Project Number: 033471

SPEEDS
Speculative and Exploratory Design
in Systems Engineering

Integrated Project
Information Society Technologies

SPEEDS Methodology – a white paper

Start date of project: May 1st 2006          Duration: 42 months
Airbus Deutschland GmbH                    Revision: V.1.2

---

<table>
<thead>
<tr>
<th>Dissemination Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PU</td>
<td>Public</td>
</tr>
<tr>
<td>PP</td>
<td>Restricted to other programme participants (including the Commission Services)</td>
</tr>
<tr>
<td>RE</td>
<td>Restricted to a group specified by the consortium (including the Commission Services)</td>
</tr>
<tr>
<td>CO</td>
<td>Confidential, only for members of the consortium (including the Commission Services)</td>
</tr>
</tbody>
</table>
## Approvals

<table>
<thead>
<tr>
<th>Family Name</th>
<th>Name</th>
<th>Company</th>
<th>Date</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPEEDS Coordinator</td>
<td>Döhmen</td>
<td>Gert</td>
<td>AIG</td>
<td></td>
</tr>
<tr>
<td>Quality Assurance</td>
<td>Döhmen</td>
<td>Gert</td>
<td>AIG</td>
<td></td>
</tr>
<tr>
<td>Configuration       Management</td>
<td>Döhmen</td>
<td>Gert</td>
<td>AIG</td>
<td></td>
</tr>
</tbody>
</table>
## Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Modification</th>
<th>Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>v.0.1</td>
<td>Dec. 12th 2007</td>
<td>Initial version</td>
<td>Marc Enzmann</td>
</tr>
<tr>
<td>v.0.2</td>
<td>March 6th 08</td>
<td>First draft</td>
<td>Gert Döhmen</td>
</tr>
<tr>
<td>v.1.0</td>
<td>April 10th 08</td>
<td>Final Version</td>
<td>Gert Döhmen</td>
</tr>
<tr>
<td>v.1.1</td>
<td>April 15th 08</td>
<td>Modifications of chapter 4.1</td>
<td>Henric Andersson</td>
</tr>
<tr>
<td>v.1.2</td>
<td>April 24th 08</td>
<td>Added chapter 5.3</td>
<td>Christian Härdt</td>
</tr>
</tbody>
</table>
# LIST OF CONTENTS

1 **MANAGEMENT SUMMARY** ........................................................................................................... 6

2 **INTRODUCTION** ........................................................................................................................................... 7

   2.1 Motivation .................................................................................................................................................... 7

   2.2 Audience and Scope of the White Paper ..................................................................................................... 7

   2.3 Brief description of the paper ..................................................................................................................... 7

3 **CONTEXT AND OBJECTIVES** .................................................................................................................. 8

   3.1 SPEEDS Vision ......................................................................................................................................... 8

4 **SPEEDS APPROACH AND SCIENTIFIC BASE** ......................................................................................... 8

   4.1 State of the Practice (of embedded systems design methods) ....................................................................... 8

      4.1.1 Introduction ...................................................................................................................................... 8

      4.1.2 System Architecture ........................................................................................................................... 8

      4.1.3 Requirements Analysis and Simulation ............................................................................................... 9

      4.1.4 Rapid Prototyping and Production Code generation ........................................................................... 9

      4.1.5 Formal analysis .................................................................................................................................. 9

      4.1.6 Tool integration .................................................................................................................................. 10

      4.1.7 Configuration management ................................................................................................................ 10

      4.1.8 Summary .......................................................................................................................................... 10

   4.2 SPEEDS Concepts and scientific foundation ............................................................................................... 10

      4.2.1 Component-based Design using HRC (Heterogeneous Rich Components) ...................................... 10

      4.2.2 Component-based Development ......................................................................................................... 12

   4.3 Contract-based Analysis/Design Methodology ............................................................................................. 15

      4.3.1 Design steps ..................................................................................................................................... 16

5 **SPEEDS TOOL ARCHITECTURE** .................................................................................................................. 17

   5.1 Integration principles ................................................................................................................................. 17

   5.2 SPEEDS Bus ........................................................................................................................................... 19

   5.3 Process Advisor Toolset ........................................................................................................................... 21

6 **SPEEDS DELIVERABLES** ............................................................................................................................ 22

   6.1 Overview ................................................................................................................................................... 22

   6.2 SPEEDS Process ..................................................................................................................................... 22

   6.3 SPEEDS Methods .................................................................................................................................. 23

      6.3.1 HRC modelling methodology ............................................................................................................. 23

      6.3.2 ADT analysis methodology .................................................................................................................. 24

   6.4 SPEEDS Tools ....................................................................................................................................... 24

      6.4.1 Modelling Tools ................................................................................................................................. 24

7 **BENEFITS OF SPEEDS** ............................................................................................................................. 25

   7.1 Tool Users ................................................................................................................................................ 25

   7.2 Tool providers ....................................................................................................................................... 25

   7.3 Research ............................................................................................................................................... 25

   7.4 Final product user ................................................................................................................................. 25

8 **SPEEDS INTELLECTUAL PROPERTY HANDLING** ...................................................................................... 26

9 **EXPLOITATION** ......................................................................................................................................... 26
LIST OF FIGURES

FIGURE 1 HETEROGENEOUS RICH COMPONENTS ................................................................. 11
FIGURE 2 SYSTEM DESIGN SCENARIO ............................................................................... 12
FIGURE 3 VIRTUAL INTEGRATION BASED ON CONTRACTS .................................................. 13
FIGURE 4 VALIDATING AN IMPLEMENTATION ..................................................................... 14
FIGURE 5 CHECK USABILITY OF LIBRARY ELEMENTS ......................................................... 14
FIGURE 6 HORIZONTAL AND VERTICAL INTERFACES/CONTRACTS OF A COMPONENT ........ 15
FIGURE 7 ABSTRACT REPRESENTATION OF THE DESIGN METHODOLOGY FRAMEWORK .. 15
FIGURE 8 .................................................................................................................................. 16
FIGURE 9 COOPERATION BY FILE EXCHANGE ................................................................. 18
FIGURE 10 CENTRALISED SERVICE ARCHITECTURE .......................................................... 18
FIGURE 11 SPEEDS - OPEN SERVICE NETWORK ............................................................... 19
FIGURE 12 SPEEDS BUS SUPPORTING THE SPEEDS ENVIRONMENT ............................. 20
FIGURE 13 EXAMPLE SERVICE INTERACTION DIAGRAM ............................................... 20
FIGURE 14 TOOL CLASSIFICATION ..................................................................................... 21
FIGURE 15: ASSESSMENT, PROCESS AND ADVICE ......................................................... 22
1 Management Summary

SPEEDS is a concerted effort to define the new generation of end-to-end methodologies, processes and supporting tools for embedded system design. They will enable European systems industry to evolve from model-based design of hardware/software systems, towards integrated component based construction of complete virtual system models.

