Final Thesis

Portable Automotive Electronic Models using Standard XML Technologies

by

Syed Zia Akbar Zaidi

LiTH-IDA-Ex-02/71

2002-06-04

ABSTRACT

Today embedded systems are everywhere. They control most of the devices around us, from phones to vehicles. This thesis concentrates on automotive electronic systems, which are a special class of embedded systems that are responsible for controlling the functionality of a vehicle. Automotive electronic systems are complex distributed systems that have important requirements on performance, safety and cost.

Designing and developing such systems is not the sole responsibility of the vehicle manufacturer. Suppliers provide many of the components, and the task of the manufacturer is to design, develop and integrate different systems. Until now, the design of embedded systems has concentrated on developing systems that perform as efficiently as possible their dedicated functions. However, with the increase in complexity, other issues like the formal modeling of the requirements and the interoperability between the manufacturer and the suppliers become important.

In this thesis we have addressed the problems of interoperability and modeling of the requirements by using standard XML technologies. Thus, we have defined a simple architecture definition language (ADL) used to model the automotive electronic systems. Our simple ADL has an XML representation that makes the models portable among manufacturers and suppliers. Not only the portability of models is important, but also of the immense requirements. Our major contribution is the modeling and representation of automotive electronic requirements using XML technologies like xlinkit, XQuery and CommonRules.

Our approach has been validated through an industrial case study representing a vehicle cruise controller. The cruise controller has been modeled using our ADL, and the requirements placed on the cruise controller have been modeled using xlinkit, XQuery and CommonRules.

Acknowledgement

I am thankful to Petru Eles and all my colleagues at Embedded Systems Lab for providing me a pleasant working environment. My special thanks to Paul Pop whose keen supervision and invaluable guidance at every stage helped me a lot in carrying out this thesis work.

Contents

1 Introduction

The increase in performance of micro controllers with decrease in price makes it economically attractive for the manufacturers to replace the conventional mechanical and electronics control system within many products by an embedded real time computer systems. It is often not very obvious in the newer devices that the real time embedded computer is now controlling the product behavior because the external interface or man-machine interface of the product often remain same as for the previous product generation [2].

An embedded real time computer system is not a complete device but it is always a part of a larger well-specified intelligent system. An intelligent product consist of a mechanical subsystems, the controlling embedded system and, most often, a man-machine interface.

We can say that embedded systems are systems that are not necessarily computers with standard monitor, CPU and keyboard. The use of embedded system is more than the use of computer systems in our daily life. They are present in automobiles, airplanes, cellular phones, pagers, microwave ovens, washing machines, ammunition systems, medical devices and electronic goods.

We can safely say that most of the devices, which we are using now a day, are controlled by embedded systems. The microprocessor market share in the year 1999 is shown in Figure 1. As we can see from the figure that more than 99% of world's microprocessor are used in Embedded Systems in comparison to only 1% that are used in general computer systems [1].

This thesis concentrates on automotive electronic systems, which are a special class of embedded systems that are responsible for controlling the functionality of a vehicle. Automotive electronic systems are very complex distributed systems that have important requirements on performance, safety and cost.

