Topic: *(Architecture Design) Method*

Context

Software engineering project work is confronted with many challenges, including:

- Different backgrounds and preferences of project participants
- Repetitive tasks, dealing with many unknowns
- The desire to produce reproducible, predictable results under uncertainty and risk
- Stakeholder concerns such as deliver promised/contacted functionality and quality on time and within budget
- Attrition (dt. "Fluktuation") of project participants, for instance external consultants

Methods response to these challenges. Method designs differ just like software designs; two extremes are waterfall processes and agile practices; most pre-agile methods have been ans still are iterative and incremental in nature, which means that the intermediate results are touched and refined multiple times on a projects.

Method engineering is a field and topic in its own right. P. Kruchten commented about the state of the art and provided some historical insights in this blog post[1].

Definition(s)

We will use the following terms:

- *Method* rather than methodology because the latter term primarily refers to the science of/about methods (Greek suffix -ology for "Lehre").
- General methods for software engineering exist, as well as more specific *architecture design methods*.
- OMG SPEM[2] defines a metamodel for method engineering and clarifies terms such as *role*, *artifact*, and *process* step/activity.

The (traditional) building blocks of software engineering and architecture design methods are shown in the Figure "Method building blocks" (in line with OMG SPEM and an emerging ISO standard).

The agile movement favors practices that can be combined flexibly over monolithic methods and processes (although the subway lines shown here[3] actually *are* methods).

Examples

Open Unified Process (Open UP)[4] is a prominent, rich process model that also provided guidance and concept information on how to create its artifacts.

[1] https://philippe.kruchten.com/2011/03/11/we-do-not-need-richer-software-process-models/
[2] http://www.omg.org/spec/SPEM/
[3] https://www.agilealliance.org/agile101/subway-map-to-agile-practices/
[4] http://epf.eclipse.org/wikis/openup/

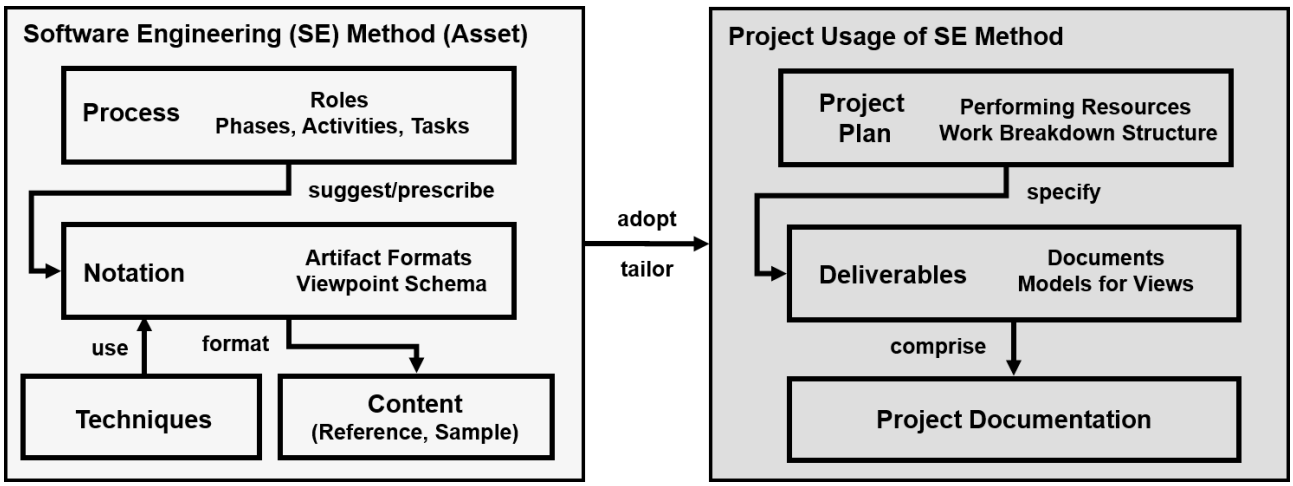

Figure 1: *Method building blocks (reference: O. Zimmermann, 2009)*

Application in Products and Projects

Usage of Concept/Topic

Five architecture design methods often cited in academic publications are Attribute-Driven Design (ADD), Siemens' 4 Views, RUP (Rational/IBM), BAPO/CAFCR (Philips), and ASC/ARES (Nokia); their common elements are established in Hofmeister et al. (2007). OpenUP[5] is the open source version of the Rational Unified Process (RUP). It is an architecture-centric iterative and incremental method.

In the following, some of many currently practiced methods are listed (note that not all of those described in the literature are actually practiced; some are stalled).[6]

General methods (with architectural elements):

- Disciplined Agile Delivery (DAD)[7] and its component and predecessor Agile Modeling[8]
- IBM Unified Method Framework (UMF), e.g., Application Development (AD) 2.0 and IBM Agile with Discipline (AWD)[9]
- Object-Oriented Analysis and Design (OOAD), e.g., in Open Unified Process (Open UP)[10]. A sample technique (a.k.a. method guidance): Focus on the architecture early to minimize risks and organize development[11]

Architecture design methods:

- ADD Version 3.0, see this tutorial[12]
- Risk- and Cost-Driven Architecting[13]
- The patterns presented by S. Toth in "Vorgehensmuster für Software-Architektur", Hanser-Verlag (in German), also form a method (informally, pragmatically).

[5] http://epf.eclipse.org/wikis/openup/
[6] Domain-specific methods also exist, e.g. for SOA design.
[7] http://www.disciplinedagiledelivery.com/
[8] http://www.agilemodeling.com/
[9] https://www.ibm.com/developerworks/community/blogs/c914709e-8097-4537-92ef-8982fc416138/entry/high_level_discussion_of_ibm_s_agile_with_discipline_awd
[10] http://epf.eclipse.org/wikis/openup/
[11] http://epf.eclipse.org/wikis/openup/publish.openup.base/guidances/concepts/core_principle_focus_346C6FAF.html
[12] http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=436536
[13] http://www.sei.cmu.edu/library/assets/presentations/poort-saturn2013.pdf

The collection of practices presented throughout this lecture can also be considered a method (pragmatically specified).

Tips and Tricks

When selecting and applying methods and method elements, be advised to:

- Always customize and reduce to a tailored "minimum viable method" suited for your context and project/product development effort.
- Combine elements from different methods as/if needed.
- Always write for a particular target audience (stakeholders with concerns). Never fill out a template or create an artifact *just* because a method or a method exponent have told you so, but model with a purpose (which may include creative thinking and self reflection, communication with stakeholders, documentation of understanding and design).
- When deciding to include or exclude an artifact, apply the less-is-more principle ("in doubt, leave it out").
- Apply patterns and other reusable assets to reduce risk and cut cost (more on this later in the lecture).

Related Topics and Concepts

- Architectural Significance (Precursor)
- Viewpoints (Sibling)
- Agile Architecting (Sibling)
- SMARTer NFRs (Successor)
- NFR templates (Successor)

More Information

- See ZIO's IFS website¨on method selection and tailoring[14] for resources.

References

Hofmeister, Christine, Philippe Kruchten, Robert L. Nord, J. Henk Obbink, Alexander Ran, and Pierre America. 2007. "A General Model of Software Architecture Design Derived from Five Industrial Approaches." *Journal of Systems and Software* 80 (1): 106–26. doi:10.1016/j.jss.2006.05.024[15].

[14] https://www.ifs.hsr.ch/index.php?id=13193&L=4
[15] https://doi.org/10.1016/j.jss.2006.05.024