



System Architectures, a Structured Approach

System Architecture for Everyone

Software systems are increasing in complexity due to evolutionary concepts such as object-orientation, multi-threaded processing, distributed architectures, etc.

This introduced the need for structure in software design and development. One can find proof in *recent* technologies such as automatic documentation generation tools, modeling languages, refactoring tools, etc. Now, how do we cope with the constant tendency of growing software systems both in size and complexity?

As easy as one, two, three

Software is – until further notice – a product of mankind, and we function better in a structured environment; students structure their syllabi using colors and text-markers; business people can't imagine living without their PIM solutions; think of driving on the highway without road markings.

System architecture provides this urge for structure as a "holy trinity".

First, we see it as a representation of all human-understandable information representing the basic (and detailed) building blocks of the software system. The Architectural, Functional and Technical design documentation provide a clear (and detailed) overview to all participants in the software development process, up and including the customer.

A design should be modular, this way it is possible to add loosely coupled functionality without impacting the existing processes. Therefore, we emphasize the importance of low-level (code artefact = module) as well as high-level (overview = coupling) documentation.

Secondly and perhaps most importantly, it is the process of producing and maintaining this representation. All design/development/maintenance activities will be based on this documentation and extend or update it.

All these activities are realized according to predefined guidelines and templates (discipline) in order to provide structured, streamlined and efficient processes. Reviewing the outcome of the processes described above (design/code reviews) allows executing them in a controlled manner and fine-tuning the guidelines and templates.

Complex and large systems

System architecture should be scalable. No matter how large or complex the system becomes, the cat should always be capable of finding the kittens.

During impact-analysis it is very convenient to have the documentation which can navigate down- and upwards. We create a link between the software artefacts and the related business processes.

Analysis

People designing and developing software are in desperate need of getting a grip on the software system. System architecture provides a collection of best practices and clearly identified processes to cope with the intangibility of their and their colleagues' software. The documentation – as a result of these processes – must be used by every actor of the software system. Users can verify whether their business processes are correctly implemented, whilst IT infrastructure managers how to maintain the system in a *fit* condition.

References:

- <http://www.sei.cmu.edu/opensystems/glossary.html>
- http://en.wikipedia.org/wiki/System_architecture