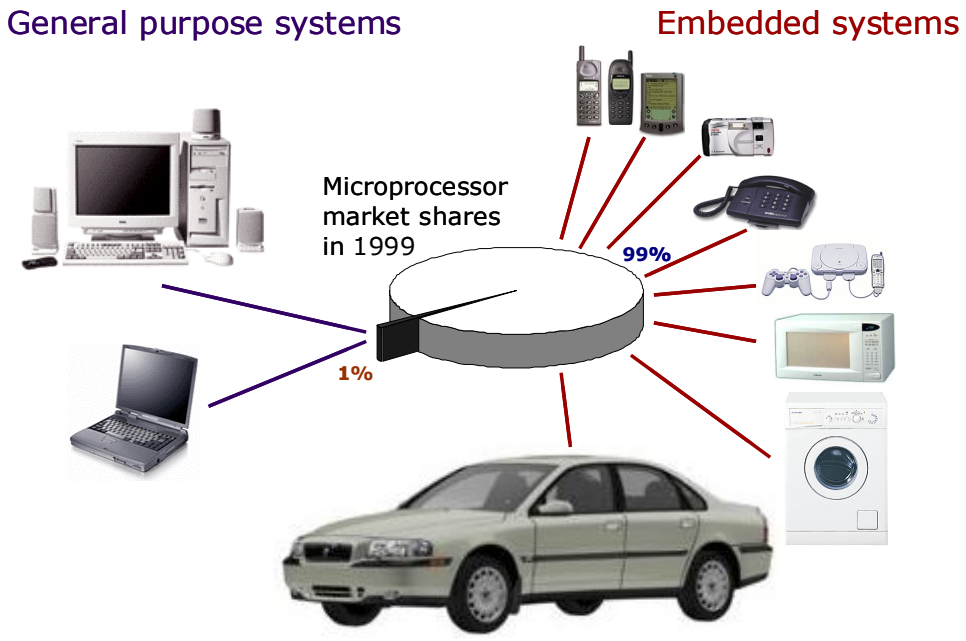


Figure 1. The microprocessor market share.

Designing and developing such complex systems is not the sole responsibility of the vehicle manufacturer. Suppliers provide many of the components, and the task of the manufacturer is to design, develop and integrate different systems. Until now, the design of embedded systems has concentrated on developing systems that perform as efficiently as possible their dedicated functions. However, with the increase in complexity, other issues, like the interoperability between the manufacturer and the suppliers become important.

For example, in Figure 2, the carmakers and the suppliers have to exchange design information with their own manufacturing plants, and between themselves. In such a context, the problem of interoperability becomes very complex and also important.

In order for the stakeholders to communicate, they have to decide on a common understanding of the concepts in the automotive electronic field, together with a standardized representation. The result of such an agreement is an architecture description language (ADL) that is used to model the design entities.

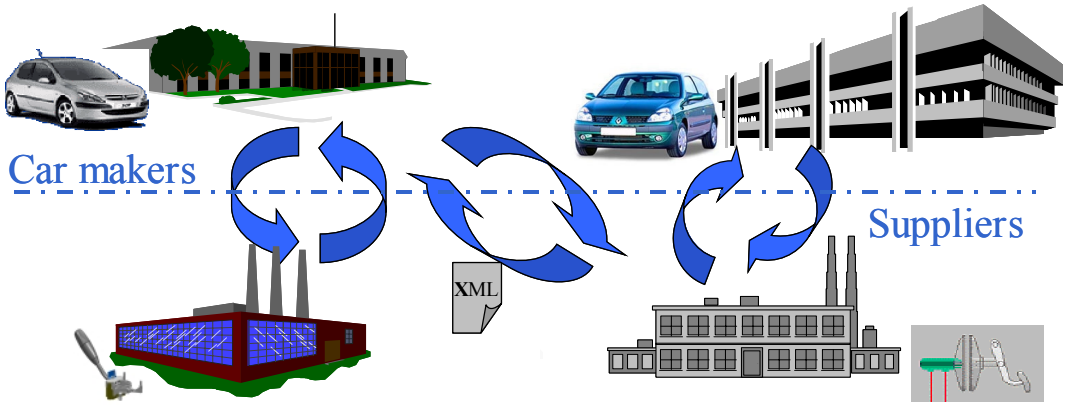


Figure 2. The automotive electronics industry

Not only the portability of models is important, but also of the immense requirements on the system. The requirements are typically separated from the design entities, and these are mainly represented as simple textual descriptions. In order to be sure that the plant is manufacturing the desired system, or that the suppliers are designing the system as agreed in the specification, the requirements have to be modeled in a more formal way, and they have to be provided at the same time with the model of the design.

1.1 Contributions

In this thesis we are addressing the problems of interoperability and modeling of the requirements by using standard XML technologies.

Our contributions are the following:

- Using the document type definition (DTD) language we have defined a simple architecture definition language (ADL) used to model the automotive electronic systems. Our simple ADL has an XML representation that makes the models portable among manufacturers and suppliers.
- We have identified and analyzed the constraints of the automotive electronic field, and what are the types of requirements that they lead to.
- Our major contribution is the modeling and representation of different types of automotive electronic requirements using XML technologies like xlinkit, XQuery and CommonRules.

- We have used an industrial case study representing a vehicle cruise controller to validate our approaches. The cruise controller has been modeled using our ADL, and the requirements placed on the cruise controller have been modeled using xlinkit, XQuery and CommonRules.

