Initial Experience in Contracts Based Systems Engineering

Erik Herzog, Henric Andersson[*]
{erik.herzog, henric.andersson}@saabgroup.com
SAAB Aerosystems
Sweden

Abstract. Engineering process efficiency is the target of any systems engineering organisation. In this paper we review initial experience from applying the Contracts based systems engineering method developed by the SPEEDS project. The method is outlined along with experience gathered from pilot projects performed at SAAB Aerosystems. Outlines for adapting the method to mainstream tools as well as future adoption activities are also presented.

Introduction

Divide and conquer is the established approach for mastering complexity in development of systems. Of course, other terms are preferred, e.g., systems engineering, but the basic premise remains the same: We seek to divide intangible problems in order localise smaller tangible ones that enable future integration and realisation of a working solution. Multiple strategies are applied, e.g., partitioning according to functional or technology domains.

Formulation of proper requirements and architectural design is key factors for successful implementation, and literature on these topics is abundant. Multiple development processes have been proposed and accepted by industry, e.g., waterfall, spiral, or the iterative process. Over the last half century they have also been tuned to properly address the challenges of embedding software in systems. Space is not sufficient here to capture software's role in system development and the associated challenges, see (Doyle and Pennotti 2008), (Wenzel and Bornemann 2006), and (Sheard 2004) for an overview. Concurrent Engineering practises, i.e., the initiation of subsystem development prior to completion of system level development, has also be applied to systems engineering process and is now an accepted part of the systems engineering toolbox. Multiple papers on concurrent systems engineering has been published at INCOSE events, e.g., (Rhodes and Smith 1992) and (Wade 1995).

The traditional approach in Systems engineering is to refine requirements such that they can be allocated to a single system component or component interface and pass down the responsibility for further development to the component level engineering team. With this approach, feedback in the form of integration and verification results to system level engineers is often late and far from satisfactory. One reason for this is, of course, the complexity of component interfaces and also the fact that the knowledge of the integrated system is limited at the time when requirements are allocated. Integration occurs usually late in the process. Early, virtual (in model based tools), integration activities are seldom performed.

[*] Also at the Division of Machine Design, Department of Management and Engineering, Linköpings Universitet, Sweden

We believe that one contributing factor to problems in integration is the prevailing practise in industry to focus exclusively on textual requirement statements. Systems engineers have been taught to refine and distribute requirements among the components, interfaces and behaviour elements that shall fulfil the needs while no or little regard is placed on the assumptions made on the components and interfaces in the design process.

The importance of documenting assumptions lies in the fact that for any complex system the sum of requirement statements on a component will not be exhaustive – in industrial reality the requirements stated on a component will only specify a subset of the complete functionality, behaviour or other properties of a component. At first such statements may appear absurd – intuitively completeness is a matter of adding requirement statements until the characteristics of a product are "completely" defined. But in reality the lack of system knowledge at the time of requirement formulation makes attempts to achieve completeness impractical – even counter productive as many of the characteristics captured when striving towards completeness are bound to be either self-evident or be unnecessary constraints for the designing engineers. Additionally, overly detailed requirements may add unreasonable costs in implementation and verification.

Requirements engineering literature is adamant on the importance on requirements validation, i.e., ensuring that requirements are well formed, sound and affordable, but we have yet to see literature placing equal importance on assumptions validation – with the notable exception of the ARP-4754 standard (SAE 1996). This focus on requirements, at the cost of assumptions, is also evident in our standard design documentation: In our templates we have sections for capturing requirements of different kinds, but there is at most a single section for capturing the assumptions made. Likewise, in SysML we have a dedicated requirement artefact, but no corresponding one for assumptions (although there is a rationale item defined in the language).

In the rest of this paper we outline the approach for adapting the design by contract method popularised in the Eiffel language to systems engineering and to SysML as performed within the EU funded SPEEDS (SPEEDS 2008) project [†]. Initial experience from applying the SPEEDS methodology on a fictitious transportation system is presented along with experience gained on SAAB internal projects.

Design by Contract

The term "design by contract" was introduced with the Eiffel (ISO 25436 2006) programming language and associated development method. The underlying idea is to create a development context similar to that of well formed business agreements. For each component formal assertions on software block pre-conditions and post-conditions are developed prior to or in parallel with development of the source code.

