Concept/Topic: *Viewpoints*

a.k.a. Viewpoint Models

Context

By definition, software architecture deals with many aspects/facets of a system (as architecture has a "bridge-building", communicating, problem-solving type of role). Hence, architects deal with a large set of diverse concerns; they interact with stakeholders from different communities, including economics/business, development, and operations. The information need, technical background, and communication preferences of these stakeholder communities differ greatly.[1]

It is next to impossible to show an entire architecture in a single diagram, model or specification because of the inherent complexity and the diversity of the various stakeholder concerns. Designing an architecture of a social network, for instance, so that it meets compliance-by-design and privacy-by design requires an end-to-end security perspective; optimizing its deployment so that it meets demanding/advanced performance and reliability requirements under ever growing load requires an operational viewpoint that looks at server nodes and network specifics in detail.

Viewpoints divide a complex problem (here: architecting) into multiple simpler ones (here: architecting for a particular stakeholder concern such as budget and business benefit, security, or systems management).

Definitions

To manage the inherent complexity, slices through and/or projections of the overall end-to-end architecture are taken to tackle and document one stakeholder's viewpoint and perspective at a time. A standardized or widely used *viewpoint model* should be used when creating such *views* (see below for commonly applied ones).

According to Software Systems Architecture[2] by N. Rozanski and E. Woods, one of the books that popularized the concept, viewpoints are defined like this:

```
"A viewpoint is a collection of patterns, templates, and conventions for constructing one type of view.
It defines the stakeholders whose concerns are reflected in the viewpoint and the guidelines,
principles, and template models for constructing its views."
```

Views and viewpoints have an instance-class relationship. Sophisticated viewpoint models use two dimensions to be able to map cross-cutting concerns such as security and performance to primary viewpoints that represent a group of stakeholders.

The classical 4+1 picture is (adapted from: "The 4+1 View Model of architecture"[3], IEEE Software 1995):

[1] Even if some terminology such as "component" or "pattern" seems to be common and shared, one can never be sure that it actually means the same thing everywhere; it is a good idea to ask a lot of "what do you mean by …?" questions and request access to sample exemplars when joining an organization or a project in a role that involved architecture).
[2] https://www.viewpoints-and-perspectives.info/home/viewpoints/
[3] http://ieeexplore.ieee.org/document/469759/

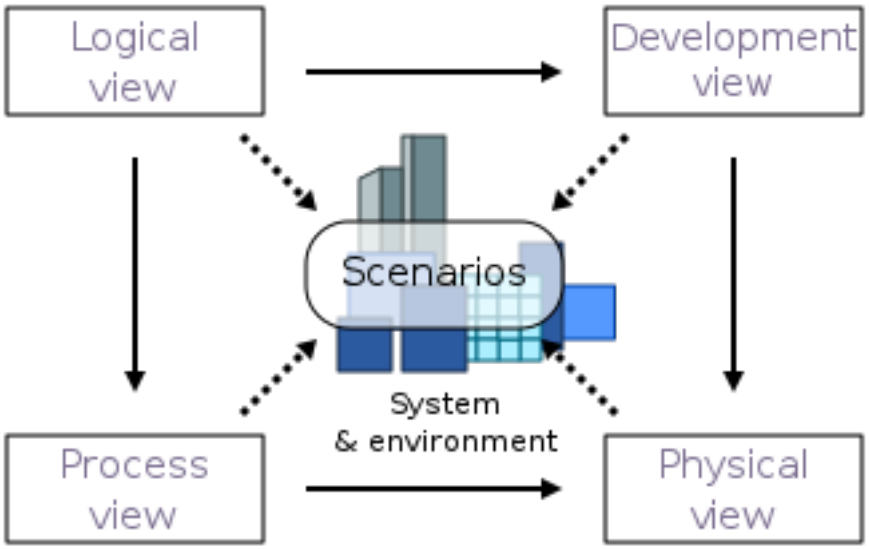

Figure 1: *Illustration of the 4+1 Architectural View Model (reference: https://en.wikipedia.org/wiki/4%2B1_architectural_view_model)*