This thesis consists of 6 chapters. The next chapter presents the automotive electronics field, and what are the research challenges that we are trying to address in this thesis. Chapter 3 introduces the vehicle cruise controller system that we have used as a case study for our approaches. The simple ADL is detailed in chapter 4, together with the resulted model of the cruise controller. Chapter 4 also presents a short overview of the XML, and the motivation for choosing it as the representation mechanism. The general problem of requirements engineering is discussed at the beginning of chapter 5, and several requirements on the cruise controller are identified. Starting from an analysis of the match between the types of requirements and the existing XML technologies, several ways of modeling the cruise controller requirements are presented. The last chapter our conclusions, with pointers towards possible future work.

2 Automotive Electronics

Due to technological advancements and cost improvement in the semiconductor industry with the increase in demand of the market for improved vehicle performance, less fuel consumption, efficient exhaust and high safety requirements, the use of computer-controlled functions onboard a car is increasing.

The decreasing cost of the microelectronic components with the ever growing demands of the client and the society on the functionality, dependability and environment compatibility of a car (e.g., driving comfort, handling, fuel efficiency, safety, minimal pollution etc) have opened a big new market of automotive electronics.

There is a tremendous increase in the quality and quantity of sophisticated electronic systems in the past few decades. The cost of electronics has reached to over 23 percent of the total manufacturing cost of vehicle. Almost 80 percent of automotive innovations are in the electronic systems. According to [6] the average cost of electronic systems and silicon components such as transistors, microprocessors, and diodes have increased from $110 in 1977 to $1800 in 2001. More than fifty million cars are produced annually in the world, due to which the volume of the automotive electronics market is huge.

Many automotive manufacturers now consider the proper exploitation of computer technology as a key competitive factor in the ever-growing demands of increase vehicle performance and reduced manufacturing costs. Some years ago, the computer application onboard a car focused only on non-critical body electronics or comfort functions. Today, there is a substantial growth and development in the computer control of core vehicle functions including engines, brakes, transmission, suspension etc. These functions are also developed to increase the stability of vehicle in critical driving maneuvers. There are a lot of safety implications for the control of many of these core vehicle functions.

An automobile comprises of number of sophisticated subsystems including engine, transmission, chassis etc. These subsystems individually contain hundreds of

different components. The smooth interaction of these components provides the transportation facility system.

The application areas of automotive industry are usually divided in three parts i.e., body electronics, system electronics and networking [3].

- Functions that do not directly effect the movement of car e.g., window control, dashboard display controls, mirror adjustment are included in the area of body electronics.
- System electronics is concerned with the control of those functions that are related to the movement of the vehicle.
- These functions share certain parts and cooperate with each other for smooth functioning. Therefore, there is a need for a communication structure that integrates all these functions coherently. This is the area of in-vehicle networking.

2.1 Automotive Electronic Architectures

Automotive electronic architectures consist of several ECUs (Electronic Control Unit), interconnected in a network (see Figure 3). The hardware of each ECU is composed of a CPU, memory, I/O to sensors and actuators. An important hardware component of each ECU is the communication controller, particular for each communication protocol.

2.1.1 Communication Network

In the automotive industry there are several communication protocols that are commonly used, or planned to be used: Time-Triggered Protocol (TTP), Controller

