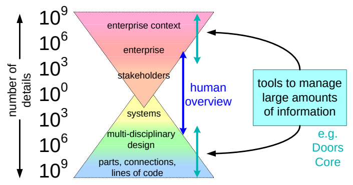
The Tool Box of the System Architect

-



Gerrit Muller
University of South-Eastern Norway-NISE
Hasbergsvei 36 P.O. Box 235, NO-3603 Kongsberg Norway
gaudisite@gmail.com

This paper has been integrated in the book "Systems Architecting: A Business Perspective", http://www.gaudisite.nl/SABP.html, published by CRC Press in 2011.

Abstract

The toolbox of a systems architect is filled with a quite diverse collection of tools. We will discuss the "intellectual" tools, practical low-tech tools, a number of classes of computer assistance tools, and architecting related standards.

version: 0.1 status: draft September 6, 2020

1 Introduction

The subject of tools for systems architecting creates numerous debates. We will use a broad interpretation of the word tool, including intellectual tools, low-tech tools (such as pen and paper), but we will also discuss computer assisted tools. One of the key questions is when to apply what tool. An essential capability for systems architects is to pick an appropriate tool, and, if needed, to adapt it to the situation at hand.

We discussed in Sections **??** and **??** that the role of the systems architect depends on the organizational context. Similarly, there are organizations that force a set of tools on systems architects based on the perceived role and way of working of system architects.

We base our discussion of tools on the deliverables, responsibilities and activities as described in Sections **??** and **??** (Figures **??** and **??**). Key contribution of systems architects in these sections is the simplification of complicated systems into understandable essentials. Main challenges in achieving this contribution are the heterogeneity of the system and its context, and the uncertainties and unknowns in the system and its context. The goal is to make systems specification and design decisions communicable, and to facilitate debate and reasoning about decisions.

Many organizations move in practice too fast to extensive use of computer assisted tools. As consequence the architects and stakeholders move away from overview and understanding essentials to more detailed concerns (that also have be to be addressed!). The purpose of this section is to help understand the impact of tool selection, and especially to bring balance in the application of *intellectual* tools versus *computer assisted* tools.

2 Overview of Systems Architecting Tools

Figure 1 shows an overview and a classification of systems architecting tools. The left side shows the tools that are independent of computers and related software programs. The right hand side shows tools that depend on computers and specific software. The bottom part of the figure shows some of the standards that impact the selection and application of tools.

2.1 Human Experience Based tools

Experience is crucial for systems architects. Systems architects meet new approaches during their entire career, and they build a rich frame of reference by seeing many systems in many circumstances. Reflection on the way of working transforms events into valuable experience: approaches are transformed into *methods* and *techniques*, and problems and solutions are transformed into *patterns*.

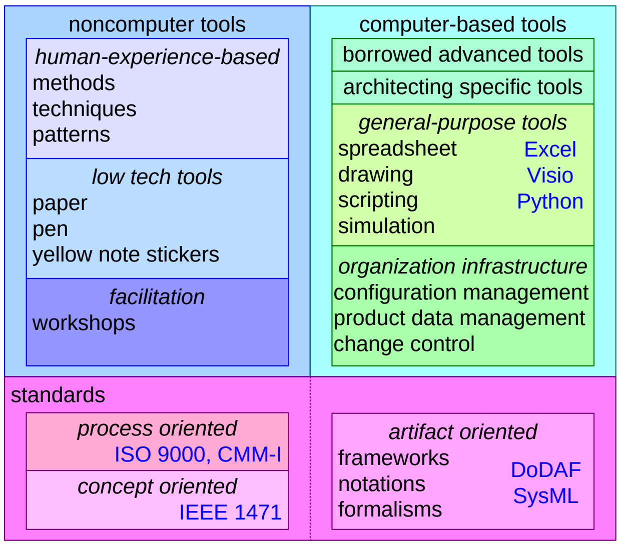


Figure 1: Classification of Architecting Tools

Methods describe an approach in terms of objective, the order or logic of steps to follow, the techniques that can be applied, and the models, tools, notations, and formalisms that can be supportive.

