Contract-Orientented System Design & Specification

Author: Wolfgang Mueller
Address: Cadlab
Bahnhofstr. 32
4790 Paderborn - Germany
Phone: (+)49 52 51 / 284 141
Fax: (+)49 52 51 / 284 140
E-mail: wolfgang@cadlab.cadlab.de

This paper is mainly based on the work done in the ESPRIT project ECIP2, Ref. #2072, on high-level system specification [LEB91, MUE90]. The goal within this work was first to enlighten the main streams and activities during the very early phases of contract-oriented high-level system design and specification. Second, based on the first evaluation, the applicability of the IEEE standard hardware description language VHDL should be studied. In this paper I will only concentrate on the first topic and on applicable standards.

1 High-Level System Design & Specification

System engineering should include all life cycle activities, like specification, design, implementation, and support of systems, where systems may include hardware as well as software components. It is important not to limit systems engineering to the technical domain, such as physical, mechanical, and electrical engineering but to consider the logistic, and the contractual view as well. Contractual view means to consider all contractual constraints, such as limits on budget as well as the flow of deliverables and reports through the entire system development.
High-Level design & specification, which is also often referred to as *requirement engineering*, covers the very first phases of systems engineering. The major input is the customer requirement specification. The major output is first a specification (Customer External Specification) delivered to the external customer indicating all adequate design alternatives. Second, the already executable engineering specification of the design alternative finally chosen is passed to the further design synthesis to be implemented and manufactured.

In the following I would like to sketch the main stream and the main ideas of the contract-oriented high-level system design & specification process such as it was defined within ECIP2 Workpackage 2 (Specification of Systems)[MUE90].

This process may be partitioned into two major phases:

1. Phase: Informal Review/Analysis
A rough design can be developed out of the customer requirement specification appearing in the first contract, applying rules of thumb. This approximate design should be 'post-reviewed' and checked against the specification. The major goal is to detect and to eliminate most of the obvious electrical, physical, mechanical, and financial contradictions at a very early phase. This is, to check the electrical constraints against the physical, the mechanical, and the financial constraints and vice versa. This very first phase is often not explicitly recognised but it is more or less deeply applied in nearly every project.

2. Phase: Analysis & Dimensioning
The first step of decomposition has to deliver a formalised specification at a very high, abstract level. This is, to synthesise the customer requirements to get a first approximate executable specification that may be simulated and/or formally proofed. This specification is the input for the analysis procedures to check the consistency of the specification. Thereafter the dimensioning of the specification can take place. The specification has to be completed, functional parameters have to be adjusted. Thereafter, an approximate optimisation can take place, such as optimising the system's layout. Only design properties that have influence on some contractual considerations should be optimised at this level. Figure 1 reflects this second phase of this contract-oriented system specification process.

Though the entire system specification & design process is highly iterative and all subprocesses are highly concurrent interrelated, a main flow within this process can be identified. This flow only indicates in which order the subprocesses are initialised. After their initialisation all subprocesses run concurrently performing a permanent feedback with those processes formerly initialised. For matters of simplicity all these feedback loops are omitted in Figure 1.

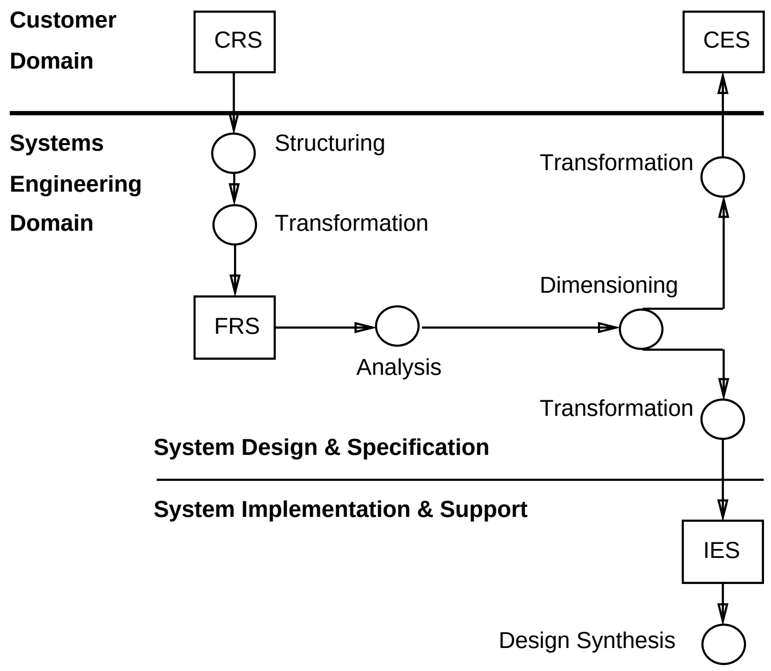

Figure 1: Contract-oriented system design & specification

In Figure 1 subprocesses are denoted by ◯ and key milestones to deliver documents and reports are denoted by □.