The SPEEDS project aims at significant improvements of the competitiveness of the European 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 tools.
- Novel formal analysis tools and techniques that will allow to explore architectural implementation alternatives and “first-time-right” design using the “design-by-contract” paradigm.
- A new tools-supported design process, the “Controlled Speculative Design”, that will minimize the risks of concurrent design activities, by providing a trustworthy development environment through the definition of formal contracts between design groups.

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 overcome multi-dimensional constraints (e.g. safety, reliability, maintainability, resource usage, cost, time, etc.).
- 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 design caused by missing or unstable requirements and design uncertainty.

The SPEEDS consortium contains partners from potential end-users (OEM and system-suppliers) in the aerospace and automotive industries, tool-vendors, and renowned European research institutes:

- Airbus Deutschland GmbH
- Airbus France
- Israel Aerospace Industries Ltd. (formerly Israel Aircraft Industrie Ltd.)
- Daimler-Chrysler AG (till 31.12.2006)
- Carmeq (from 01.05.2007)
- Robert Bosch GmbH
- Magna Powertrain Engineering Center
- Knorr-Bremse Fekrendszer Kft
- SAAB AB
- EADS (from 01.05.2007)
- Telelogic Israel Ltd. (formerly I-Logix)
- Esterel Technologies SA
- Extessy
- GEENSY (formerly TNI)
- Université Joseph Fourier - Verimag
- PARADES GEIE
- Kuratorium OFFIS E.V.
- INRIA

The project-coordinator for SPEEDS works at Airbus Germany and can be contacted as follows:

Gert Döhmen  
Head of Methods and Processes  
EDYND – Information Systems Architecture & Integration  
Airbus Deutschland GmbH  
Kreetslag 10  
21129 Hamburg  

Tel.: +49 (0)40 743-83740  
E-Mail: gert.doehmen@airbus.com
2 Introduction

2.1 Motivation

The embedded systems sector is today one of the pillars of European industrial development. In recent years improvements have been made in the productivity and efficiency of their development as a result of constantly improved design methodologies and their supporting tools for model based design. However two issues are apparent, the need for increased industrial efficiency continues to be at the heart of international competitiveness and the landscape of model based design and its supporting tools is mostly one of independent islands. Tools proliferate and improve selected sections of the process but lack connectivity and a common sound semantic foundation that enforces, in practice, a glass ceiling to the productivity gains that can be obtained.

The SPEEDS project is essentially a concerted effort to define the new generation of end-to-end methodologies, processes and supporting tools for embedded system design. They will enable European systems industry to evolve from model-based design of hardware/software systems, towards integrated component based construction of complete virtual system models.

2.2 Audience and Scope of the White Paper

This paper is intended to provide information for the following audience:

- SPEEDS individual project members:
  - all people that are actively involved tin the SPEEDS project
- the external stakeholders of SPEEDS:
  - Project officers within the European commission
  - Project reviewers
  - Managers of the participating partners
  - Associated Parties of the SPEEDS project
  - Systems-Engineers interested in new methods, processes and tools for Model-Driven Systems-Engineering

2.3 Brief description of the paper

After the introductory part (chapters 1 and 2), the white paper describes the context and objectives of the SPEEDS project, seen from today’s point of view after nearly two years in the project. It will be laid out, what the SPEEDS team envisions with regard to modelling methodology for software-intensive embedded systems, and why SPEEDS has unique and perfectly fitting solutions.

In the subsequent chapter, the SPEEDS concepts and the scientific base will be depicted, starting with the state of the practice in systems engineering (SE) in the aerospace and automotive industries, followed by short overviews of concepts and foundation on concurrent development process in the face of uncertainty (speculative design), on design-space exploration, component based design, formal methods and analysis of design models.

The fifth chapter will elaborate on the tool integration, clarifying the SPEEDS tool architecture and the cooperation of the tools in the design process. In the sixth chapter, the SPEEDS results and deliverable, as planned today, will be listed and brought into context, before the document then comes to the wide field of exploitation of the SPEEDS results, starting with a description of the business perspective of SPEEDS, the issue of handling of intellectual property and finally with a short summary of the ongoing and planned exploitation activities.
3 Context and Objectives

3.1 SPEEDS Vision

The SPEEDS project aims at significant improvements of productivity and competitiveness of the European industry in the fields of embedded system- and component design, design-quality, testing and integration and certification of avionics / electronics applications. The project aims at these significant improvements competitiveness by means of

- Semantics-based modelling to
  - Support the design of complex embedded systems using heterogeneous sub-system models, and
  - Enable sound integration of existing and new tools.
- Novel formal analysis tools and techniques that will allow to explore architectural implementation alternatives and “first-time-right” design using the “design-by-contract” paradigm.
- A new tools-supported design process, the “Controlled Speculative Design”, that will minimize the risks of concurrent design activities, through the definition of formal contracts between design groups.

The seamless integration of SPEEDS enabled COTS modelling tools into custom design flows will allow European system companies to pick best-in class tools for selected domains while reducing the cost for new vendors to enter this market segment.