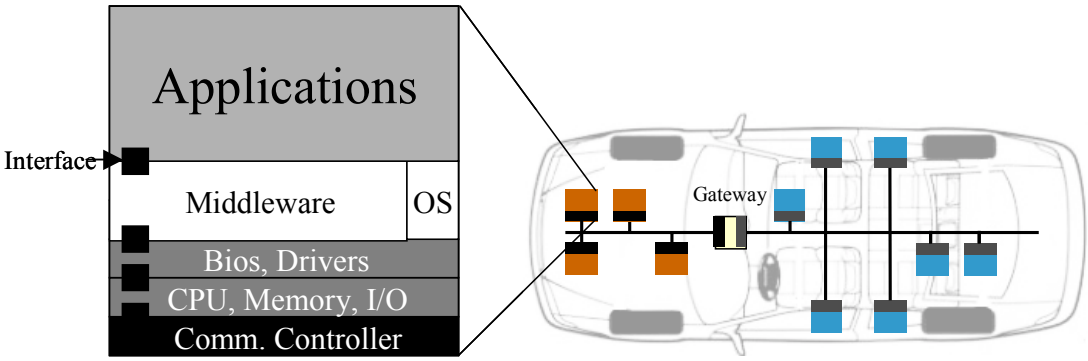


Figure 3. Generic Electronic Architecture

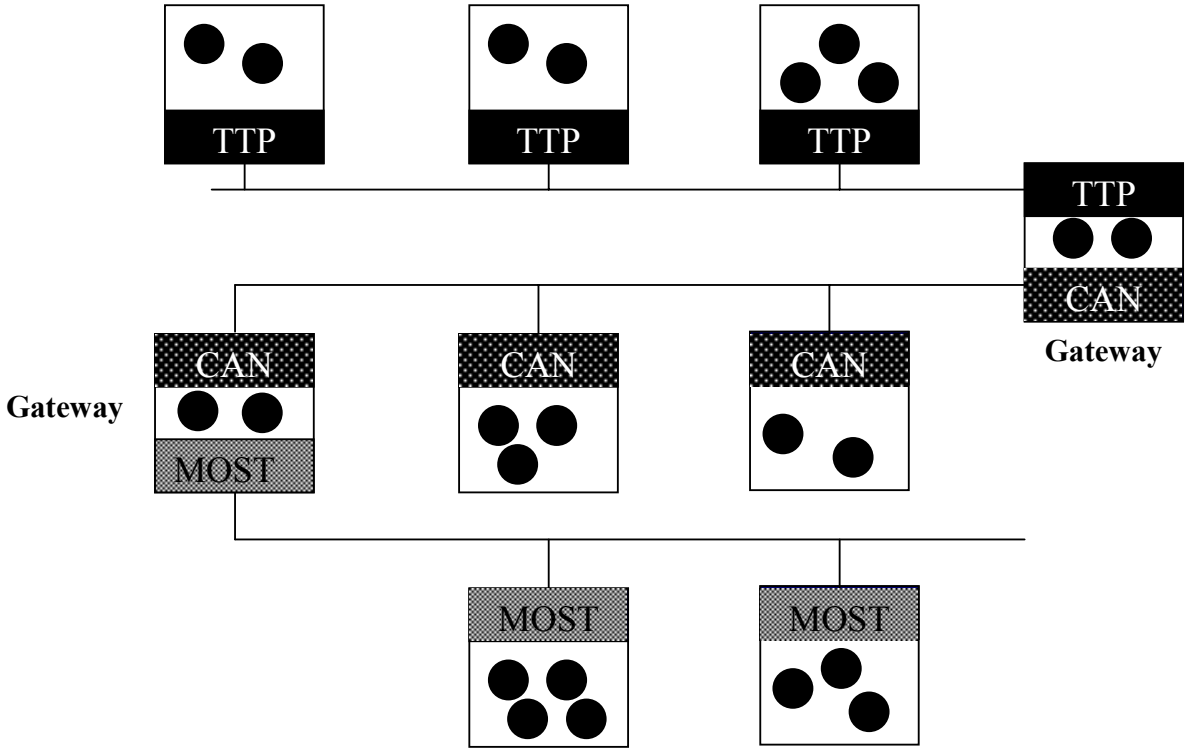


Figure 4. Communication Network

Area Network (CAN), Local Interconnect Network (LIN), Media Oriented Systems Transport (MOST), etc. We do not intend in this document to give a description of each of these communication protocols. However, for more information about these protocols, the reader is directed to [1]. It is important, however, to note that TTP and high speed CAN are intended for safety critical applications found in the powertrain, chassis and safety critical systems. Low speed CAN and LIN are used mainly in the body domain. While MOST is targeted towards multimedia in-vehicle entertainment systems. Also, the bus access policy differs from one protocol to another. For example, TTP employs a Time-Division Multiple Access (TDMA) scheme, while CAN uses Carrier Sense Multiple Access/Collision Detection with Arbitration on Message Priority (CSMA/CD +AMP).

Although there are many types of interconnections that can appear in a distributed architecture, only certain topologies are common in practice. Figure 4 considers, for example, a TTP network interconnected with a low speed CAN network via a dedicated gateway, and a MOST network which is connected through the CAN network, one of the ECUs on the CAN network serving as a gateway (see Figure 4).

