisors/VM
  - ("cattle")

heterogeneity now comes from end-to-end compute architectures, all the way from edge devices, gateways, edge and core routers to SaaS datacenters and public and private IaaS clouds
IoT in real projects today…

- edge devices such as sensors, actuators, gateways with some level of compute resources (if not at the sensor / actuator, then on the gateway)
- multiple organisations with their own, underspecified concepts and interfaces ("vendor specific API") for data exchange and control functionality
- a melting pot of (almost standard compliant) networking technologies
  - a requirement to connect data sources, data sinks and processing logic together, regardless of network technology ("data broker")
  - less than rock-solid round-trip latencies (~ 2s = no TCP...)
- a mix of communication protocols
- a SW architecture containing legacy components that were created at a time where elasticity was a concept of mechanical designers, not SW systems and process communication was RPC (or evolutions such as CORBA, SOAP based WS, ...)
- private or public VM based IaaS cloud infrastructure to run the majority of the software functions
  - data brokering
  - data analytics (real-time or offline)
  - backend application
    - actuation support, business logic
    - management layers
Multiple programming paradigms
when to use them, how to use them together

- imperative
  - C(++), Fortran, Go, Java, lua, perl, python, ruby, matlab, javascript, ...

- functional
  - Haskell, C++11, javascript, erlang, java8, lisp, scala

- declarative
  - prolog
  - table based DB (SQL)
  - document based DB (rethink, mongo)
  - tuple based DB (SPARQL, quadstore)

- rule based
  - prolog, CHR, drools

- linear algebra / array processing
  - torch, matlab, R, fortran (linpack,...)
system decomposition implies the need for an IPC mechanism

**RPC** is problematic
- no differentiation between client stub crashing and remote server crashing
- requires argument list construction: one monolithic memory copy at the server (cfr XML DOM/SAX parser)
- typically blocking, synchronous calls - timeout?
- reply always goes back to the requesting client
- difficult to make compatible with transactional semantics (who owns a transaction that started on the client?)
- tightly coupled, sensitive to interface creep (and SW guys like bleeding edge...)

**message passing** preferred
- typically asynchronous, event driven
- message handling scales extremely well
- inherent point-to-multipoint capability
- offers at least once semantics in a very natural way
- messaging can easily handle transactional semantics (e.g. Kafka: write all messages in batch or drop all of them)
- if something RPC style is needed, trivial to code, with required flexibility and error handling, on top of messaging infrastructure
  - e.g. OTP template for erlang servers on top of actor model
- can do distributed error handling
- typically lightweight protocol stacks, minimalistic state that just fits on IoT devices
- an application becomes a directed dataflow graph (HW guys would call this a netlist)
designing and testing for a 3rd party operated cloud
e.g. C-ITS backend logic by Nokia deployed on AWS WZ datacenter of Vodafone
see https://tv.theiet.org/index.html?videoid=15510

challenges:
• authentication complexity
  • SSO into 3rd party infrastructure (“jumpserver”)
  • SSO into kubernetes control layer (time limit)
• deployment complexity
  • setup pods / services on provided k8s resources
  • setup networking to allow application data communication
  • setup networking to allow application OAM
• OAM complexity
  • wire logging / resource monitors into 3rd party kibana / grafana stack

extremely “hostile” environment to do anything except finalised app deployment
value chain complexity

• makes it difficult for companies to commit design resources, due to “complicated” business plan outcomes
• makes it difficult for stakeholders to invest

e.g. road safety
- Nokia can design SW infrastructure, but who is going to pay for it?
- less people are killed, but people do not directly fund infrastructure
- a road might be “owned” by central government, but local government is suffering the problems
ITEA Smart Systems Engineering workshop
Contact details

- Philippe Dobbelaere
  philippe.dobbelaere@nokia.com
- +32 474 860063

- Copernicuslaan 50
  B-2018 Antwerpen
  BELGIUM
Thank you for your attention
ITEA Smart Systems Engineering workshop

7 April 2022 | online
Oliver Lenord, Robert Bosch GmbH
**Introduction slide**

**Background experience**

**Researcher and Project Leader of PFPs**
- Bosch Corporate Research\(^1\) (Renningen, Germany)
- Dependable Cyber-physical Systems Engineering, Model-based Development