Application in Products and Projects

Usage of Concept/Topic

The notion of viewpoints has been added to a de-jure standard for software architecture descriptions, the ISO/IEC/IEEE 42010:2011, Systems and software engineering – Architecture description for architectural descriptions[4].[5]

Popular viewpoint models include:

- Philippe Kruchten's 4+1 views on software architecture[6] model, strongly associated with Object-Oriented Analysis and Design (OOAD) and the Rational Unified Process (UP), but also useful without them (see above).
- The two-dimensional *viewpoints* and *perspectives* model introduced in a seminal book by N. Rozanski and E. Woods with the somewhat misleading title "Software Systems Architecture" (see below).
- IBM viewpoint model[7] by P. Spaas et al. A logical-functional and a physical-operational viewpoint dominate and define the architecture artifact landscape in this model (see below).
- arc42 viewpoints: *Context and Scope*, *Building Block View*, *Runtime View*, *Deployment View* (these four views are four out of twelve sections of arc42 architecture descriptions; more on this later).
- Research works on the subject, for instance the proposal to add a decision viewpoint[8].

N. Rozanski and E. Woods describe a rather elaborate two-dimensional viewpoints-and-perspectives model:

- Primary viewpoints[9]:
  - *Context*: What are the relationships, dependencies, and interactions between the system and its environment (the people, systems, and external entities with which it interacts)?
  - *Functional*: What are the main functional elements of your architecture?
  - *Information*: What data will be managed, stored, and presented?

[4] http://www.iso-architecture.org/ieee-1471/
[5] The standard itself is not freely available, but a supporting set of templates is: http://www.iso-architecture.org/ieee-1471/templates/.
[6] https://en.wikipedia.org/wiki/4%2B1_architectural_view_model
[7] https://www.ibm.com/developerworks/rational/library/08/0108_cooks-cripps-spaas/
[8] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.474.8760&rep=rep1&type=pdf
[9] https://www.viewpoints-and-perspectives.info/home/viewpoints/

  - *Concurrency*: How will these elements interact with one another and with the outside world?
  - *Development*: What development, test, support, and training environments will be provided?
  - *Deployment*: What physical hardware and software elements will be required to support these functional and information elements?
  - *Operational*: What operational features and capabilities will be provided?

- Cross-cutting perspectives[10]:
  - *Accessibility*: The ability of the system to be used by people with disabilities.
  - *Availability and Resilience*: The ability of the system to be fully or partly operational as and when required and to effectively handle failures that could affect system availability.
  - *Development Resource*: The ability of the system to be designed, built, deployed, and operated within known constraints around people, budget, time, and materials.
  - *Evolution*: The ability of the system to be flexible in the face of the inevitable change that all systems experience after deployment, balanced against the costs of providing such flexibility.
  - *Internationalization*: The ability of the system to be independent from any particular language, country, or cultural group.
  - *Location*: The ability of the system to overcome problems brought about by the absolute location of its elements and the distances between them.
  - *Performance and Scalability*: The ability of the system to predictably execute within its mandated performance profile and to handle increased processing volumes.
  - *Regulation*: The ability of the system to conform to local and international laws, quasi-legal regulations, company policies, and other rules and standards
  - *Security*: The ability of the system to reliably control, monitor, and audit who can perform what actions on what resources and to detect and recover from failures in security mechanisms.
  - *Usability*: The ease with which people who interact with the system can work effectively.