Techniques are ways to address a specific aspect of a problem. For example, how to analyze timing requirements and problems. Techniques can be supported by specific tools and formalisms. A technique may require specific models. For instance, the analysis of response time may require functional flow models.

Patterns are recognized problem-solution combinations, including the considerations in what context and circumstances a solution is appropriate for the problem. Patterns can be highly technical, e.g. the publish-subscribe pattern in software to solve flexibility and extendibility needs. However, patterns can also be high level organizational or business, such as considerations about products versus services in Figure **??**.

2.2 Low-tech tools

Systems architects, like building architects, often make sketches. The sketches are on napkins, paper, flip-charts, white-boards, using pens, pencils et cetera. Sketching is a fast way to express and exchange ideas, and as such has to be values as essential part of the systems architect toolbox. Note that similarly other low tech means, such as folded paper, wire frames, and yellow note stickers provide fast and intuitive ways to express and exchange ideas.

It might be a challenge to capture these sketches for further communication, later re-use, and archiving. However, with today's ubiquitous digital cameras this is easily captured. Later in the process these sketches get captured electronically in more structured form, e.g. in Visio.

2.3 Facilitation tools

System architects can contribute to teams by applying facilitation techniques, as described in Section **??**. An example is the organization of work shops, where teams can explore and share ideas effectively. There are many more facilitation tools and techniques, such as:

- the use of flip charts to create a common memory on the wall
- the use of balanced feedback, e.g. soliciting benefits and concerns
- working in teams and plenary groups
- preparing meetings together with the leader
- round robin or random order contributions to get input from less dominant team members

2.4 Borrowed Advanced Tools

Systems architects cooperate with a large amount of experts. Every expert has its own set of tools. Sometimes the architect borrows such tool and adapt it to be used as system level. For example, mechanical engineers are used to tolerance budgets. Systems architects use budgets for many different system qualities (e.g. response time), where granularity of the budget and the algorithms behind the budget have to be adapted to the quality at hand. Most tools in systems architecting find their origin somewhere in another discipline.

2.5 Architecting Specific Tools

The problems to be addressed by a tool and the solutions for these problems need to be well-defined and repeatable Before a computer assisted tool can be made. The nature of many systems architecting problem is often quite opposite, with characteristics such as heterogeneous, uncertainties and unknowns. The systems architecting effort is mostly spend in understanding the problem. Solving well understood problems in a repeatable and predictable way is the domain of engineering.

Most *systems* specific tools are more engineering related (nailing down all detailed information to facilitate the ordering, production, sales, and support of the system) than architecting related. Examples are tools to capture requirements (e.g. Doors), functional and physical architectures such as IDEF0 (e.g. Core), or object oriented architectures in for instance SysML.

2.6 General Purpose computer based tools

Architects and engineers use computers all the time for many different purposes. Architects will use a lot of general purpose tools, such as spreadsheets (e.g. Excel), drawing programs (e.g. Visio), scripting (e.g. Python), or simulation (e.g. Python, MATLAB or many others).

The general purpose nature of these tools makes them attractive for architects, since that helps them to cope with heterogeneity, unknowns and uncertainties. The class of more advanced tools can be too restrictive to allow adaptation to the problem at hand and its circumstances.

2.7 Tools prescribed by the organization infrastructure

Organizations do have an engineering tool infrastructure that systems architects can not ignore. However, systems architects have to decide when and how to interface to the organizational infrastructure. Examples of typical organizational infrastructures are many data bases and repositories for engineering related information:

Configuration management describing the parts and the rules how the parts can be configured. This repository can be part of a larger system such as an Enterprise Resource Planning (ERP) system (typically SAP).

Product Data Management (PDM) storing all product and part related information required for the Customer Oriented Process.

Change Control and Problem Report data bases, where all Change Requests, Internal Problem Reports, and Field Problem reports are stored.

System architects sometimes have to work for some time outside these systems, because these systems tend to slow down more creative work full of unknowns and uncertainties. The challenge for project leaders and systems architects is to migrate to these systems at the right moment: using these systems too early slows dona too much, starting to use them too late might cause loss of information and quality problems.