**CAE Product Manager**
- SISW (Cypress California, USA)
- Mechatronics Concept Design and Systems Engineering

**Leader of Simulation Software Development**
- Bosch Rexroth (Lohr a.M., Germany)
- Sales enabling Simulation-Tools

**PhD Mechatronics**
- Gerhard-Mercator-University (Duisburg, Germany)
- Biologically inspired Virtual Prototype of 4-legged walking machine

\(^1\) 1740 employees, 90% scientists

PFP: publicly funded project
Introduction slide

Background experience

- **BMWi** ¹) **PHyMoS** (2021 – 2024)
  - proper hybrid models

- **ITEA3 15016 EMPHYSIS** (2017 – 2021)
  - from physics models to embedded software

- **ITEA2 08021 OPENPROD** (2009 – 2012)
  - wholistic model-driven product development

¹) BMWi: German Federal Ministry for Economic Affairs and Climate Action
Session topic…

Key challenges

- **Manage Complexity**
  - by abstraction
  - by graphical representations
  - by modeling languages

- **Support the Engineering Process**
  - from idea to concept to functional models to physical design
  - from whole system to subsystem to component
  - from behavior to algorithms to target code

- **Enable Collaboration**
  - virtualization: hybrid (physics + data) driven modeling
  - heterogeneous environments: model exchange, traceability, meta data
  - confidence and trust in simulation: model quality considering uncertainties

---

1) Source: “hPLM Workstream GlueParticle” by SmartSE, Hans-Martin Heinkel (Bosch)
Collaborative System Development Between Partners

SmartSE Vision

Establish best practices for distributed collaborative system development between partners using Systems Engineering methods and standards.

Mission Statement

Phase 5

Enabling collaborative development and validation of complex products by simulation along a multi tier supply chain.

Source: “hPLM Workstream GlueParticle” by SmartSE, Hans-Martin Heinkel (Bosch)
Session topic…

Key challenges

- **Methods to enable Transition between Levels of Abstraction**
  - “Model on demand”
  - define requirements on a “proper” model
  - automate model transformation to generate “proper” models
  - empower V&V to assess model quality

- **Bring the Pieces Together**
  - “Credible Simulation Process” (SmartSE\(^1\))
  - develop standards and tools
  - connect/enhance existing standards (FMI, eFMI, SSP,...)

\(^1\) Smart Systems Engineering: Project of the ProStep IVIP since 2012
ITEA Smart Systems Engineering workshop
Contact details

- Oliver Lenord, Robert Bosch GmbH, oliver.lenord@de.bosch.com
Thank you for your attention
Introduction slide Jonathan Menu

Background experience

Research manager for Simcenter MBSE @Siemens

Background: MSc + PhD in (astro)physics; at Siemens since 2015

Personal involvement in ITEA projects:

- Reflexion (2015-2019): React to effects fast by learning, evaluation, and extracted information
- REVaMP² (2016-2019): Round-trip engineering and variability management platform and process
- EMBrACE (2019-2022): Environment for model-based rigorous adaptive co-design and operation of CPS
- OXILATE (2020-2023): Operational excellence by integrating learned information into actionable expertise

Siemens Industry Software NV

- Engineering innovation partner
- Simulation & test tool provider
- 450 employees in Leuven
- ITEA founding company

Reflexion (ITEA Award of Excellence 2019)

REVaMP², EMBrACE, OXILATE
Session I - Complexity of the applications

Key challenges

Radical changes to designs required:
- Regulations (e.g., climate neutrality, environmental footprint)
- User expectations: automation, adaptability, performance, availability, response time
- Other “ilities” and/or constraints: cost, security, safety

Tool provider:
what we want to design impacts how we design
Session I - Complexity of the applications

Key challenges

Pathways to solutions:
✓ Generative techniques
✓ Correctness-by-design
✓ Usability & decision support

Generative engineering:
using reasoning and ML techniques to create conceptual alternatives

User assistance built into engineering tools
(OXILATE ITEA project)

Formal requirements linking with design and V&V
(EMBrACE ITEA project)
ITEA Smart Systems Engineering workshop
Contact details

Dr. Jonathan Menu
Research Manager

Siemens Industry Software
Digital Industries Software
Simulation and Test Solutions
Interleuvenlaan 68
3001 Leuven
Belgium
T: +32 16 38 43 69
jonathan.menu@siemens.com
Thank you for your attention