Software Architecture for Automotive

Patrizio Pelliccione
Associate Professor (Docent), Chalmers|GU

www.patriziopelliccione.com
• Architectures and Architecture description
  • The state of architecture descriptions in industry
  • Architecture Description Languages
  • Standards ISO/IEC/IEEE 42010 - Architecture descriptions
  • Architecture Framework of Volvo Cars
Learning Objectives

• After this lecture, you should be able to…
  • Argument about the challenges of producing architecture descriptions
  • Describe which formalism might be used to describe a software architecture of a complex system
  • Describe what an Architecture Description Language is
  • Describe the main characteristics of an architecture framework
Architecting in practice...

- **Development**
  - Feature
  - Function
  - Electrical Architecture
  - System
  - Component
- **Verification**
  - Parallel development: suppliers + In-house development
  - “Big-bang” integration!
  - Collaboration based on contracts

Architecture description

Implementation
Architecting in practice...

Architecture degradation

- **Architectural drift** – discrepancies that do not violate any decision that is documented in the as-intended architecture

- **Architectural erosion** – some decisions violate the as-intended architecture

Discrepancy between as-intended and as-implemented architectures!
Risk of architectural erosion

- The actual architecture of the car is not exactly the one conceived by the architects
  - The architecture is also emerging during development (bottom-up)
  - Some architectural decisions are made unconsciously
    - Which decisions have an impact on the architecture? – not easy
  - Some “actual” architects do not have the title of architect

Architecture

- Ideas/vision of the system to be realized

GAP

Design

- Actual blueprint for the implementation teams, being used in their daily work, and evolving over time
Different points of view

What the design groups think of the high-level architecture group

- High-level architects lack an understanding of the current situation and the system under implementation.
- High-level architect group focuses too much on what might be good for the future, while neglecting a concrete vision of what is the best solution for the current situation.

What the architecture group thinks of the working architecture group

- The design groups are very focused on everyday problems and then they miss an overall picture.
- The design groups are too focused on short-term solutions: choosing the best solution in the short run might cause problems in the next future.

“But sometimes you feel that the architecture-group thinks that we should change everything. While we [design group] are more focused on that we have to solve something to the project, and yes, what we have is maybe not the optimal solution but it is what we have.”

Ulf Eliasson, Rogardt Heldal, Patrizio Pelliccione, Jonn Lantz, “Architecting in the Automotive Domain: Descriptive vs Prescriptive Architecture”, WICSA 2015, 12th Working IEEE/IFIP Conf. on Software Architecture, Montreal, Canada
Identified antipatterns

- **GoldPlating** – the architecture that has been created is a perfect architecture but it is describing the wrong system.

- **Ivory tower** – the architect team is isolated from the other groups with few communication. They might experience rejection from developers.

- **Architecture watch** – the group of architects is limited to a watching group. They provide recommendations without making any architectural decision.

What do architect really do?

Architecting:
- design
- validation
- prototyping
- documenting
- etc.

50%

Getting input:
- user, requirement
- other architecture
- technology

25%

Providing Information
- communicating architecture
- assisting other stakeholders

25%

External Focus

Internal Focus

What do architect really do?

The [60:30:10] antipattern – goldplating

**GoldPlating** – the architecture that has been created is a perfect architecture but it is describing the wrong system.
What do architect really do?

The [70:15:15] antipattern – ivory tower

Ivory tower – the architect team is isolated from the other groups with few communication. They might experience rejection from developers.

Introvert-Extrovert Dichotomy

Introvert Architect
• Disciplined development
• Design activities
• Capturing design decisions
• Impact on quality attributes, such as cost, safety, evolution, performance, security, etc.

Extrovert Architect
• Communication of architectural decisions and knowledge to other stakeholders
• Collaborate with project teams, customers, etc.