4 SPEEDS Approach and Scientific Base

4.1 State of the Practice (of embedded systems design methods)

4.1.1 Introduction

Text based, document centric development is still the dominating paradigm for engineering information management. However, there is an ongoing shift from this traditional systems engineering paradigm towards a paradigm based on objects, models and views. This new paradigm is attractive as it promises improvements in terms of interpretability, changeability and model element reuse.

Multiple methods and languages have been proposed in literature evaluated and used in industry. In recent years there has been considerable focus on Model Based System Engineering (MBSE) (also referred to as Model Based System Development) as the mean for managing complexity and improving specification clarity and consistency. The acronym MBSE is used in many different contexts with slightly different meanings. For a control engineer, MBSE usually refers to model based specification and simulation of controllers in tools like MATLAB/Simulink, and for a software engineer, MBSE might involve modelling of software in a tool with code generation such as TargetLink for embedded code or a UML-based tool for classical IT-systems.

Modelling is becoming a standard and an accepted method for specification, design and analysis in each of these engineering domains. This is in contrast to Systems Engineering (SE) of complex systems where, until recently, systems’ modelling was the exception rather than the rule. This is currently changing with the introduction of languages such as UML2 and in particular SysML. Further, an increasing part of the system verification activities relies on results from analysis and simulation models rather than expensive tests.

4.1.2 System Architecture

The architecture definition includes the system boundary, its internal parts (components or subsystems), interfaces (internal and external) and behavioural responsibility of each component. To complete the picture, the top level model describes actors acting on the system but also the physical environment of the system to operate in. Different views are required to describe the use of a system in different phases of its
life cycle and to analyze possible architectures, i.e., to “explore the design space”. Several frameworks and languages exist to support architecture definition and documentation; ANSI/EIA-632, DodAF, AUTOSAR, AADL and SysML.

4.1.3 Requirements Analysis and Simulation

The field of simulation is rapidly growing. For large systems, integration of models created in different modelling environments is challenging. Two approaches of simulation model integration have emerged:

- "co-simulation" where two (or more) tools run in synchronization and interchange information during simulation and
- "hosted simulation", where a model created in one tool is generated to executable code and imported (hosted) in another tool where simulation is performed.

Tools as MATLAB/Simulink are increasingly used in industry as they support both model based design and simulation capabilities integrated in one environment. Libraries with reusable standard or user defined function/blocks reduce development time and cost but also support good quality.[F3]

In model-based development, the seamless use of executable models is characterized by functionality and control system design integrated with the subsequent implementation phase. This means that models are used throughout entire control system development: from the preliminaries to the detailed design. In the first design stage, a behavioural model (with physical aspects) is created, which is derived from the requirements specification. The behavioural model describes the behaviour of the control function to be developed, containing transformation algorithms related to continuous input signals as well as incoming events. These algorithms are usually described using floating-point arithmetic.

4.1.4 Rapid Prototyping and Production Code generation

Since the behavioural models focus on the design of the system functions and on checking its functional behaviour with regard to the stated requirements (simulation), it cannot serve directly as a basis for production (or target) code creation. Implementation details, which are the prerequisite for automatic coding, are not considered here. Therefore the behavioural model needs to be manually revised by implementation experts with regard to the requirements of production code (e.g. function parts are distributed between different tasks). For example, in order to enhance the model from a realization point of view, the floating-point arithmetic contained in the behavioural model is adjusted to the fixed-point arithmetic used by the target processor. If fixed-point arithmetic is used, the model must be augmented with necessary scaling information in order to keep imprecision in the representation of fixed-point numbers as low as possible.

Apart from the change in the type of arithmetic, it may be necessary to substitute certain model elements that are not part of the language subset supported by a particular code generator. Furthermore, it may be necessary to restructure the behavioural model with regard to a planned software design. The result of this evolutionary reworking of the behavioural model is called an implementation model. The implementation model can be used as a basis for automatic code generation with a code generator and it contains all the information that is needed for code generation and enables creation of efficient source code by the generating tool. The same code generators are normally used to produce code for simulation purpose, as well as for target code, but it may require with different generation rules/templates.

Regarding target code for safety critical systems, qualified code generators reduce the effort of software verification, code review and test. Standards such as MISRA, RTCA/DO-178C and IEC 61508 partly address issues related to automatic code generation for safety critical systems. Today, there is only one certified code generator in the market (SCADE) which is applied in practice, mainly within the aerospace and defense domain.

4.1.5 Formal analysis

Formal analysis of state and domain space is emerging as a complement to simulation in many fields. Commercial tools find increased usage in industry and there are large numbers of tools from the academic
community that approach industrial strength. Today, the capabilities of, e.g., formal model-checking have reached a state of maturity to be applied for practical development projects of embedded software systems. Main preconditions for the applicability of these formal approaches are seamless integration into the established tool-chain and the usability of a language for defining constraints on the models.

4.1.6 Tool integration

Advances in domain specific modelling has highlighted the need to integrate engineering data from different domains for the purpose of analysis or simulation as models are based on different modelling techniques and reside in tools for their respective problem area. One desirable example would be a system model integrated from three major kinds of sub models:

- models of hardware (such as resistors and capacitors in electricity, or pipes and nozzles for fluids) for performance evaluation and dimensioning
- models of the embedded software for control and monitoring of the hardware
- models of the environment of the system, e.g. other subsystems

Currently text documents and spreadsheets are the dominating formats for information exchange in systems engineering but standards for data exchange are emerging, such as:

- AP233 – ISO 10303 Standard for exchanging systems engineering data
- XMI – XML Metadata Interchange defined for the UML framework
- RIF – Requirements Interchange Format

4.1.7 Configuration management

Due to reuse of designs and relying on the product platform approach, configuration of existing parts and controlled (concurrent) changes to a system baseline, the needs and means for change- and configuration control has created a specific engineering discipline. The incremental or iterative method with short development steps and many baselines require extended support of version-, configuration- and change management. New challenge in this area is the stringent overall product configuration management by the integration of engineering configuration data handled in different systems/tools such as PDM (Product Data Management), SCM (Software Configuration Management), software build tools and system simulators.