Any pre-condition captured is an *assumption* on the data provided to the block, i.e., the block itself does not exercise any control over data provided upon the entry of the block. Similarly, a post-condition is a *promise* in that it guarantees properties of the data upon completion of execution of the block. *Contracts* are formed by the combination of promises on a code block with assumptions on the following code block. The contract *holds* if the *assumptions made for a block are compatible with promises* made by interfacing code blocks. Contracts in the Eiffel

[†] SPEEDS is an acronym for **SPEculative and Exploratory Design in Systems Engineering.** The project is funded under European Union 6th Framework Project in Embedded Systems Development (2006-2009) IP Contract No 033471. The support is gratefully acknowledged.

language provides the means for efficient code inspection, for on-line monitoring of software execution, for conformance with contracts, and for formal analysis.

Consequently, the Eiffel language provides valuable tools to engineer reliable software. It must however be noted that despite its technical merits and it being an ISO standard it is our perception that Eiffel is not widely used in industrial software development projects.

SPEEDS

SPEEDS is a pan European project with the objective to define a new generation of *end-to-end methodologies, processes and supporting tools for embedded system design.* The SPEEDS project aims at significant improvements of the competitiveness of industry in the sector of embedded systems by means of enhancing model-based systems engineering with

- semantics-based modelling to
    - support the design of complex embedded systems using heterogeneous sub-system models, and
    - enable sound integration of existing and new modelling and analysis tools.
- novel formal analysis tools and techniques that will allow exploring architectural implementation alternatives and “first-time-right” design using the “design-by-contract” paradigm.

The problems addressed by SPEEDS are:

- How to reach a system design solution in a multi-dimensional, concurrent, and multi-disciplinary development environment.
- How to meet multi-dimensional constraints (e.g. safety, reliability, maintainability, resource usage, cost and time).
- How to overcome concurrent, multi-disciplinary, and cross-organizational environment and ensure robust and flexible design, including manufacturers/suppliers cooperation.
- How to provide cost-efficient mapping of applications and product variants onto embedded platforms.
- How to manage risk in the design caused by missed/unstable requirements and design uncertainties.

In this paper we focus on the aspects of SPEEDS associated with the design-by-contract paradigm. The theoretical aspects of SPEEDS are reported in (Engel et al, 2008, Josko et al, 2008). Focus herein is on the end-user oriented aspects of the method. This said, some brief overview of the underlying technology is called for to put the end user experience into context.

Contracts in Systems Engineering

The SPEEDS methodology is based on a model based engineering approach, in particular component based modelling. An information model, HRC (Heterogeneous Rich Component), has been developed providing a semantically well founded base for the method. The HRC model is extensive and defined to support representation of component models originating from languages and tools such as SysML, Matlab/Simulink and SCADE. The semantics is defined such that any of these tools could be used as frond-end modelling editor for creating HRC compliant models. The core notion in HRC is that of the *component*. HRC supports the definition of contracts captured between components in a manner similar to that of the Eiffel method. A component in HRC may have any number of interfaces – each conveying a functional or non-functional property of the component. Interfaces may be flow- or service oriented similar to the ports in SysML and they carry information with an associated type. In this aspect HRC is very much similar to any block oriented specification method, e.g., SysML Internal Block Diagram or Simulink block models. Any number of assertions (*assumptions* or *promises)* may be associated with an interface, where:

- An *assumption* is a statement on the expected properties (physical or logical properties, signals) fed to the interface through the environment, i.e., properties that are not controlled by the component. In flow oriented design environments such as SysML Internal Block Diagrams, *assumptions* are captured on input ports.
- A *promise* is a statement of the properties guaranteed for a component interface – given that assumptions made on other component interfaces are fulfilled by the environment. In flow oriented design environments such as SysML Internal Block Diagrams *promises* are captured on output ports.

When looked upon in isolation, an HRC component and associated assumptions and promises can be defined as presented in

Figure 1. The example is also available in Andersson (2009).