At least two different customer/supplier domains can be identified within this process: customer domain and the systems engineering domain including the system specification, design, implementation, and support.
Four major contractual documentation milestones can be fixed:

- Customer Requirement Specification (CRS)
- Formalised Requirement Specification (FRS)
- (Supplier to) Customer External Specification (CES)
- Internal Engineering Specification (IES)

The CRS is the input to the *Analysis & Dimensioning* process. The CRS is transformed into an FRS that can be analysed and dimensioned. Analysis means to check the FRS for completeness and consistency. Dimensioning covers soft-, hard-, and firmware partitioning as well as the performance analysis of the selected parts. After several design iterations the CES and the IES will be submitted. The CES is delivered to the external customer to be acknowledged in order to fix the enhancements and improvements of the customer requirements and to give the customer a detailed view of the product that will be produced. The IES is a supplier internal specification that is used as input for the next major design cycle the *Design Synthesis* followed by the *Manufacturing.*

For this contract-oriented view is very important to be supported by a systems engineering environment to make this discipline widely acceptable for wide industrial application. To integrate all this subdisciplines into a whole will reduce costs and improve the schedule of a product life cycle and the quality of products. These key ideas are also covered by the CALS (Computer Aided Aquisition and Logistic Support) Initiative that was founded by the U.S. DoD. In the CALS scenario also the term *Concurrent Engineering* became popular. In this context concurrent engineering means to specify and to design reliability and maintainability (R&M), supportability, manufacturability, producebility, testability, safety, and quality assurance into the products rather than to install them afterwards.

2 Standards in System Design & Specification

At present the contract-oriented system development is not very much covered by presently available tools. There exist only some few standards within this system development domain. Each company applies its own, mostly informal, in-house methodology. There seems to be a strong need for standards in tool environments and recommended practices for more formal methodologies in systems engineering.

Three main fields for required standardisation may be identified by the following classification:

1. Methodologies
2. Document Representation
3. Tool Environments

Considering the first one, there exist some methods, such as SA, SADT, JSD, etc., that not only define languages but also introduce methods as well. All these methods concern only the technical view of system design. A system specification and design process within a contract has to produce internal documents as well as external documents to make the design decisions visible to the external customer. At present no standard or recommended practices beside the CALS initiative seems to be on the way to define such methodologies and required deliverables.

Considering document representation, CALS has introduced the following standards for electronic file exchange: SMGL (text file exchange), CGM (2-D vector graphics), IGES (3-D vector graphics), GRP 4 Raster (raster scanned images), and MIL-STD-1804 (composition of files into documents) [CALS89]. But considering standards for document representation does not only mean to concern the exchange format of files. This also means standardising the representation and the semantics of diagrams. A lot of different representation are presently used for this purpose. Only two examples should be mentioned here: Statemate © (i-Logix), Grapes © (Siemens Nixdorf). There seems to be a need for an open standard for graphical diagrams in system design, since at present each tool uses its own graphical symbols. Changing the tool implies the need to learn a new language. Changing the language means in most cases to change in addition the design method which might be very time consuming and result in additional costs.

Standardisation of tool environments is mainly covered by the framework standardisation within the CFI activities. Practical systems engineering means to apply a huge set of management and engineering tools. To overcome all problems in this tool communication, tools have to be tightly integrated into one common environment. To minimise data exchange most tools should work on the same common data within one unique database. Thus, minimising data exchange means a full white-box integration of all the tools within one environment. It is very important to automise the design management to guide the user through the design flow. Since the engineer would be overwhelmed by the output of all the design and analysis tools, the decision-making and analysis process should also be automised in most parts by the use of a decision support system (DSS). The DSS suggests plausible strategies and supports the decision between different design alternatives by comparing alternative designs limited by different constraints. Constraints may be described by spreadsheet-like equations, inequalities, and relationships building constraint networks linking participating parameters. Constraints may include manufacturing rules or financial and physical laws. Two different kinds of constraints can be identified: *Quantitative Constraints* and *Qualitative Constraints.* Quantitative Constraints define inequalities over attributes. Qualitative Constraints describe relationships among sets of alternatives.

3 Conclusion

Concluding the above remarks it should be stated that systems engineering should not only concentrate on the technical engineering but has to be extented towards contractual aspects. Different standards are needed to formalise the system engineering process especially at the specification & design level. CAX frameworks are needed to accelerate the development of products and to manage the handling of multiple analysis and design tools. All in all for the establishment of a fully automised integrated systems engineering there is still a long way to go.

References

[CALS89] CALS, An Overview of the CALS Standards, CALS Policy Office, DASD(S)CALS, The Pentagon, Room 2B322, Washington, DC 20301-8000, April 1989.

[LEB91] J. Lebrun, Specification of System Requirement Specification, ECIP/SCTF/910092, ESPRIT Deliverable, January 1991.

[MUE90] W. Mueller, Data Flow in High-Level Design and Specification, ECIP.D2.1.D24, ESPRIT Deliverable, December 1990.