4.1.8 Summary

The overview presented in this section clearly shows the ongoing progress in the area of model based systems engineering, but also highlight the absence of the technology that provides the integrated development environment for complex system development. It is this glue and integration technology combined with industrial strength processes that the SPEEDS project is providing.

4.2 SPEEDS Concepts and scientific foundation

4.2.1 Component-based Design using HRC (Heterogeneous Rich Components)

Despite its big success in both software and hardware domain respectively, component-based design encounters new challenges in modelling embedded systems, where software and hardware need to be co-designed, various heterogeneity are manifested, only limited resources are available, while strict performance and dependability are required. To tackle these issues the SPEEDS project provides an appropriate heterogeneous rich component model – called HRC – which is (1) expressive enough to cover the complete development cycle from high-level specifications to design models and which (2) addresses both functional and non-functional aspects. HRC components are characterized by formal contracts allowing various analysis techniques to validate a design already in early design stages. Notably, HRC-based modelling extends the conventional component-based modelling techniques by providing a unique, multi-dimensional framework to be used in all development phases. It comprises the following features:
Design by contract paradigm.
In addition to traditional static interfaces that only define the interaction points of components, richer information is exposed on the boundaries of HRCs, in terms of contracts. Attached to a component, contracts abstract dynamics constraints on the component in terms of assumption-promise pairs, with the meaning that a promise offered by the component is guaranteed only if the corresponding assumption required about the environment is fulfilled.

Organize in viewpoints.¹
Not only functional characteristics, HRC contracts also cover non-functional aspects of a component, such as real-time, safety, resource. Following the principle of separation of concerns, the multiple aspects covered in HRC contracts are further organized into different viewpoints, each of which collects a part of the component's dynamics constraints from a certain perspective. In other words, one viewpoint collects one class of contracts w.r.t. a given (non-)functional aspect, hence can be used to filter the component's characteristics w.r.t. that view. We focus on the following viewpoints within SPEEDS, (although the design of HRC meta-model allows defining arbitrary viewpoints): functional/behavioural viewpoint, real-time viewpoint, and safety viewpoint.

It is worth noticing that different viewpoints are not orthogonal. A contract may be classified into more than one viewpoint. Moreover, cross-viewpoint dependency exists. As an example, the end-to-end timing latency from the brake regulator (which sends the braking command) to the brake actuator (which receives the braking command and acts) may heavily influence the guarantee of the safety property that a car will not collide with another preceding car. In this case, a promise (of a contract) in the safety viewpoint depends on an assumption about the real-time property, which would be guaranteed by a corresponding promise (of a contract) in the real-time viewpoint. As a consequence, to express assumptions and promises uniformly across different viewpoints becomes highly beneficial. HRC meets this need by the consistent use of automata for expressing assumptions and promises. Different kinds of automata are considered for different viewpoints. For example, the functional viewpoint can be modelled by classical I/O automata with guards and actions attached to transitions and trajectories of variables attached to states, the real-time viewpoint can be modelled by timed automata, and the safety viewpoint can be modelled by automata with probabilistic inputs, etc.

¹ The notion of viewpoints used here corresponds to the notion of view of IEEE 1471-2000 definition or as used in the SysML package diagrams.
Uniform concepts across all **layers**. Principally analogous with the PIM (Platform Independent Model) / PSM (Platform Specific Model) distinction advocated by MDA (Model Driven Architecture), different layers are identified in HRC models in correspondence to different architectural abstractions of an embedded system. Examples of such architectural layers are: the functional layer that abstracts the functionality of the system; the ECU layer that abstracts the system as a network of ECUs (with the tasks on them) and buses (with the messages on them).

Layers are all uniformly represented as HRC models. More specifically, HRCs serve as the basic syntactical units of construction for all layers, and we rely on the HRC methodology to further guide and distinguish the characteristic, definition, usage, and maintenance of different layers.

Layers of one system are related via mapping, which can generally be considered as *allocating* elements of the higher-layer more logical HRC model to those of the lower-layer more physical HRC model. As an example, the mapping between the functional layer and the ECU layer would tell how to implement the system functionality in terms of tasks and messages deployed on ECUs and buses.

### 4.2.2 Component-based Development

In the concepts of HRC contracts play a central role. Contracts encapsulate the internals of a component by providing an abstract characterization of the component. It describes the promises of a component – functional as well as extra functional ones:

- The output *out* is the sum of the inputs *in1* and *in2*;
- Every request will be served;
- A request will be served within 10 ms.

A component may not work in all environments. There might be restriction to its use. For that purpose constraints may be given under which a promise will be guaranteed. Such constraints are called assumptions:

- The inputs *in1* and *in2* have to be within a given range;
- A request signal should be stable for a given period (to be recognized as a request);
- A task will be invoked with a periodicity greater than *x*.

Hence a contract will be given in the form of a pair of an assumption and a promise:

**Contract** = (Assumption, Promise)

The assumption specifies some constraints on the environment (expected property of the environment) in which the component operates, and the promise specifies what the component will guarantee provided the environment does not violate the given assumption.

![Figure 2 System Design Scenario](image-url)
We will discuss the role of contracts in a distributed design scenario as shown in Figure 2. A simplified system design will consist of the following activities:

1. Specify the system requirements in form of contracts:
   \[ C_{i}^{\text{SYS}} , \ldots , C_{k}^{\text{SYS}} \]

2. Decompose the system \( \text{SYS} \) into subcomponents (\( H1, \ldots, H4 \)), which should realize well-defined sub functions.

3. For each subcomponent \( Hi \) specify the required characteristics again in terms of contracts:
   \[ C_{i}^{\text{Hi}}, \ldots, C_{r}^{\text{Hi}} \]

At this stage the SPEEDS approach helps the designer by providing analysis techniques, which allows an early virtual integration test. Based on the contracts one can check whether the local contracts of the subcomponents together with the components interconnections are consistent with the global system contracts (see Figure 3). By this analysis method the system designer get the information whether the subcomponents are sufficiently specified such that the composition can satisfy the global system requirements.