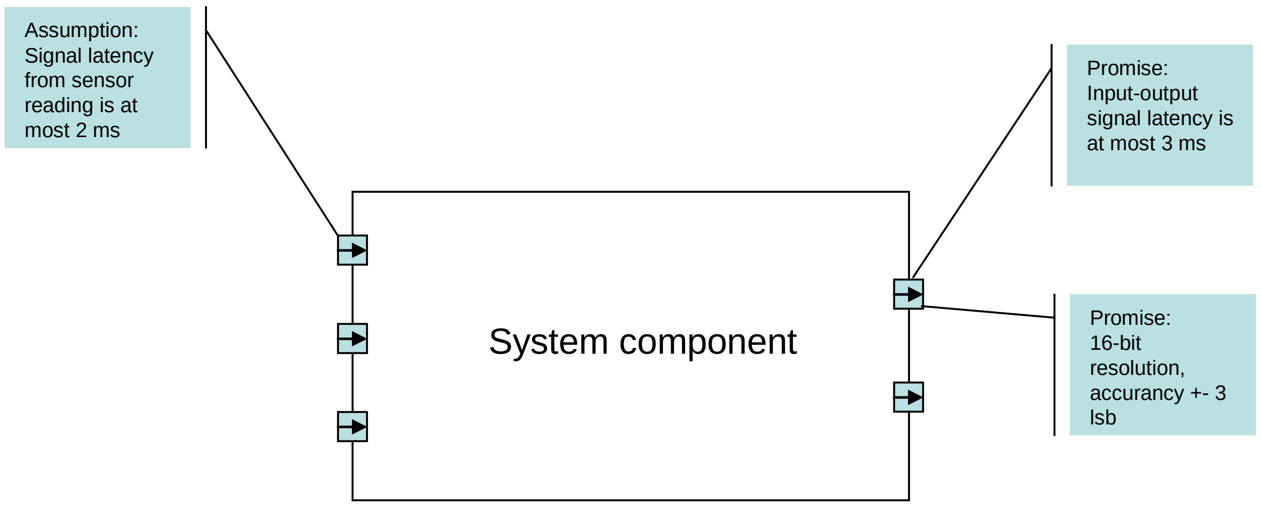
Assumption: Signal latency from sensor reading is at most 2 ms
Promise: Input-output signal latency is at most 3 ms
System component
Promise: 16-bit resolution, accurancy +- 3 lsb

Figure 1: Component model with Assumption - Promises

In

Figure 1 interfaces are presented as flow interfaces, however interfaces may be created for any system property that shall be set in the engineering process, e.g., component weight, size or cost.

Note: In this paper we present assertions in plain English. There is also a formal language, CSL (Contract Specification Language), developed by the SPEEDS project which allow for formal definition of assertions, which in turn enables formal analysis of contracts. This aspect of SPEEDS is discussed in (Engel et al, 2008), but has not yet been applied by SAAB and consequently not further touched upon in this paper.

Contracts: Combining Assumptions and promises

***Assumptions* and *promises* from different components are linked to form *contracts* indicating that the assumed characteristics of an input interface of a component are intended to be satisfied by the output of the component providing the information.**

Figure 2 illustrates *contracts* captured between interfaces of two components K_1 and K_2. The semantics is such that for *Contract* C_1 the assumption A_{21} must be satisfied by P_{11}. As stated above, contract satisfaction could be established through formal analysis (provided that K_1 and K_2 are unambiguously specified) or through manual review.

The use of the term Contract to denote a modelling concept deep within a modelling methodology is not trouble free. Our intuitive interpretation of the term *contract* is that it is a business level agreement which is far from the ‘micro’ style contracts introduced herein. Still we believe the term contract is appropriate as it is our objective to create a clear view of obligations between development teams. Appropriate or not it takes a short extra effort to communicate the semantics of the word *contract*.