2.8 Process Oriented Standards

There are many process oriented standards that influence the way of working of systems architects. For example the maturity models in CMM-I more or less prescribe most of the tools (Configuration management, change control) discussed in the previous paragraphs.

Process oriented standards tend to be agnostic for specific tools. In general these standards try to capture best practices from the past in an attempt to preventing past mistakes. Systems architects in practice suffer when these processes are implemented to the letter rather than the intent. An unintended side effect can be that systems architects are transformed into administrators, while their main contribution is in content rather than administration.

2.9 Concept oriented Standards

Some standards try to capture the shared understanding of the architecting discipline. A good example is the IEEE 1471 standard, where the concepts *stakeholders*, *concerns*, *architecture description*, and *viewpoints* are captured. These standards do no prescribe a way of working but provide a set of concepts and their relations to ease communication.

2.10 Artifact Oriented Standards

In the defense world several frameworks have been created defining the artifacts that can describe an architecture. Typical examples are DoDAF and MoDAF of respectively the USA Department of Defense and the UK Ministry of Defense. These frameworks do not define the process, but rather limit themselves to defining the artifacts that may describe the architecture. These standards tend to see the artifacts as electronics artifacts with a significant degree of formalization to facilitate computer assistance.

Part of the Systems Engineering community has transformed UML from the software engineering world into a more systems oriented modeling language SysML. SysML is a set of formalisms to create artifacts that can be used for computer assisted tools.

3 Human versus Computer Assisted Tools

One of the main challenges is to decide when and for what to use computer assisted tools, as stated in the introduction. Figure 2 shows a so-called four quadrant analysis of intellectual (human) tools and computer assisted tools. The four quadrants are obtained by adding a second dimension: strength and weakness.

Strengths of humans , based on their intellect, are:

- to be able to focus on overview
- to be able to identify the essentials
- to understand relationships

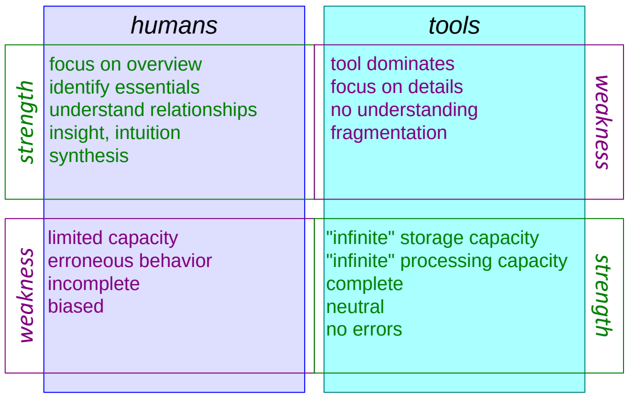


Figure 2: 4 Quadrant analysis of computerized and human tools

- to have insight and intuition
- to be able to synthesize (to combine heterogeneous information into a meaningful picture)

Strengths of computers , based on current technological level, are:

- near-infinite storage capacity
- near-infinite processing capacity
- the ability to be *complete* by storing *all* information
- to be neutral, without emotions, opinions, or (political) interests
- to be perfect in execution, making *no errors*

Weaknesses of humans , inherent to their social and psychological background, are

- storage and processing capacity is limited.
- showing behavior that is erroneous
- memory is imperfect, information is often *incomplete*
- biased, for emotional, social or political reasons

Weaknesses of computers , inherent to their mechanistic technical nature, are:

- the *tool dominates*, because there is no “reasonable” flexibility
- the information is in full detail, moving the *focus on details*

- computers do not have any *understanding* (garbage in, garbage out)
- the data tends to be *fragmented*, only stored relations are present.