![Figure 3 Virtual integration based on contracts](image)

When the subcomponents are characterized by contracts they can be designed independently. Furthermore, the SPEEDS framework allows that subcomponents can be implemented using different tools. Alternatively, the designer may select a predefined component from a library:

4. For each subcomponent \( H \) do one of the following steps:
   a. Iterate the decomposition into subcomponents as in step 2.
   b. Use the component interface specification together with the contracts to implement the component in a design tool.
   c. Select an appropriate library component.

Let us discuss the latter alternatives in more details.

Ad 4b (cf. Figure 4): If we provide an implementation for a specific component we have to guarantee that the implementation satisfies its requirements, i.e. that the contracts specified for that component are valid.
Validation can be done as usually by simulation or by using more advanced formal verification techniques.

**Figure 4** Validating an implementation

**Ad 4c** (cf. Figure 5): In the case we use a predefined component from a COTS library we have to check whether that library component guarantees the specified component requirements. This can be done again by comparing the associated contracts. The contracts of the library element should be dominated by the specified contracts.

**Figure 5** Check usability of library elements

Using all the indicated validation steps in the design process will lead after integration to a verified global system. Hence the SPEEDS approach allows verification of systems in early design phases. The design process discussed so far covers only one layer of abstraction. Contracts are not only used on a horizontal level to describe the interaction of a component with its environment, but the concept of contracts will also be used vertically between layers (cf. Figure 6). To validate timing properties on the functional layer one may use timing assumptions of an underlying ECU architecture. Again these can be formulated in terms of formal contract assumptions. Selecting a concrete ECU architecture the timing assumptions on the architecture have to be proven.
4.3 Contract-based Analysis/Design Methodology

In the previous Section, we have shown how contracts allow an early verification of the system model. In particular, we have shown that if the contracts of a component are dominated by the contracts of another one, defined in a library, the component can be substituted without destroying the proved propriety of the system. These operations among contracts set the basis for composition and manipulation of components composed of contracts belonging to more than one viewpoint. This is at the basis of the SPEEDS design methodology, which is built around four categories of design elements: contracts, implementations, obligations and relations. System obligations are typically high-level requirements that the designer would like to hold without considering any environment or assumption. System obligations are formally defined as assertions, i.e., sets of behaviours. An important point is that system obligations should be checked on contracts as early as possible in the design flow, because this significantly reduces the analysis effort, required to prove or disprove the obligation, and the design effort, required to revise the contract if the obligation is not met. A contract is said to be conformant to the obligation if the obligation is made true by the contract itself.

Relations are instead desired and/or validated implication among the other design elements. Dominance between contracts, satisfaction between implementations and contracts and conformance between contracts and obligations are example of already introduced relations. SPEEDS provides a methodology to orchestrate the usage of the relations and give guidelines to the user on how to design and verify her model against a number of requirements/constraints. Contracts and system obligations are specifications that are intended to guide the designer(s) towards a consistent system's implementation. Hence, in the
design process we intend to relate implementations to contracts and system obligations. In particular, implementations are used in the contexts defined by contracts and are meant to satisfy all system obligations. In more precise terms, given a component with an implementation that satisfies a contract that conforms to a system obligation, we want that such an implementation also satisfy in some sense to the system obligation. Other important relations are the compatibility and the consistency. A component is said to be compatible if a hypothetical environment can avoid violating the assumptions of the contracts given any of its possible outputs. A component is said to be inconsistent if its contract’s promises exclude all behaviours under a certain input.

We distinguish between the **Design** and the **Analysis** methodology. The former defines the design steps that the user can take for the evolution of his/her system, while the latter specifies the relations that should be established (or re-established) depending on the corresponding design step. An abstract representation of the design methodology is shown in Figure 7. Initially, from a set of requirements, we derive a system model composed of obligations and rich components and a (initially empty) set of relations between them. From this point, the user may take a number of steps, perform analysis to enhance this set of relations or to perform design space exploration. We detail on the design steps later on, after we specify the different elements of the design.

As part of the design methodology, we consider an organization of those elements in three **design spaces** and furthermore in **layers**. These are: the implementation space, the contract space and the obligation space. Relations (since they are not syntactic elements of the model) are represented as ”connections” between elements of the same or different spaces. For example, dominance ”connects” two elements within the contract space, while satisfaction ”connects” one element from the contract and one from the implementation space. Note that a relation may ”connect” an element with itself, as in the case of the compatibility relation. Each space may be further subdivided into **layers**. For SPEEDS, only the layering of the contract space is relevant, because the main purpose of the SPEEDS model is to represent contracts. However, a layering of the obligation and implementation spaces is also possible.

![Diagram](Figure_8[CH4])

### 4.3.1 Design steps

A design step is an evolution of the development of our design, which can be seen also as the evolution of the design in time. A design step is said to be **validated** if the required relations hold. This validation is performed using the high-level analysis services provided by SPEEDS, and is briefly discussed in a following Section. Moreover, we introduce the notion of **valid rich component**, that is a rich component whose contract is compatible, consistent, it is satisfied by its implementation and conforms to its obligations. When a design step is validated, and if the source rich components are valid, then also the target components are valid, which means that the resulting component can be used “safely” in place of the originating one, i.e., we can substitute without losing any verification and validation results obtained previously.
Design steps might involve single or multiple rich components. The first category contains design steps which specify or modify only one element of a rich component, resulting in a new rich component where the remaining elements are unchanged. An example of this design step is the contract modification. If the user modifies a contract of a component, then if the new version dominates the old one, then the modification impact is only local, i.e. it does not propagate to other components, and no additional verification effort is needed. Other design steps are the obligation and implementation modifications.

The second category of design steps contains those design steps having more than one source or more than one target rich components. An important design step is the decomposition: a component is substituted by a parallel composition of a set of components. Since parallel composition of contract preserves dominance and satisfaction, no additional verification task is needed for each integrated components. Moreover, if the decomposed component is dominated by the composition then it can be substituted without any additional effort.

The above design steps are all possible actions that can advance the system design and are the “bricks” to build the design methodology. The design methodology in SPEEDS uses these building blocks in a viewpoint centric approach.