See this example from IBM for another two-dimensional viewpoint organization scheme:

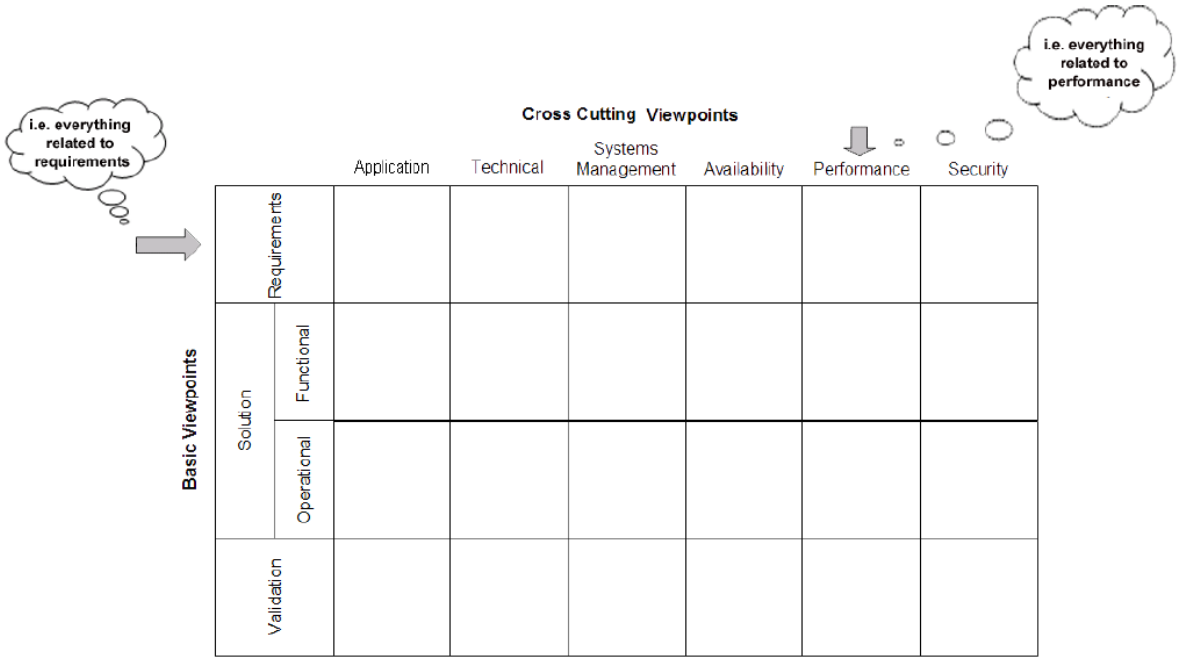

Figure 2: *IBM Viewpoint Model (source: https://www.ibm.com/developerworks/rational/library/08/0108_cooks-cripps-spaas/)*

In the lecture, we will primarily use the classical 4+1 model and blend in elements from S. Brown's C4 model

[10] https://www.viewpoints-and-perspectives.info/home/perspectives/

(which will be introduced later) as well as arc42 (and leave the Rozanski/Woods and/or IBM models for a software architecture master class).

Examples

A concrete example of viewpoint usage, population, and linkage for a fictitious Travel Bookin System (TBS) is provided in this book chapter[11]:

Partially populated Artifact and Model Transformation (AMT) matrix, using IBM viewpoint model

Tips and Tricks

- Choose one viewpoint model per project (or, even better, per organizational unit) and stick to it.
- Only add viewpoints if really needed (for instance, a decision viewpoint).
- Chose one notation per viewpoint, name it, profile it, apply it consistently.
- Discuss both static structure and behavior in each viewpoint.
- Refine the views during the project iterations/sprints; apply source code control/versioning when/while doing so.

Follow-On Topics and Concepts

- Architecture design methods[12] typically embrace or introduce a particular viewpoint model.
- S. Brown's Context, Containers, Components, Classes (C4)[13] model can also be seen as a viewpoint model (although he argues against viewpoint models in some of his presentations)
- Logical component modeling, physical/operational deployment modeling as taught later in AppArch and other modules at HSR FHO (APF, VSS).

More Information

- Via IFS website Architectural Knowledge Hubs[14]

[11] http://www.soadecisions.org/download/AMT-BookChapter-KVZ11.pdf
[12] https://gitlab.dev.ifs.hsr.ch/ZIO/AppArch/tree/V1.0/2-lecture-script/lesson2/ZIO-MethodFactSheet.pdf
[13] https://c4model.com/
[14] https://www.ifs.hsr.ch/index.php?id=13193&L=4