Limitations of the actual architecture description

- Importance varies over time
- Easily becomes out of date
- Too many details
- Variability management
- Should better document the design decisions
- Should better document / make explicit the assumptions made
- Should be a living document connected with the other development phases
- Should handle different views and viewpoints of different stakeholders’ concerns
- Present and Future mixed in the same document

State of Practice

Recent work: Architecture Gap Survey

- Involved Volvo Cars (VCG), Volvo Group Truck Technology (VGTT), Ericsson, Jeppesen AB, plus many other companies around the world

- Research questions that we are investigating:
  - Is the architecture driving the development?
  - Is the architecture “emerging” from the development?
  - Is there any gap between what specified in the architecture and what is developed?
  - If so, what are the reasons for that and what are the consequences?
  - How could the architecture description be improved to be more useful during the development and maintenance phases?
Recent work: Architecture Gap Survey

Architecture Survey

The aim of this survey is to understand the role of the architecture and its different views for the development of the system.

Some questions are still open and merit further investigation:

- Is the architecture driving the development?
- Is the architecture “emerging” from the development?
- Is there any gap between what specified in the architecture and what is developed?
- If so, what are the reasons for that and what are the consequences?
- How could the architecture description be improved to be more useful during the development and maintenance phases?

Answering to this survey will require approximately 20 minutes and your responses are completely anonymous.

If you will be interested to result we will send you the result of the survey if you provide us your e-mail below. The e-mail addresses are stored separately and can’t be traced back to any responses.

We really appreciate your input!

If you have any questions, please e-mail us: Ulf Ellasson

☐ I am interested in the results of this study.
Architecture driving the implementation or the other way round?
How could the architecture description be improved?
Main findings of the study

Finding
• Typically, more than one architecture description (often at different levels of abstraction) exists for describing the architecture of a system

Implication
• Several reasons for representing architectures in architecture descriptions
• Need of investigating
  • how to ameliorate the creation and maintenance of architecture descriptions
  • languages and notations to describe architectures
  • how to maintain more than one architecture description for representing the architecture of complex systems
Main findings of the study

Finding
• Architecture descriptions serve **various purposes** and should be conceived for different types of stakeholders

Implication
• Need of multiple views and viewpoints
• Languages and notations to represent architecture descriptions should be conceived with the flexibility to satisfy various concerns of different stakeholders
• The different purposes might also be conflicting each other, and suitable tradeoff analysis should be put in place
Main findings of the study

Finding

• While it is important to have an upfront architecture and architecture description, the **architecture description should evolve** during the system development from input and feedback coming from stakeholders that are even different from architects

Implication

• How much information should be put in the upfront architecture description?
• Need to support “just in time” architecting, thus enabling stakeholders (even different from architects) to refine, add information, or provide feedback to the architecture description.
Main findings of the study

Finding
• We identified some discrepancy between the architect team and other stakeholders

Implication
• There is the risk that within the same company will grow and will become established different cultures and beliefs.
• Innovative and more effective communication means are needed to enable communication among different stakeholders.
Main findings of the study

Finding

• Exist **inconsistencies** both among different architecture descriptions and between architecture descriptions and design/implementation
• Some of the inconsistencies might have high impact

Implication

• This finding triggers the need of investigating causes and mechanisms to discover, avoid, and mitigate inconsistencies
Which formalism to use to describe the architecture of a complex system?
Architecture Standard

• According to the ISO/IEC/IEEE 42010:2011 there are three mechanisms for architecture description:

  1. **architecture viewpoints**: common ways of expressing recurring architectural concerns reusable across projects and organizations;
  2. **architecture frameworks** (AFs): coordinated set of viewpoints for use within a particular stakeholder community or domain of application;
  3. **architecture description languages** (ALs): any mode of expression used in an architecture description.
Vehicle architecture: why?

- Modeling and description of a vehicle architecture is important:
  - A manifestation of the earliest design decisions
  - Prerequisite for a deeper understanding of a vehicle system and an effective reasoning and communication about it

- The Automotive industry can be regarded as a complex network (ecosystem) of highly interdependent organizations which collaborate in all phases of the vehicle lifecycle.