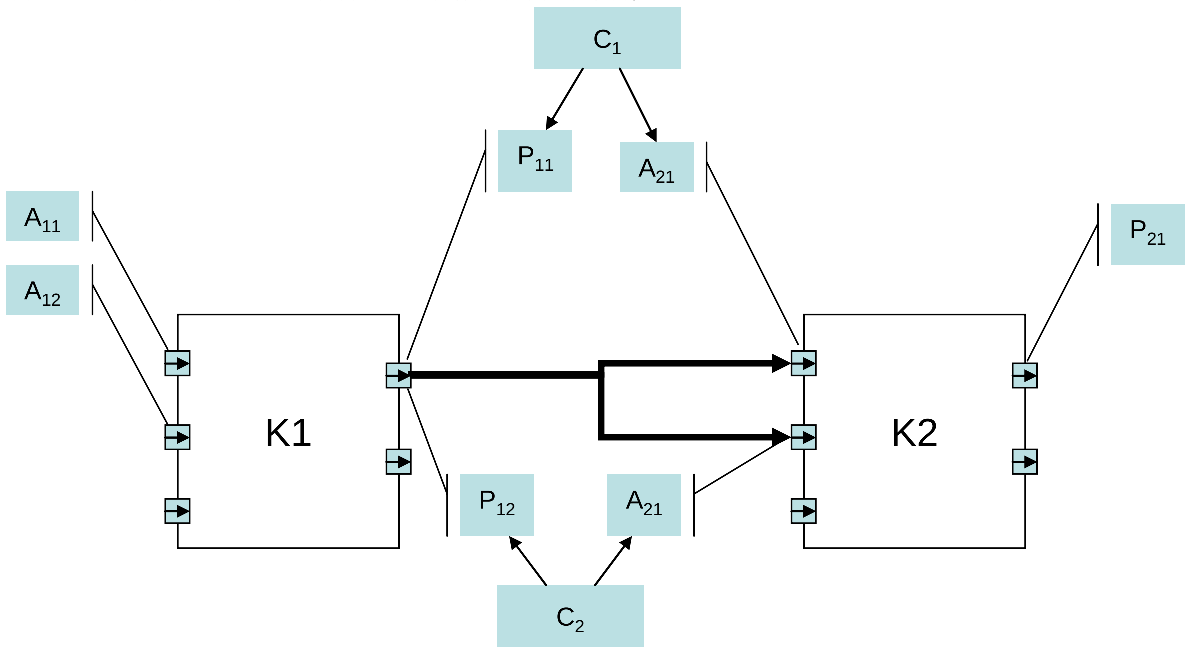

Figure 2: Assumptions and promises combined to contracts

Contracts always group assumptions and promises and may be formulated between interfaces in:

- components at the same level of decomposition within a model
- components in a parent – child relationship
- components capturing the system at different abstraction levels

SysML as an HRC front-end

So far we have presented the HRC format and its capabilities in isolation. There is, on purpose, a formal mapping between SysML and HRC as outlined in

Figure 3. In SysML terms the following diagrams are supported by HRC:

- Internal Block Diagram
- Block Definition Diagram
- Statechart diagram

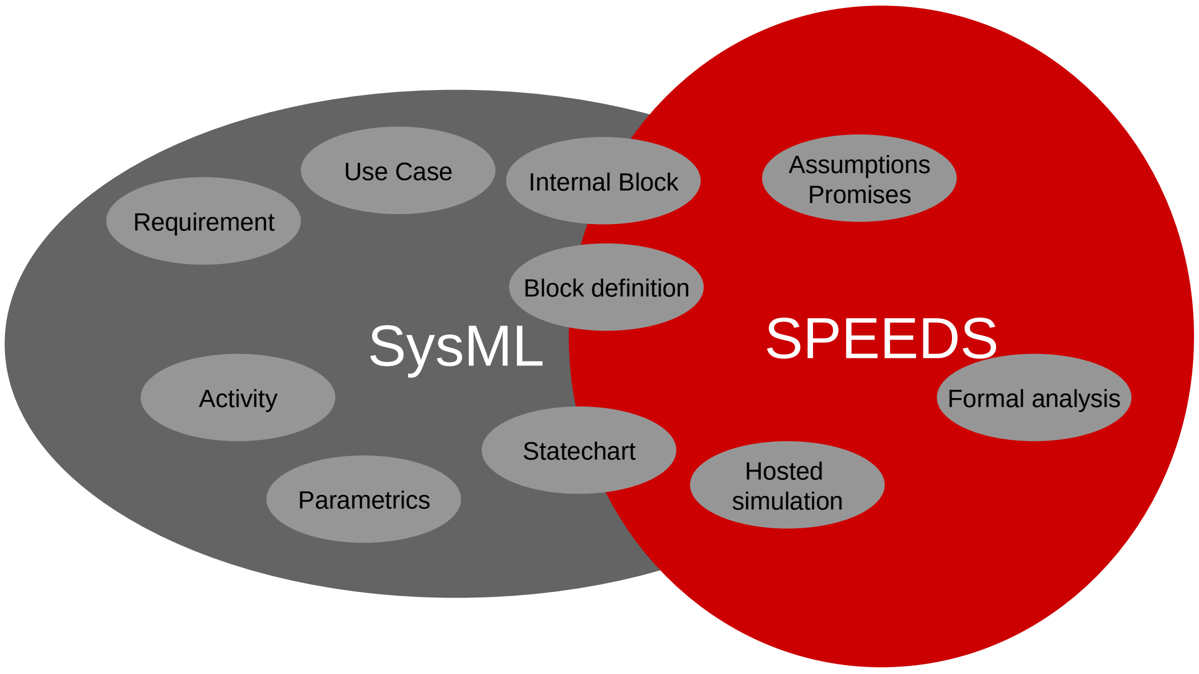

Figure 3: Relationship between SysML and SPEEDS

The partial overlap with SysML allows engineers to perform design work up to the definition of contracts in SysML syntax in a standard SysML tool. For the contract specification and management parts of a system model, it is transformed to HRC format and assumptions/promises are added through a dedicated A/P editor (Assumption/Promise editor) developed within the SPEEDS project.

Perceived advantages

The primary driver for SAAB's interest in the assumption/promise paradigm is in inclusion of *assumptions* in the toolbox provided to our engineering teams. We consider formal analysis promising, but in the general case not yet mature enough for large scale industrial application.

A *promise* does coincide with requirements as used in traditional development methods, in the sense that it expresses function, performance or any non-functional property of a component. Assumptions on the other hand allow engineers to state expected properties or behaviours of the environment in a persistent format. This is in itself highly valuable as information about the design assumptions, in our experience, are seldom captured and maintained. In addition, the advantage is that concurrent subsystem development can be initiated and coordinated before system requirements are stabilised. As contracts are formed by combining assumptions and promises it is natural to review their compatibility and consistency as part of system reviews and hence a natural element in the design process. Additionally the assertions allow design teams to independently specify properties that are assumed or guaranteed allowing for raising the visibility of a property at any time.

This paradigm is also expected to clarify to the engineers the interaction between the design process and the configuration management process. Change proposals and CCB (Configuration Control Board) interaction on system level is called for upon changes in component interface, assumptions, or promises.

The Utopar experience

A fictitious transportation system, Utopar, was used to provide early validation of the SPEEDS methodology. Utopar is outlined in (Engel et al 2008) and consists of a set of autonomous taxis (U-cars) serving a future metropolis. Within the validation activity of the contract based approach, SAAB chose to focus on the Propulsion and Brake System (PBS) within the U-car. Our interest in this system is due to its close connection to physical products/parts and their properties. The special objective with SAAB's validation work is to evaluate the applicability of the SPEEDS methodology to this kind of systems as we see a potential for applying the methodology in our systems engineering process. In particular, SAAB has been interested in capturing the allocation of system properties such as weight, cost and reliability information from system to subsystem level, and capturing timing and performance parameters. For modelling, SAAB has used the Telelogic Rhapsody tool.

System overview

The Utopar propulsion and brake system (PBS) is the component in the U-car that provides the means for controlling the speed of the U-car, i.e., the means for ensuring a safe travel function for Utopar residents in terms of well balanced acceleration and retardation. The environment of the PBS is presented in Figure 4. The PBS interfaces with a sense and avoid system for collision avoidance and a Communication and Navigation system for receiving commands and transmission of status messages. Note that the product structure component is not available in the model, i.e., the U-car component is not visible in

Figure 4.