2.1.2 Software Architecture

The vehicle's electronic features are realized by several applications mapped on the nodes of the architecture. The automotive applications have different requirements, depending on their domain, e.g. body, powertrain, etc. In this thesis, we are especially interested in control-dominated applications that have strict timing requirements. These applications can be distributed over several ECUs and networks, each network with its own communication protocol.

To be able to reason about the timing constraints, we consider the applications at a lower abstraction level, where an application consists of a set of interacting processes, or tasks. The process model is assumed to be similar to the process model of OSEK [8].

The architecture of an ECU can be either time-triggered, event-triggered, or a combination of both. In the *event-triggered* approach all the activities occurs with the significant change of state. The significant events are brought to the attention of the CPU by the interrupt mechanism. In the *time-triggered* approach all the activities are initiated at predetermined points in time. Thus, there is a time interrupt in each node of a distributed time-triggered system, and it is assumed that the clocks of all nodes are synchronized to provide a global notion of time.

The operating system in each ECU is OSEK compliant [8]. The OSEK OS is an operating system for event-triggered systems, while OSEKtime OS is an operating system especially tailored for the needs of time-triggered architectures. These two OSes can coexist with each other, OSEK OS running in the idle time of OSEKtime.

The scheduling policies can differ from one node to another. Event-triggered systems typically require *preemptive priority-based scheduling*, where the appropriate process is invoked to service by the event. Time-triggered systems typically require *non-preemptive static cyclic scheduling*, where the process activation or message communication is done based on a schedule table built off-line.

The software architecture is composed of a middleware and an operating system. The communication among the components of the software architecture themselves, and with the applications and the hardware is done via clearly defined interfaces, represented with black squares in Figure 3.

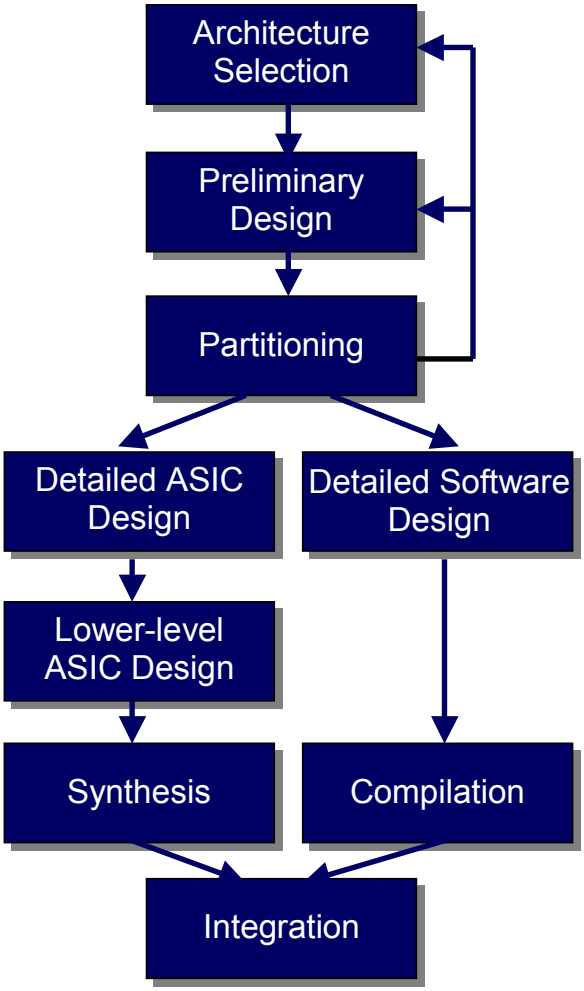


Figure 5: Waterfall Design Model

2.2. Hardware Software Co-design of Automotive Systems

This section presents the current design practices in the automotive industry. It will further introduce a new design methodology to reduce the design time and increase the quality of the implementations. The design methodologies currently in use are no longer successful for increasingly complex automotive electronics. These become more complicated and difficult due to constraints imposed by the field.