The analysis services provide information about the status of the relations to the process advisor to enable a contract-based speculative methodology.

4.3.1.1 Analysis methodology

As mentioned above, the validation of a design step is important because it ensures that substitution of components can be done without losing any prior verification and validation effort. Moreover, we identified a number of relations that should be checked for the given design step. In SPEEDS, the services that check the basic relations defined are called “high-level services”. A further decomposition of the high level services is based on the analysis tools and techniques that are provided by the SPEEDS partners. These techniques can be applied to a wide range of models; from pure discrete to continuous or hybrid and we refer to them as “low-level services”, which, orchestrated by the Analysis Methodology, provide the validation of the design steps.2

5 SPEEDS Tool Architecture

A complex development process as it is supported by the SPEEDS results involves the interaction of many tools. These tools must interoperate in order to fulfil the intended development tasks. With the SPEEDS tool architecture any (commercial or in-house) tool can be made integrable. However, during the term of the project some example tool configurations will be evaluated.

5.1 Integration principles

Typically modelling tools interact via file exchange if the tools vendors agreed for a certain file format. This is a fragile, one-to-one agreement. SPEEDS has resolved this problem with the definition of a tool independent format – HRC.

2 Note that this connection can be established given the common meta-model syntax that is defined in SPEEDS.
Another current integration approach is SOA, i.e. regarding data processing as a service that is offered by the service provider to the service users. This strong mechanism enables some major achievements in synchronizing different engineers but apply only for central services, e.g. version management.

SPEEDS has brought the notation of services to a new level: Each tool participating is a service provider as it solves a certain task for the user. The SPEEDS environment now provides the infrastructure and description mechanism to realize a multi-server SOA where each communication partner may be server and user of services. This is a typical setup of a bus infrastructure and thus the glue of SPEEDS environment is the SPEEDS bus.
5.2 SPEEDS Bus

The SPEEDS bus is built on ToolNET™. It is a bus allowing to realize the data exchange as known from the file exchange and exchange of control commands among tools by invocation of remote services.

---

**Figure 11** SPEEDS - open service network

Hierarchical development chain

Living network

pre SPEEDS

SPEEDS: Bus network of service providing tools

---

SPEEDS White Paper
Figure 12 SPEEDS bus supporting the SPEEDS environment

A tool is connected to the SPEEDS bus by a tool specific adapter. This adapter is the top layer of the bus architecture and the only tool specific implementation. Everything below, including the services and service interaction sequences are defined tool independent. This allows in SPEEDS (and later at the user premises) to customize the development process and implement it with the most appropriate tools.

Figure 13 Example service interaction diagram

The potential implementation of the abstract tool types is shown in Figure 11. The standardization in terms to services makes them exchangeable without leaving the defined process.
The introduction of the SPEEDS bus as one central means of interaction among development tools allows to access process relevant data in a unified way. Additional to the control of service interaction sequences facilities like central logging and event notification are realized. Together with the status of the system under development – measured by using the unique set of SPEEDS analysis tools – process metrics can be derived online and process advice is provided. This online consolidation of process information is the task of the SPEEDS Process Advisor tool.

5.3 Process Advisor Toolset

The SPEEDS Process Advisor toolset is going to be an application, which provides three main functionalities for the development environment: Assessment, process support and advice.

The assessment functionality facilitates the user interaction with the high-level analysis and design services. This includes entering analysis parameters, calling concrete low-level analysis services using the SPEEDS Bus, enriching the results with context information and storing them into the repository. Additionally a set of new analysis features will be offered that are required to compute metrics, which measure other characteristics of the system under development like for example ‘maturity’. Visualization techniques will be used to express aggregated analysis results like for instance trend analyses. In consequence, the Process Advisor is established as a centralized platform to initiate and evaluate analysis functions.

The process support functionality provides the means to model and monitor the speculative development process. It is possible to define decision gates and set expected goals for them, so that the progress can be monitored based on various assessments throughout the project.

---

3 HST – Hosted Simulation Tool, ACT – Architecture Composition Tool, BMT – Behaviour Modelling Tool
The *advice* functionality will aid users in interpreting results of aggregated analyses and provide solutions to emerging problems. Some of the more complex analyses e.g. a maturity metric, base their result on the aggregation of multiple intermediate analysis steps. These intermediate analysis steps are not necessarily needed, but deliberately implemented. They allow reasoning about the synthesis of the analysis result and point to detailed problem areas. Thus we are also able to offer a set of possible design steps to solve the sources of these problems. This becomes especially interesting in conjunction with decision gates: The difference between the current (sub-) system status and the goals defined for the next decision gate form the set of current problems to be considered. The Process advisor determines major factors in association to these problems and proposes design steps to solve those.

### 6 SPEEDS deliverables

#### 6.1 Overview

The SPEEDS project aims at the definition of a new methodology for model-driven systems engineering. According to this objective, the results of SPEEDS will have to comprise results

- regarding the adaptation of existing methods or the definition of new methods
- regarding the usage of new tools or the integration of existing tools
- regarding the definition of new or the adaptation of existing processes, incorporating the results mentioned above.

SPEEDS results will be made available in two different ways:

1. by providing a working tool chain that integrates all developed and adapted tools
2. by providing textual descriptions and specifications of the results within the project (for internal use only) and for external stakeholders

In the following, the SPEEDS deliverables that will reflect the project results will be discussed

#### 6.2 SPEEDS Process

The SPEEDS methodology encompasses the so called “speculative design”, an approach to handle uncertainty in the systems-engineering activities, allowing real concurrent design processes with a minimum of re-work, a small number of iterations and fast cycles through the iterations.
In order to control the speculative design, a dedicated “controlled process for speculative design” is being developed, that incorporates measurement of progress and technical maturity. The SPEEDS process will be implemented in the process-advisor tool, a support tool that will help design engineers to exploit the SPEEDS advantages to the fullest extent.