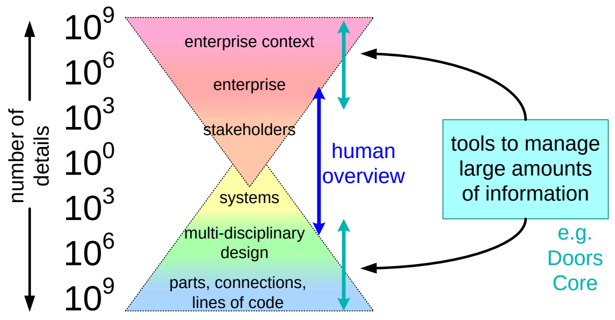


Figure 3: Tools Support Processing of Large Amounts of Details

The idea behind the four quadrants is that the weaknesses of humans can be compensated by the strengths of computers and vice versa. If we map these characteristics on the pyramids of Figure **??** then we see that human intellect is required at the higher abstraction levels where we strive for understanding between heterogeneous stakeholders. Computer assisted tools bring most of their value where large amounts of data have to be managed and processed. Most computer assisted tools address a limited set of concerns, such that the problem is well defined and the solutions can be applied repeatable and predictable. Many computer assisted tools are mono-discipline oriented, since disciplines capture repeatable knowledge.

4 Flow: from Data to Overview and Understanding

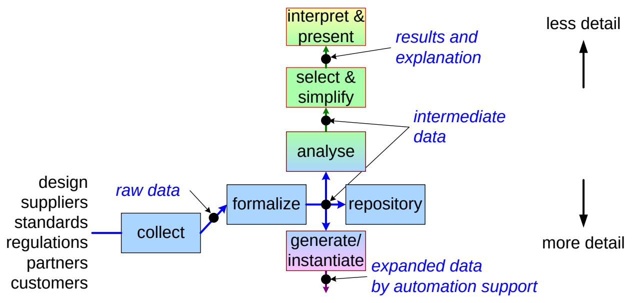


Figure 4: From Data to Understandable Information

We have seen in the previous Subsection that computer based tools create most of their value when large amounts of data have to be managed and processed. Other discussions that pop up when computer assistance is used is the degree of formalization and the use of automated outputs. Figure 4 shows the flow from input data up to the moment that the results are being used by a heterogeneous group of stakeholders. The figure shows the following functions:

Collect data from many inputs, e.g. the design, suppliers, standards, regulations, partners, and customers. The output of this function is a collection of *raw* data: data that still has to be processed to make it useful.

Formalize to be able to enter the data into computer based tools. The nature of the formalization is to look for appropriate abstractions to capture this data. The consequence of the abstraction is that the amount of detail can decrease slightly, for instance because repeated data is captured more structurally.

Repositories are used to store the formalized data so that this data can be used for many different purposes. For example an information model can be stored as entity relation ship model plus a data dictionary to capture all formatting details. This information model data can be used to generate data structures and code, it can be used to generate test cases for compliance testing, and the data can be used for analysis.

Generation and Instantiation can be applied on prescriptive data in the repository to generate or instantiate components, stubs or test harnesses.

Analysis techniques are applied on the data to determine characteristics of the design. For example, the form, shape and material characteristics of components can be used to calculate the center of gravity of components and of the aggregate of multiple components. Another example is that configurations can be analyzed for feasibility and performance.

Selection and Simplification is a function that is applied by humans (architects or designers) to make the results ripe for communication and discussion. The output of automated analysis techniques is often rather detailed and highly formal, while the essential aspects are hidden in a huge amount of other details.

Interpretation and Presentation are the last steps in making the information accessible and understandable for the broader group of stakeholders. In interpretation the meaning of the the outputs is added: is a center of gravity deviation of 10mm a problem or is it quite good? The presentation is the format of the output, what visualization will engage the stakeholders, how to ensure that the information relates to the mental model of the diverse stakeholders?

A common mistake made by engineers is that they show their own intermediate data to stakeholders that use a different mental framework themselves. The consequence is that the communication is quite incomplete and the risk is significant that stakeholders will disconnect or will not give any reaction even when necessary.

Systems architects have to make the last steps of selection, simplification, interpretation and presentation. Note also that these steps bring their own risks: every simplification is only valid within its limits, so architects are also responsible to monitor the validity of discussions and decisions in light of the used simplifications.