In spite of great diversity of automotive electronic functions, they share a number of common characteristics. They are usually control-based systems with an architecture that makes use of specific CPUs, DSPs, ASICs and other additional hardware devices depending on the function performed. These systems use analog-to-digital converters, serial interfaces, parallel ports, external bus access,

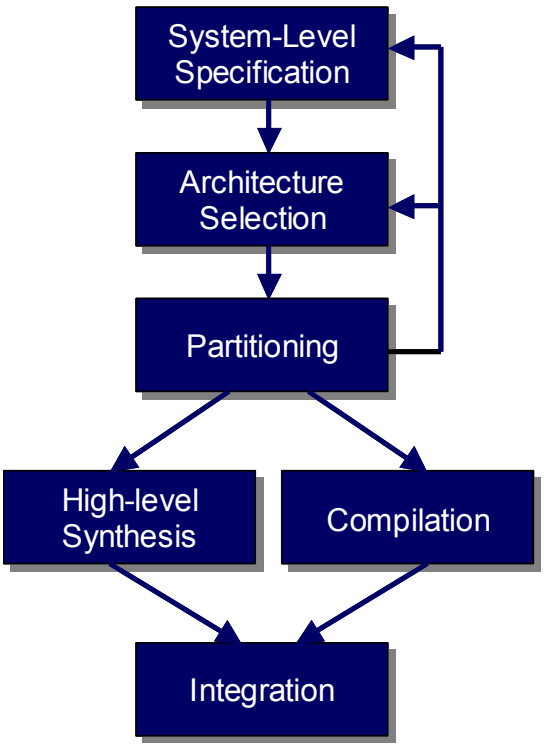


Figure 6: Typical Co-Design Flow

programmable timers, etc in order to read sensors and update actuators. We can fit most of the automotive electronic systems into the broad category of Real-Time Hardware/Software Embedded Systems. These computer-controlled functions have precise deadlines, which can lead to undesirable situations, if not met.

Waterfall model of system development shown in Figure 5 is very important for the design and development of automotive vehicles. The architecture of a system is defined on the basis of design process, which starts with the complete specification. Then, different teams develop the hardware and software parts independently. Software code is written, the hardware is synthesized and they are expected to integrate perfectly in the first attempt. Testing on hardware and software is usually done separately, with very few integration tests. A condition might rarely occur, which could force a reiteration of previous phases [4].

Today it is impossible that a complete specification can be made available right in the start of the design process and this specification will not be changed later. With the increase in system complexity, it is becoming difficult to have clear idea about

the system functions from the beginning. Based on the specification, it is also not possible to precisely determine the appropriate system architecture and the use of resources. Moreover, a separate design processes for hardware and software (which are actually dependent on each other) causes a less efficient designed system. It is often marked with bad performance because of the incomplete exploration of the trade-offs between the software and hardware domains.

A lot of research has been done over the last few years to use Hardware/Software Co-design for the development of the systems in the automotive electronics field.

The automotive industry have very tight cost margins and at the same time demanding requirements for e.g., performance and reliability. This makes HW/SW co-design attractive since it can help the designer better trade offs between Hardware and Software.

A typical co-design flow starts with an abstract homogeneous behavioral description. This specification of the system can be subject to changes depending on later stages of the design (the feedback arrows in Figure 6) as it may not be complete description in the start. This behavioral description contains no assumptions about the later implementation of different parts [4].

Designers evaluate different alternatives for implementation in the start and it is also possible to move parts of functionality, which have a uniform description, between hardware and software. However, the bottleneck on this case would be a great pressure to reuse previous design components. Further, the application area diversity restricts the use of a homogeneous behavioral description. A more practical approach is that every specific domain (e.g., control theory for the power train, human machine interaction for the information systems, ergonomics for comfort functions, etc.) will use its own specification techniques.

The responsibility of a designer for the hardware architecture is to decide the components to be included and their interconnections. This is called architecture selection. A proposed generic architecture for an automotive system is shown in Figure 3. A node in this architecture is composed of one or more programmable units (CPUs, DSPs), memory, may be some ASICs (application-specific integrated circuits), I/O devices set (sensors/actuators) and a network interface to communicate with the other nodes. The communication network is distributed, and it is required to have a certain degree of determinism.