- Because of this characteristic it is necessary for the members of this ecosystem to agree on a somewhat common way for structuring and describing a vehicle in order to increase overall efficiency within the ecosystem.

A car is a complex system

Functional architecture
Logical architecture
Technical architecture
CI&D
Car in a SoS
Ecosystem & Transparency
Autonomous vehicle

Safety
Security
Qualities
Energy
Cost
NHV – Noise, Vibration, Harshness
Weight
Variability

Autonomous vehicle

Hosting
Components of a functional architecture view of an autonomous driving system

A Functional reference architecture for autonomous vehicles

Which formalism to use to describe the architecture of a complex system?

- UML / UML profile?
- Architecture Description Language (ADL)?
Architecture description languages

More than 125!

An ideal and general purpose AL is not likely to exist

Lack of understanding on practitioners’ needs and on what they consider to be useful, useless, or missing in current ALs
Why so many ALs?

• An IDEAL and general purpose ADL is NOT likely to exist
  • Stakeholder concerns are various, ever evolving, and adapting to changing system requirements. [ISO/IEC/IEEE 42010]
  • Difficult to capture all such concerns with a single, narrowly focused notation.

• Architectural languages must be able to focus on “what is needed” by the stakeholders involved in the architecting process.
ARCHITECTURAL LANGUAGES TODAY

The up-to-date list of currently existing architectural languages

http://www.di.univaq.it/malavolta/al/
Survey on ALs – research questions

**RQ1**: What are the architectural description needs of practitioners?

**RQ2**: What features typically supported by existing architectural languages are useful (or not useful) for the software industry?

- Interviewed 48 practitioners from 40 different IT companies in 15 countries (questionnaire of 51 questions)

Most used ALs
Never used ALs, why?

<table>
<thead>
<tr>
<th>Question</th>
<th>Answers</th>
</tr>
</thead>
<tbody>
<tr>
<td>d.27.a) Why you (or your organization) decided not to use ADLs? Please, answer to this question only if you answered yes to question d.27.</td>
<td>The motivation against using ALs includes insufficient perceived ROI (i.e., Return On Investment), and too much formality (of formal AL) or too little support for stakeholder communication. Fewer participants mentioned the absence of an industrial standard.</td>
</tr>
<tr>
<td>d.27.b) Are there any ADL features the lack of which influenced your decision? Please, answer to this question only if you answered yes to question d.27.</td>
<td>As for the specific missing AL features, the participants again mentioned tools support, usability, integrability, and lack of standards.</td>
</tr>
<tr>
<td>d.27.c) Are there any features that you would require in an ADL? Please, answer to this question only if you answered yes to question d.27.</td>
<td>Some additional AL features mentioned as required are traceability and cross-view consistency.</td>
</tr>
<tr>
<td>d.27.d) Why you (or your organization) never considered to use an ADL. Please, answer to this question only if you answered no to question d.27.</td>
<td>For those participants who never even considered using an AL, the general opinion is that ALs are too heavyweight (referring to formal ALs), again in terms of required investments, effort, and formality.</td>
</tr>
</tbody>
</table>
Users of ALs