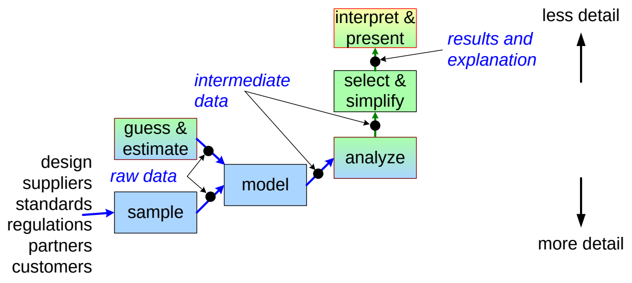


Figure 5: Data Flow Early in Creation Process

Early in the development projects architects are using a slightly simplified flow to facilitate system specification and design, as shown in Figure 5. This figure shows that early in the process many estimates and guesses are used, and that less formalization is used. Remember that formalization and computer based tools are especially relevant when large amounts of data have to be processed and managed. More simple models can be used by architects as long as the amount of information is small.

Figure 6 maps the data flow on the pyramid with the abstraction levels. This mapping shows again the relation between the amount of information and the kind of tools to be used: repositories, generator tools and analysis tools are typically computer assisted, while the intellectual challenges of selection, simplification, interpretation, and presentation are human activities.

Figure 7 summarizes these areas of application in the pyramid. The bottom parts of the pyramid with large amount of details can be characterized as more formal and requiring more rigor. Formalization requires well defined problems, data, and operations that are repeatable. The data is machine readable to allow automated tools. The use of repositories facilitates re-use over systems and components.

The upper part of the pyramid is characterized by the combination of quite heterogeneous data with uncertainties and unknowns used ba heterogeneous group of stakeholders with variable backgrounds and concerns. This upper part is less

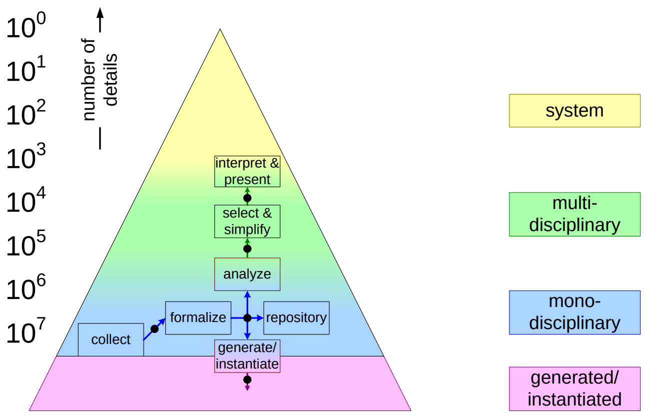


Figure 6: Data Flow Mapped on Pyramid

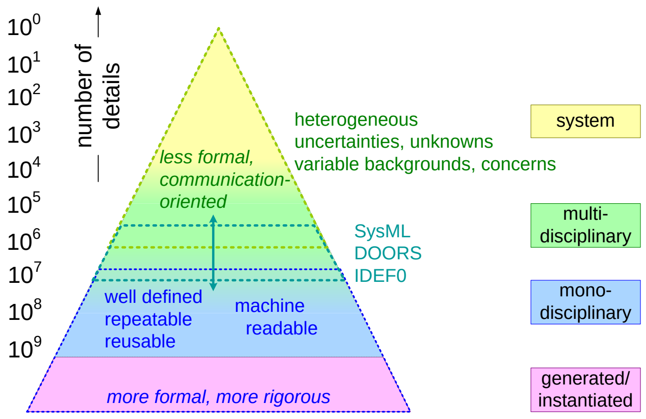


Figure 7: Formality Levels in Pyramid

formal and oriented towards communication, discussion and decision making.

References

[1] Gerrit Muller. The system architecture homepage. `http://www.gaudisite.nl/index.html`, 1999.

History

Version: 0.1, date: July 26, 2010 changed by: Gerrit Muller

- logo: KDAWStoolsDiabolo
- no further changelog maintained yet

Version: 0, date: July 23, 2010 changed by: Gerrit Muller

- Created, no changelog yet