The process of controlled speculative design can be seen as a control mechanism of a design model under uncertain circumstances or less evidence, and also ensuring the achievement of an ultimate goal of the design. It uses the continuous checking and monitoring verification technique to guarantee that the design will meets its objectives, or rapidly readjusts the process when the design shows its negative tendency.

SPEEDS process is expected to support a design for different types of product, different ways of work and different business sectors. Therefore, the process itself needs to be flexible enough to adapt to the different working environment. In consequences, the SPEEDS process tries to provide the following ability to the users:

- Every design in different levels can start at early phase
- The process can handle incoming changes at anytime of the project
- Ability to reuse the existing design component from the previous development
- Adjustable to different working methods; e.g. contract first or model first
- Possibility to integrate SPEEDS process to the existing industrial methodology
- Ability to customize the control method based on a particular project’s speculative factors

In the SPEEDS process way, the different design teams (in the same or different design level) can start performing their work about in the same time during the very early stage of the project, without waiting for a completion of higher-level design. Due to this fact, the level of uncertainty of the design can be relative high. However, in order to control and direct the progress of the whole project, SPEEDS process provides a view of a close-look to the progress of the design. The control view can also be parameterized for each type of actors of the process, whether in global control view for chief system engineer or detailed system control view for sub-system designer.

6.3 SPEEDS Methods

The term “SPEEDS methods” contains two main areas, modelling and analysis. Accordingly, the main results are distributed between the two responsible sub-projects (SP.2 / SP.3).

6.3.1 HRC modelling methodology

The modelling methodology in SPEEDS is based on the meta-model for Heterogeneous Rich Components (HRC), which is the logical foundation for the integration of different modelling tools. Accordingly, the sub-project 2 has to define the meta-model (WP.2.1). This will lead to the deliverable D.2.1.5 “SPEEDS Meta-model (Final Document)”, using other intermediate deliverables as the step stones (D.2.1.1, D.2.1.2, D.2.1.3 and D.1.2.4).

The definition of a meta-model is incomplete without an implementation with respect to suitable tools, which is the task of WP.2.2. In this work-package, Eclipse EMF will be used to create a common interchange format that can read and written by design tools. The work-package will produce the (software-) deliverable D.2.2.2 “Meta-model implementation” (with one intermediate version D.2.2.1), an implementation based on SysML, defined as a SysML profile (D.2.2.3 “Profile implementation”) and as an implementation basis for the contract specification- the (software-) deliverable D.2.2.5 “A/P Editors” (with intermediate version D.2.2.4).

In order to be able to fully integrate different modelling tools, it is necessary, to integrate the respective modelling tools simulation capabilities, which is done in WP.2.3. For this purpose, the work-package will give us the definition of an application programmers interface (API) needed to integrate hosted simulation approaches in the modelling tools, D.2.3.3 “Hosted simulation definition” (with intermediate version D.2.3.2).

The data-exchange between design-tools, and the corresponding integration of the modelling tools into the SPEEDS tool environment is the content of the work in WP.2.4, only with this task solved will the tool-chain be complete. Accordingly, the deliverables of this work-package will be a number of software deliveries:
1. D.2.4.2 “Integration of COTS tools in the SPEEDS modelling environment” (with intermediate version D.2.4.1)
2. D.2.4.3 “Hosted Simulation”

Finally, the WP.2.5 addresses the definition of a new modelling methodology, making use of the meta-models’ and tools’ capabilities within the context of the SPEEDS process. Based on an analysis of the state-of-the-art given in D.2.5.1 “Report on existing standards”, the HRC modelling methodology will be defined in D.2.5.3 “HRC methodology” (using intermediate version D.2.5.2). Additionally, the Contract Specification Language (CSL) and its usage in the modelling activities will be defined in D.2.5.5 “Contract Specification Language” (with intermediate version D.2.5.4)

6.3.2 ADT analysis methodology

For the analysis methodology it will be important to integrate existing analysis approaches and tools as well as to widen the scope of the analysis methods. One very important issue with respect to the second topic is, to make analysis feasible in view of an increasing complexity of models. The definition of such scalable analysis methods is the work executed in the WP.3.1, leading to D.3.1 “Scalable Analysis Methods”, using the intermediate deliverables D.3.1.a and D.3.1.b.

The implementation of the methods defined in WP.3.1. in software tools will be done in WP.3.2, finally ending up with the availability of suitable software packages D.3.2 “Prototype Implementation of Analysis Tools” in different versions, with increasing capabilities. Finally it will of course be necessary, to define the usage of the analysis methods in the context of the SPEEDS process and the HRC modelling methodology. This definition will be delivered by the WP.3.3, leading to D.3.3 “ADT design and analysis methodology” with intermediate versions D.3.3.a, D.3.3.b and D.3.3.c.

6.4 SPEEDS Tools

From the implementation point of view, the definition of methodologies and the delivery of single tools is not fully sufficient, if the different modelling and analysis tools can not be integrated. For this purpose, the WP.4.2 will have to provide the SPEEDS software infrastructure and facilitate its implementation. According to the layered approach used here, this will lead to the following (software-) deliverables:

1. D.4.2.1 “SPEEDS Environment Architecture” with intermediate results D.4.2.1.a “Bus Layer” and D.4.1.c “Integration Layer”
2. D.4.2.2 “Process Flow Support”
3. D.4.2.3 “SPEEDS Environment & Operation”

While the first deliverable will ensure the seamless integration of modelling- and analysis tools, the second deliverable allows supporting the SPEEDS process, giving the users the support needed to exploit the SPEEDS capabilities to the fullest possible extent. The last deliverable then will provide the fully integrated tool-chain, incorporating the environment, the process flow support and the tools as defined in the next sections.

6.4.1 Modelling Tools

The following modelling tools will be integrated by the end of the SPEEDS project:

- Geensys “RT-Builder”
- The Mathworks “Simulink”
- Telelogic “Rhapsody”
- Esterel “Scade”
7 Benefits of SPEEDS

SPEEDS technical achievements can be classified in 5 hot topics:

- HRC meta model for standardized model exchange and contract based system development methodology
- SPEEDS bus for customizable tool chain integration
- Hosted Simulation API for early system integration and validation
- Analysis tools for assessing the status of the system components at any time
- Speculative Process support for easier, targeted guidance of the development steps

These multiple benefits result in a concerted action, the SPEEDS paradigm. Nevertheless, the achievements can be used independently which allow to target the situation of each involved stakeholder.