Fig. 9. Usefulness of AL features in past projects and for future projects, where: Tool = Tool support, Life-cycle = Life-cycle wide support, ItArch = Support for iterative architecting, Ext = Extensibility, Custom = Customization, An = Analysis, Lib = Support for a library of components, patterns, etc., Views = Support for multiple architectural views, Inter = Interoperability with other ALs, Styles = Architectural styles support, Collab = Support for collaborative architecting, UML = UML support, Vers = Versioning, RTE = Code reverse/forward engineering, TextSy = Textual syntax, GrapSy = Graphical syntax, TreeSy = Tree-based syntax, SketchSy = Sketch-based syntax, CodeAl = Support for the alignment of SA description with the implemented system, Sem = Well-defined semantics.
<table>
<thead>
<tr>
<th>AL Features</th>
<th>Average usefulness by ADL users</th>
<th>Average usefulness by non-ADL us.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool support</td>
<td>1.5</td>
<td>1.6</td>
</tr>
<tr>
<td>Life-cycle wide support</td>
<td>1.17</td>
<td>0.83</td>
</tr>
<tr>
<td>Support for iterative architecting</td>
<td>1.6</td>
<td>1.46</td>
</tr>
<tr>
<td>Extensibility</td>
<td>1.4</td>
<td>0.7</td>
</tr>
<tr>
<td>Analysis</td>
<td>1.33</td>
<td>1.08</td>
</tr>
<tr>
<td>Support for a library of components and patterns</td>
<td>0.97</td>
<td>0.6</td>
</tr>
<tr>
<td>Multiple arch. views</td>
<td>0.5</td>
<td>1.74</td>
</tr>
<tr>
<td>Support for collaborative architecting</td>
<td>1.2</td>
<td>1.04</td>
</tr>
<tr>
<td>Versioning</td>
<td>1.33</td>
<td>1.38</td>
</tr>
<tr>
<td>Textual syntax</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Graphical syntax</td>
<td>1.2</td>
<td>1.37</td>
</tr>
<tr>
<td>Support for alignment between SA description and impl. system</td>
<td>1.4</td>
<td>1.04</td>
</tr>
<tr>
<td>Well-defined semantics</td>
<td>1.6</td>
<td>1.42</td>
</tr>
</tbody>
</table>
Desirable AL features

- accessible and easy-to-use language,
- bridge the gap between real world and research, fit what practitioners really need,
- support multiple views (similar to UML but with automated consistency checked between views),
- incremental adoption, i.e., allow the partial definition of the system, without constraining the architect to model it in full,
- no fancy tools, just something that works and is reliable,
- extensible, in both language and tool support.
Everything in one

ALs need to support the dual and complementary roles of software architects

i. to support the level of **formality and precision** required by disciplined development processes, and

ii. to be **simple and intuitive** enough to communicate the right message to stakeholders and to promote collaboration.

---

Everything in one

ALs need to support the dual and complementary roles of software architects

i. to support the level of formality and precision required by disciplined development processes, and

ii. to be simple and intuitive enough to communicate the right message to stakeholders and to promote collaboration.

<table>
<thead>
<tr>
<th>AL Features</th>
<th>Average usefulness by ADL users</th>
<th>Average usefulness by non-ADL us.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool support</td>
<td>1.5</td>
<td>1.6</td>
</tr>
<tr>
<td>Life-cycle wide support</td>
<td>1.17</td>
<td>0.83</td>
</tr>
<tr>
<td>Support for iterative architecting</td>
<td>1.6</td>
<td>1.46</td>
</tr>
<tr>
<td>Extensibility</td>
<td>1.4</td>
<td>0.7</td>
</tr>
<tr>
<td>Analysis</td>
<td>1.33</td>
<td>1.08</td>
</tr>
<tr>
<td>Support for a library of components and patterns</td>
<td>0.97</td>
<td>0.6</td>
</tr>
<tr>
<td>Multiple arch. views</td>
<td>0.5</td>
<td>1.74</td>
</tr>
<tr>
<td>Support for collaborative architecting</td>
<td>1.2</td>
<td>1.04</td>
</tr>
<tr>
<td>Versioning</td>
<td>1.33</td>
<td>1.38</td>
</tr>
<tr>
<td>Textual syntax</td>
<td>1.1</td>
<td>1.1</td>
</tr>
<tr>
<td>Graphical syntax</td>
<td>1.2</td>
<td>1.37</td>
</tr>
<tr>
<td>Support for alignment between SA description and impl system</td>
<td>1.4</td>
<td>1.04</td>
</tr>
<tr>
<td>Well-defined semantics</td>
<td>1.6</td>
<td>1.42</td>
</tr>
</tbody>
</table>
Survey on ALs – summary of findings

1. Architectural languages used in practice mostly originate from industrial development instead of from academic research
   • Implication: Need of understanding industrial requirement

2. ALs should combine features supporting both communication and disciplined development. We call this the introvert versus extrovert nature of architect role.