After the selection of the architecture components, the designer has to decide what part of the behavior should be implemented on which of the selected components (hardware/software partitioning). He also has to look for the execution order of the resulting tasks (scheduling). The scheduling of the tasks in this distributed environment is a big challenge. It becomes more complicated when nodes have several functioning modes and they are dependent on the other nodes' execution. The designer has also to consider the synchronization of the different clocks needed for ensuring a deterministic scheduling.

All these design steps can overlap each other, and they can be assisted by (semi) automatic synthesis tools that enable an efficient exploration strategy aimed to reach a level in the design space which corresponds to an implementation that is as close as possible to the optimal one.

3 The Cruise Control System

We want to test our approaches using an industrial case study. We have used the vehicle cruise controller case study from Volvo Technological Development Corporation [5].

The cruise controller is an important feature in cars especially for long drives on big and straight roads. It would be more tiring and unpleasant to drive for long trip without cruise controller.

The cruise controller has the following functionality:

- maintains a constant speed for speed range from 35 km/h to 200 km/h
- offers an interface (buttons) to increase or decrease the reference speed
- is able to resume its operation at the previous reference speed
- The CC operation is suspended when the driver presses the brake pedal.

3.1 Behavior

The behavior of the cruise controller is modeled using a so-called *conditional process graph* (CPG) that consists of 32 processes, and is presented in Figure 7 [1].

A c*onditional process graph* is an abstract representation of an application. Each node represents one process denoted by circles. A process is a sequence of instructions, and has a worst-case execution time (depicted to its right). Processes are mapped to the ECUs of the architecture; the hashing in circles of Figure 7 indicates the mapping.

The dependencies between processes are represented with edges (arcs between two circles). The nodes represented with solid circles, are called *communication processes*. These are introduced during mapping for each connection, which links processes assigned to different ECUs. These processes model interprocessor communication. The execution time, depicted on their left, is equal to the corresponding communication time.