7.1 Tool Users

Tool users are experts in developing embedded system that rely on COTS tools to achieve their results. Of course, the biggest achievement is the co-operation facilities of multiple tools that respects the flexibility required by modern business processes, c.f. Figure 11 top right.

This is further strengthened by the common understanding of the process as a common model (HRC) providing a common, cross tool methodology of contract based design. Contract based design is considered as one major achievement of reaching embedded system quality faster and in a more controlled manner. The publicly available exchange mechanism via HRC is a key for attacking new challenges: Existing and coming safety standards like ISO 26262 require an independent certification of all tools involved, especially when safety relevant systems are to be developed. And certification relies on transparency and vendor independence.

At an every day basis the tools users experience the comfort of seamless tool chain by SPEEDS environment without compromising the work in existing tools.

7.2 Tool providers

Tool providers receive with SPEEDS proven technology for realizing tool enhancements, both for new interaction among tools and for new modelling capabilities. This joint effort of the SPEEDS project results in a stability and quality of the achievements that would be far beyond the capabilities of the involved companies otherwise.

It particularly results from the fact that both tool vendors and user cooperate closely which ensures that user expectations are met.

The increased tool capabilities allow creating new business models like providing new services for tool chain integration or proving standardized, user domain specific libraries and tool setups.

7.3 Research

Researchers benefit from the increased technical capabilities of the SPEEDS enhanced tools by finding acceptance and experience from the tool users that guides the direction of research. The formal definition of SPEEDS abstract results, i.e. meta models and interfaces, are a much stronger basis to realize further research that proprietary definitions. Such research promises to be clearly related to the existing definitions allowing an easy evaluation of the practicability by users and tool providers.

7.4 Final product user

The whole project is created to better service the European public. Embedded systems are of ever increasing importance and standardization and openness of methods and tools will lead to future innovation in ES products.

This must be and is achieved by allowing faster innovation and higher quality in parallel. As an example, the constant analysis of the system under development using the contract based design allows to run more debugging cycles earlier, reaching final test phase and production with one single shot.
8  SPEEDS Intellectual Property Handling

Handling of the IP of the project shall help attaining three goals:

- Maximising exploitation of projects results
- Allowing further open access to SPEEDS results
- Allowing coordinated development based on SPEEDS project results

Therefore the IP will be handled in a threefold manner. Formats and APIs are published including example implementations. This accounts for an easy adaptation of the provided information by third parties. This publication is part of the continuation of the SPEEDS project into an open but managed consortium.

Some implementations in SPEEDS are jointly created foreground that is shared among the project partners also on source code level. The exploitation plan under development at executive level of the partners will define a further exploitation towards the end of the project.

Other proprietary side ground information created in SPEEDS is shared on a free basis among project partners for the sake of providing best quality solutions to the customers in the project. It will be part of the results of the executive exploitation group to establish an equally successful instrument for the time after the project contract is finished.

SPEEDS is successful acquiring recognition by researchers in other research projects. The project follows a liberal publication strategy with information of potential interested parties through multiple channels: industry tradeshows, scientific conferences and direct communication with industry consortia like AUTOSAR. Through the cross-linked SPEEDS partners participating in other projects SPEEDS concepts are discussed throughout the whole technical domain.

9  Exploitation

The SPEEDS project produces many exploitable results. It is a clear goal of the participants to realise these benefits, incl. new business for the tool vendors. The different results are independently exploitable results which allow minimizing the exploitation risk and eases the flow of SPEEDS results in the companies’ individual strategies.

The different technologies combined in SPEEDS jointly create maximum benefit for system developers. A successful exploitation of these technologies requires the creation of an SPEEDS ecosystem. Ecosystem includes available tools, solutions, customers, competition, and services around the exploitable results.

The SPEEDS partners consider the SPEEDS results a well-formed basis for building such ecosystem. This is shown in the ITEA initiative of providing the SPEEDS environment as reference technology platform for future joint industry projects. This major European wide effort will directly benefit from the integration and cooperation technologies realized in SPEEDS.

The partners in SPEEDS consortium offer a strong starting point, as several competitive networks exist inside the consortium’s tool vendors and customers. Competition generates the forces that ensure quality and sustainability of offerings; the joint IP on foreground during the projects ensures that no entry restrictions exist.

The exploitation requires coordination – technically and in management – which is represented in a two level exploitation management.

The executive management group is formed by executive managers of those partners, which are undertaking commercial exploitation efforts of SPEEDS results. Their role is the identification of competition, the alignment of exploitation strategies avoiding legal conflicts and the concentration on business goals.

Common foreground is enabling definitions and implementations like the HRC meta model or the HRC MM reference implementation. The executive management group assigns the budget enabling the common further developments and alignment of these developments.

Such common foreground will be open to other users as well. During the project duration, associate partners are invited to participate and strengthen the SPEEDS group. After project an open consortium will be formed for the common IP. The goal is to achieve restricted open IP, i.e. open source but with limited licensing in order to avoid uncommitted modification.

The projects exploitation team realises the work set by the executive group. Its tasks include the identification of exploitable results, merge with companies’ product plans and technical definition of each individual exploitable offering.
All SPEEDS results are close to emerging technologies and promise to strengthen the exploitation capabilities. For instance, HRC meta model is specifically suitable becoming aligned to system development standards like SysML or AUTOSAR. This proximity also ensures the interoperability to market-proven tools which allows easy adaptation and migration for users towards using SPEEDS technology. The realization of these goals is subject of a common initiative of SEEDS partners of providing a SPEEDS–AUTOSAR showcase as well as a close communication with the relevant AUTOSAR working groups. The meetings and demonstrations are aligned with the AUTOSAR consortium schedules for maximum impact.