3. Organizations prefer semi-formal and generic ALs to formal and domain-specific ones
   • Implication: ALs should be simple and intuitive to communicate the right message to stakeholders, while enabling formality so to drive analysis and automatic tasks

4. …
Framework of AL requirements

- Architect as analyst and quality auditor
- Architect as negotiator and communicator

**Language definition**
- A.1 Support to specify extra functional properties
- A.2 Formal semantics
- A.3 Support for graphical and textual specification

**Language features**
- B.1 Multi-view management
- B.2 Extensibility and customization
- B.3 Programming framework

**Tool support**
- C.1 Automated analysis
- C.2 Support for SA-centric design
- C.3 Large view management
- C.4 Support for collaboration
- C.5 Support for versioning
- C.6 Support for knowledge management

Which formalism to use to describe the architecture of a complex system?

- UML / UML profile?
- Architecture Description Language (ADL)?

A set of them, many tools?
Multiple views!
Multiple views!
Multiple views !?!

- Who are the intended consumers of a view?
- What’s the purpose of a view?
- What’s the rationale of a view?
- Why are we using this specific modeling environment?
- How the different views relate each other?
Let’s (re-)start from the Needs

• A structured, repeatable method for
  • sufficiently describing and evaluating the system to be developed
  • communication among the system stakeholders
  • evaluating investments and investment alternatives, based on scenario and impact driven implementation decisions
  • planning and managing the development activities, such as division of labour and integration
  • verifying the compliance of the system’s (or sub-system’s) implementation
Architecture Framework!

"An architecture framework is a coordinated set of viewpoints, conventions, principles and practices for architecture description within a specific domain of application or community of stakeholders."


Architecture Framework

• A set of general elements of an architecture described by use of a concise and consistent terminology
  • Guidance and rules for modelling, documenting, developing, understanding, analysing, using, and comparing architectures based on a common denominator (/ISO42010/) across a (virtual) development organization (i.e. value net).

• The intention of an architecture framework for the automotive industry is to
  • ensure that descriptions of vehicle architectures can be compared and related across different vehicle programs, development units and organizations
  • establish the foundation for overall value creation efficiency, risk reduction and, ultimately, increased innovation
Architecture Framework!
Architecture Description

42010-2011 - ISO/IEC/IEEE
Systems and software engineering --
Architecture description
Conceptual model of an architecture description language

42010-2011 - ISO/IEC/IEEE
Systems and software engineering --
Architecture description
Volvo Cars Architecture framework

- Focus on:
  - System of Systems (SoS) viewpoint
  - Continuous Integration & Deployment (CI&D) viewpoint
  - Ecosystem viewpoint
### Architecture Framework: stakeholders