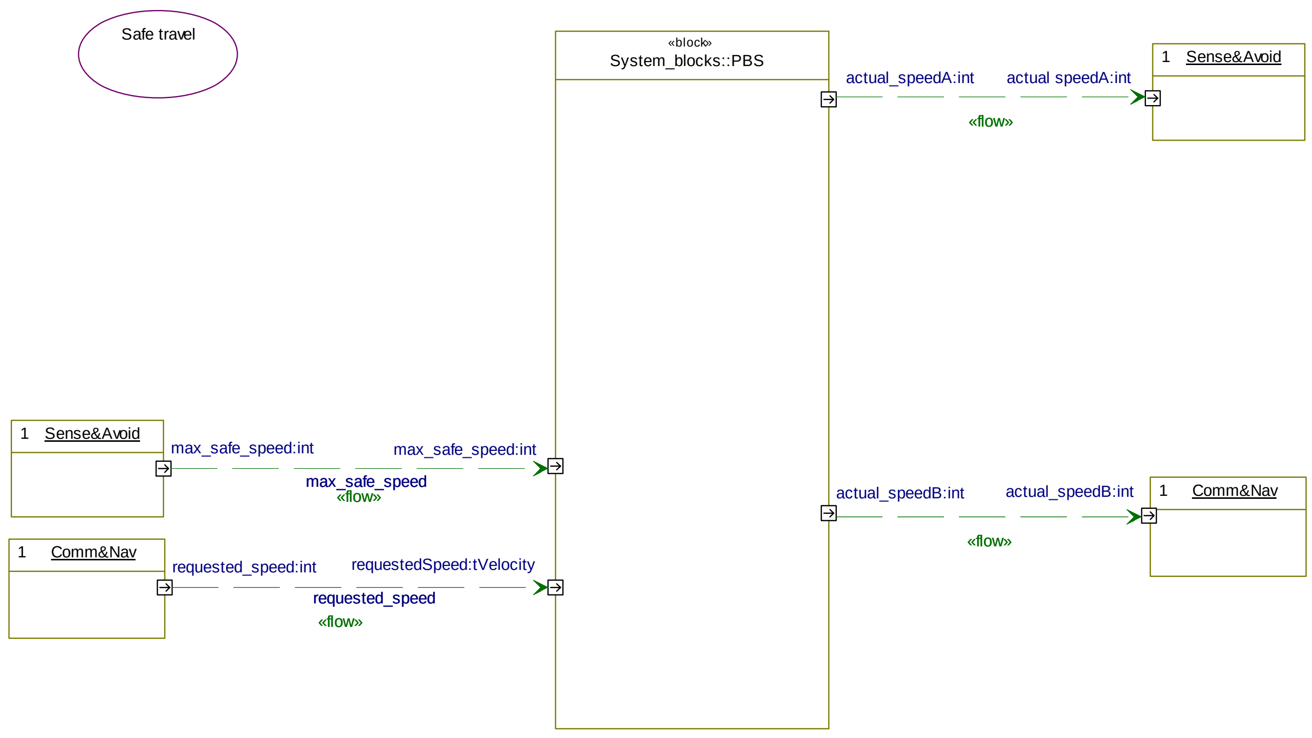

Figure 4: The PBS system environment

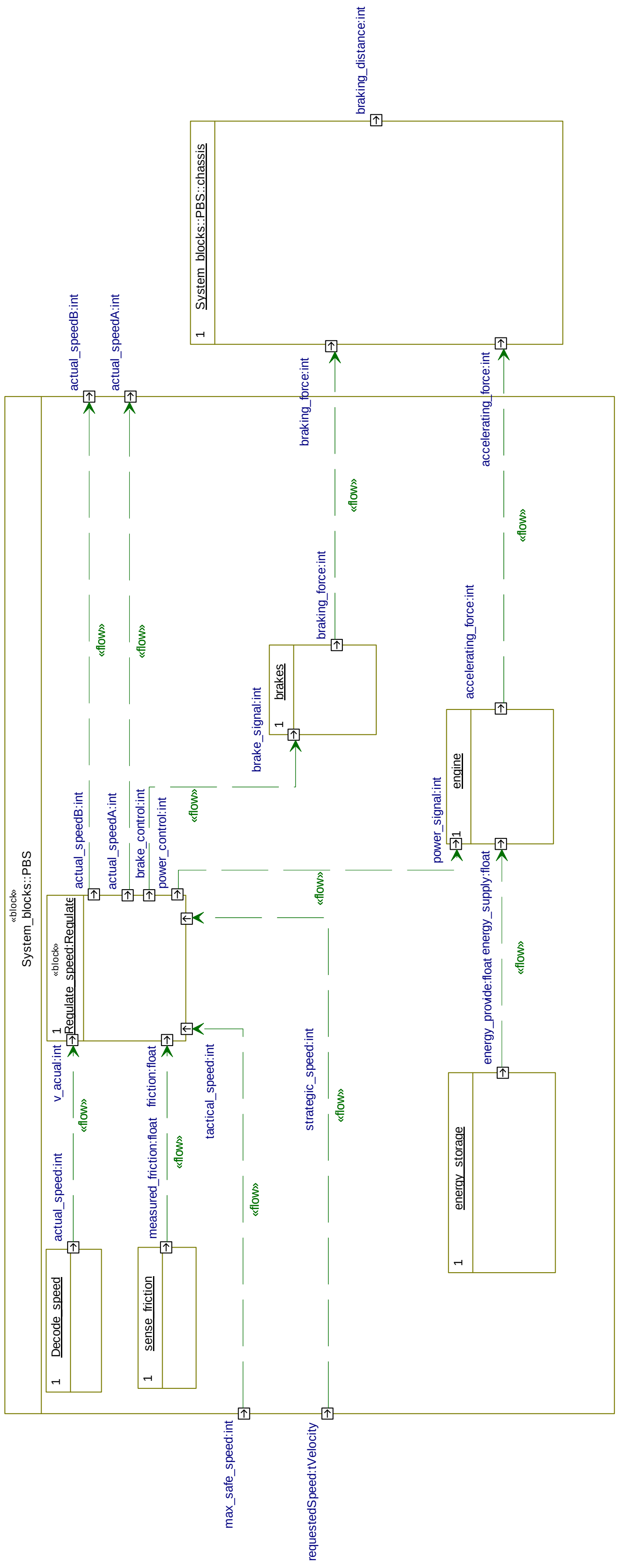

Figure 5: PBS internal structure

Internally the following PBS components were identified as illustrated in Figure 5:

- Speed sensor
- Friction sensor
- Regulator
- Brakes
- Engine
- Energy storage

In addition, to keep the connection to the U-car, the chassis of the car is identified as a component external to PBS.

The following examples of assumptions and promises were captured internal of the PBS and with external components:

- Static propositions of component cost and weight with relationships to U-car performance. The case is created to produce a multi-variable trade space exploration example where objective functions are used to determine the optimal component configuration.
- Temporal separation of messages
- Constraints on rate of data exchange

A sample of the assumptions and promises captured on PBS system level (corresponding to

Figure 4) are presented below

ID	Signal	Unit	Time-domain	Direction	Assertion
PBS-4	max_safe_speed	Float/ m/s	discrete	In	Sense and avoid data is less than 0.2 seconds old (as compared to observation made by ideal sensor) when available at input.
PBS-5	max_safe_speed	Float/ m/s	discrete	In	Latency between update of signal by sense & avoid and reading is less than 0.05s.

Further SAAB work

Based on the positive early experience in Utopar, SAAB is currently initiating a method validation study where the methodology will be applied to the integration of a podded subsystem onto the Gripen aircraft. Integration considers physical properties, physical flows (power and air cooling) and digital communication in the system.

Discussion

What is striking about the initial work performed is the amount of communication between subsystem development teams that have been initiated by assumptions stated. Such assumptions have:

- highlighted design problems that may not have been considered at all, or alternatively discovered late in the design process
- served as drivers for harmonising subsystem design solutions towards optimum. We have noticed that the existence of both assumptions and promises on components creates a climate where subsystem developers can more easily communicate the rationale for assumptions made and also non agreeable consequences of the assumptions not being met, or promises not being relaxed.

Assumptions form a good mechanism for initiating interaction among subsystem engineers. Likewise we have seen its value in communication between engineers at different system levels for allocation of system properties, e.g., weight. Here we let the parent component capture an assumption on the child system property and form the contract with the promise made by the subsystem for the property. For example, the parent component may assume that the weight of a child component is 5 kg or less, while the child component may promise a weight of 5.4 kg. In this case the contract would obviously be violated forcing a renegotiation between engineers responsible for the systems.

As we are focusing on contracts in natural language, the early verification is performed through contract inspection. Such verification is based entirely on model content, i.e., the assumptions and promises in individual contracts. Of course, normal verification on the realised system will be based on the requirements captured, i.e., the promises.