#### Table 1: Overview of Stakeholders

<table>
<thead>
<tr>
<th>Stakeholder</th>
<th>Group</th>
<th>Comment</th>
<th>Synonyms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passengers</td>
<td>end-user</td>
<td>Purchaser of a car or related service</td>
<td>function owner; function</td>
</tr>
<tr>
<td>Drivers</td>
<td>end-user</td>
<td>Acquirer of electrical system</td>
<td>realizer; function developer;</td>
</tr>
<tr>
<td>Customers</td>
<td>customer</td>
<td>Purchasers of electrical system</td>
<td>function realizor, system</td>
</tr>
<tr>
<td>Product planner</td>
<td>customer</td>
<td>Owners of scheduling responsibility; includes group, department</td>
<td></td>
</tr>
<tr>
<td>Purchaser</td>
<td>customer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line managers</td>
<td>management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project managers</td>
<td>management</td>
<td>Owns budget for development</td>
<td></td>
</tr>
<tr>
<td>System architects</td>
<td>developers of electrical system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functional developers</td>
<td>developers of electrical system</td>
<td>Owns functional and non-functional aspects</td>
<td></td>
</tr>
<tr>
<td>Component developers</td>
<td>developers of electrical system</td>
<td>Can be internal or external from the perspective of the OEM</td>
<td></td>
</tr>
<tr>
<td>SW supplier (internal/external)</td>
<td>developers of electrical system</td>
<td>Can be internal or external from the perspective of the OEM</td>
<td></td>
</tr>
<tr>
<td>HW supplier (internal/external)</td>
<td>developers of electrical system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testers</td>
<td>developers of electrical system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute Owners</td>
<td>developers of electrical system</td>
<td>Owns non-functional attributes like performance</td>
<td></td>
</tr>
<tr>
<td>Tool Engineers</td>
<td>developers of electrical system</td>
<td>Specifically testing tools, including design tools (e.g. for requirements)</td>
<td></td>
</tr>
<tr>
<td>Calibrators</td>
<td>developers of electrical system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diagnostic method engineers</td>
<td>maintainers of electrical system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Workshop Personnel</td>
<td>maintainers of electrical system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Faults Tracing Specialists</td>
<td>maintainers of electrical system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Technical Specialist</td>
<td>specialists</td>
<td>Support developers and maintainers on specific topics</td>
<td></td>
</tr>
</tbody>
</table>
Architecture Framework: scenarios
Car in a SoS: concerns

• Once the car is part of a SoS, how to guarantee functional safety requirements?

• Once functional safety requirements involve devices that are outside of the vehicle (other constituent systems of the SoS), how to ensure that these requirements will be guaranteed?

• How the methods and processes for end-to-end function development and continuous delivery of software need to evolve to be suitable in a systems of systems setting?

• Which functions in the car are allowed to use data coming from other constituents?
Car in a SoS: concerns

- How to enable a **reliable and efficient communication** between the vehicle and heterogeneous entities, like other vehicles, road signals, pedestrians, etc.?

- How to be sure that the vehicle and other constituent systems of the SoS will be **able to exchange information and to use** the information that has been exchanged?

- How to **keep the data** shared within the SoS (and possible replication of data) sufficiently **updated or synchronized**?

- How to manage the **age of available information**?
Car in a SoS: concerns

• How to guarantee that the security of the vehicle is preserved once the vehicle becomes connected?

• How to identify the right tradeoff between shared data and users' privacy?

• Which functions in the car are allowed to make use of data coming from other constituent systems?


CI&D viewpoint: concerns

• How can we avoid building the wrong architecture?
• How can we reduce the number of architectural assumptions?
• How can a system respond quicker to changes in the market?
• How can we deal with changing interfaces?
• How can we deal with dependencies?

C. Yang, P. Liang, P. Avgeriou, U. Eliasson, R. Heldal, P. Pelliccione, T. Bi, Documentation of software architectural assumptions: An industrial case study, Submitted to Journal of systems and Software (JSS)
Ecosystem and transparency: concerns

- Which types of **value-chains** are implied by a given system architecture and what is their purpose?

- How to map **supplier development capabilities** to demands created by a specific system architecture?

- How can we **establish** the required level of **transparency** in a value-chain?

- How can we **manage transparency** (e.g. of architectural decisions) in the face of changing suppliers?
Ongoing work

- Detailing the viewpoints
  - Identifying Model kinds

- Modeling and managing relationships among viewpoints

- Integrating Architecture framework with ATAM (Architecture Tradeoff Analysis Method)
Takeaways

• Defining an architecture for a complex and real system is much more than just modeling
  • Besides technicalities we need to consider also the business, process, and organization dimensions

• In addition to UML there are many other modeling languages and tools for describing an architecture

• Architecture frameworks are one of the way to manage the complexity of the architecture of a complex and big system