Capturing functional properties using SysML as a front end to the formal HRC is really straightforward. A flow- or service port in SysML corresponds to an interface in HRC. But, HRC requires all properties to be involved in a contract to be expressed using a port (in HRC format). In the current experimental implementation between SysML and HRC the only mapping between HRC interfaces and ports are through SysML ports. This forces the creation of ports for capturing non-functional aspects. For instance, ports must be created for allowing assertions on component weight. This is of course not a natural way of modelling. It would be desirable if future HRC implementations in SysML would allow capturing contracts on non functional properties directly.

In the validation study we have used a project proprietary tool for capturing assertions and forming contracts – the A/P editor. For future projects we plan to use our standard requirements management tool – DOORS for this task. Extending DOORS to manage contracts is a matter of creating modules for assumptions, promises and contracts. Standard DOORS links would be used to link assumptions and promises to form contracts. DOORS also provides the means for capturing additional attributes on each assertion, e.g. stability for assertions or verification status for contracts. The DOORS – Rhapsody interface is already in existence off the shelf, so maintaining consistency between model and assertions is not expected to be problematic.

Conclusions

In this paper we have presented initial experience from application of the Contracts Based Systems Engineering method developed by the SPEEDS project. For industrial application we believe this to be a substantial step forward in specification practise in concurrent engineering projects; the method provides a dedicated specification element for capture of design related assumptions and the mechanism to couple these with requirements - the promises. Initial application of the method indicate that assumptions improve communication between engineers in a natural way, and it provides means for all parties involved in an interface to express constraints on the interaction. Our next step in deploying the method will be through additional and larger pilot projects – potentially also through a cross-organisational project.

References

Andersson H. 2009. Aircraft Systems Modeling – Model Based Systems Engineering in Avionics Design and Aircraft Simulation, Linköping Studies in Science and Technology, Licenciate Thesis No. 1394.

Doyle and Pennotti 2006. Doyle, Laurence; Pennotti, Michael Impact of Embedded Software Technology on Systems Engineering, In Proceedings INCOSE 06, 2006.

Engel Avner, Winokur Michael, Enzman Marc, Döhmen Gert. 2008. Assumptions / Promises Shifting Paradigm in Systems Engineering, In Proceedings of INCOSE 08, INCOSE 2008.

ISO/IEC 25436:2006 - Information technology -- Eiffel: Analysis, Design and Programming Language, 2006

Josko Bernard, Ma Qin, Metzner Alexander. 2008. Designing Embedded System Component Using Hetergeneous Rich Components, In Proceedings of INCOSE 08, INCOSE 2008.

SAE ARP 4754. 1996. *Certification Considerations for Highly-Integrated or Complex Aircraft Systems,* Society of Automitve Engineers, 1996.

Rhodes Donna and Smith Carey. 1992. Practical Application of Concurrent Engineering for Systems Integration. In Proceedings of the second annual international symposium of the National Council on Systems Engineering. NCOSE, 1992.

Sheard, Sarah. 2004. Adapting Systems Engineering for Software-Intensive Systems. In Proceedings INCOSE 04, 2004.

SPEEDS, 2008, The SPEEDS project website: http://www.speeds.eu.com, accessed 2008-11-24.

Wade James. 1995. Systems Engineering Process Standardization: A Road Map for Concurrent Engineering. In Proceedings of the fifth annual international symposium of the National Council on Systems Engineering. NCOSE. 1995.

Wenzel, Stefan; Bornemann, Falk. 2006. Managing Compatibility Throughout the Product Life Cycle of Embedded Systems: Definition and Application of an Effective Process to Control Compatibility. In Proceedings INCOSE 06, 2006.

Biography

Dr. Erik Herzog is a Systems Engineering specialist at Saab Aerosystems AB. Dr. Herzog received his Ph.D. at the Department of Computer and Information Sciences at Linköping University, Sweden. His professional interests include development and introduction of Systems engineering processes, specification methods, information modelling and tool integration techniques. Dr. Herzog is active in many INCOSE working groups and has implemented the INCOSE i-pub system.

Henric Andersson is a control engineer at Saab Aerosystems in Linköping. He works in the Systems Engineering Division responsible for recommending methods and tools for use on Saab aerospace development programmes. The key interest is modeling techniques, requirements management and tool integration with special focus on systems design and simulation. He is also an industry PhD student in model based systems engineering at Linköping University.