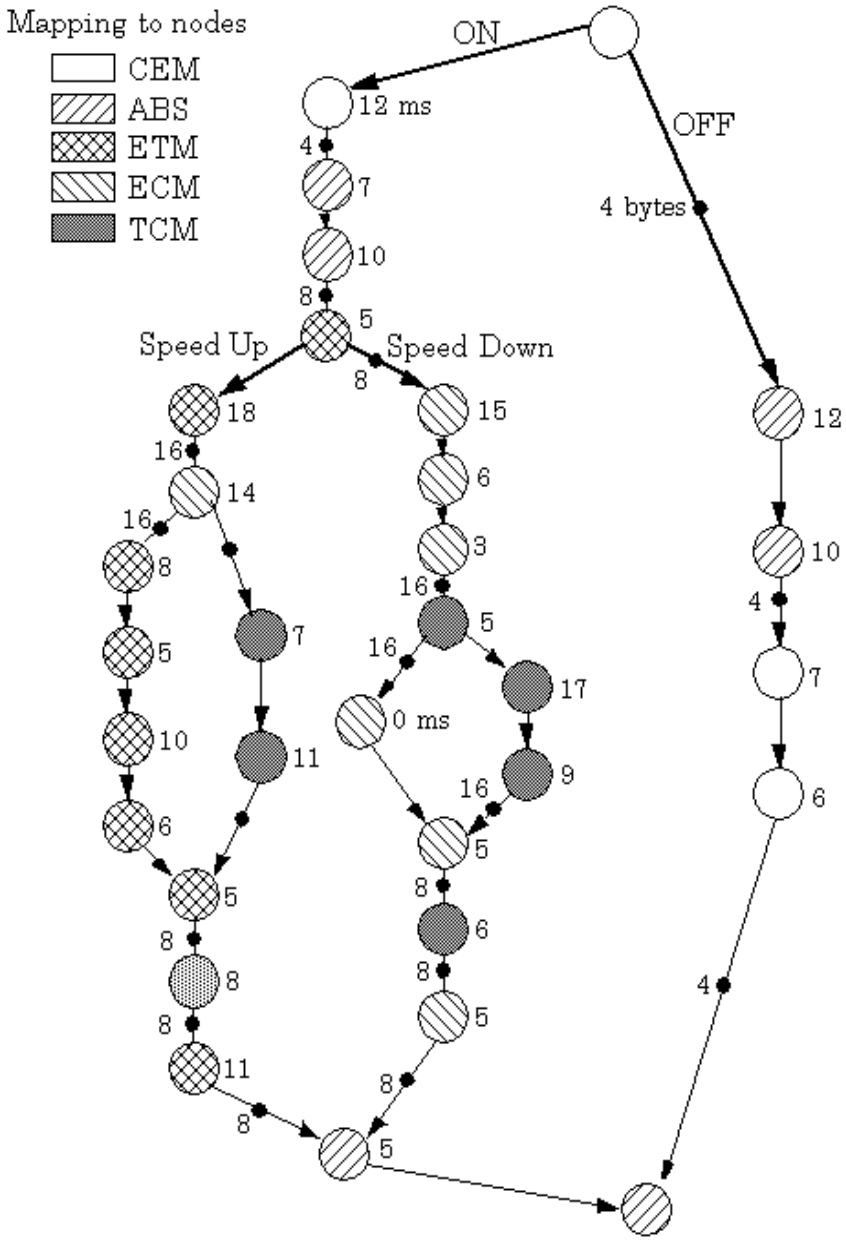


Figure 7. The cruise controller behaviour

A process can be activated only if all its inputs have arrived, and when it finishes it transmits the information to its successors. The exact semantics of a CPG are described in [7]. In this thesis, we consider a simplified CPG, where there is no alternative conditional path execution.

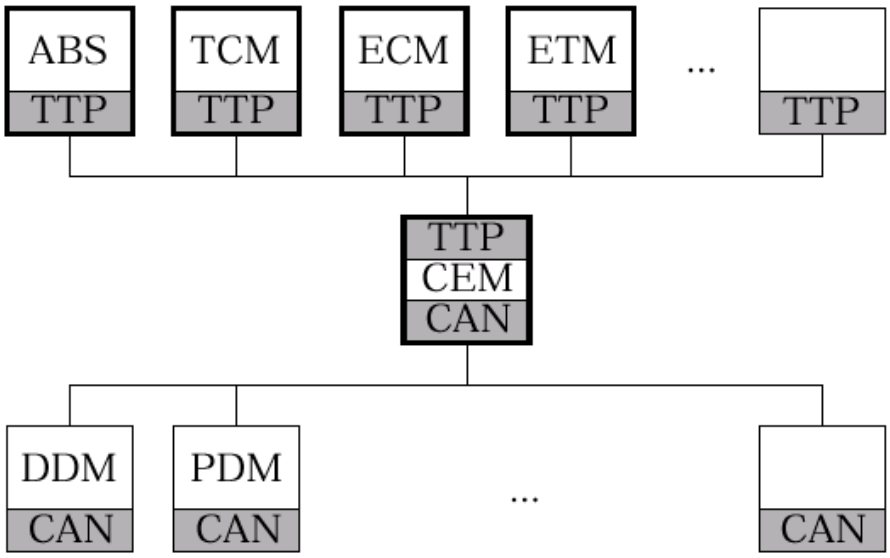


Figure 8. The cruise controller architecture

3.2 Architecture

Our specifications assume that the cruise controller operates in an environment consisting of several nodes internally connected by TTP channel (Figure 8). There are five nodes, which functionally interact with the cruise controller system: the Anti Blocking System (ABS), the Transmission Control Module (TCM), the Engine Control Module (ECM), the Electronic Throttle Module (ETM), and the Central Electronic Module (CEM). The functionality (processes) of the cruise controller is distributed over these five nodes. The transmission speed of the channel is 256kbps and the frequency of the TTP controller is chosen to be 20 MHz [1].

The responsibilities of each of the mentioned ECUs are the following [5]:

Anti Blocking System (ABS)	Produces a CAN signal representing the Vehicle speed including Quality factor.
Transmission Control Module (TCM)	Produces Can signals representing Vehicle speed & Gear Lever Position, both including Quality factors. All cars are assumed to have automatic gearbox.

Engine Control Module (ECM)	Actuates the engine torque requested by cruise control function. Produces all CAN signals related to Accelerator pedal and Brake pedal.
Electronic Throttle Module (ETM)	Actuates the Throttle angel requested by the ECM.
Central Electronic Module (CEM)	Transmits CAN signals representing the Cruise control button.

3.3 Allocation and Schedule

After the selection of the architecture components, the next stage is to decide the *allocation and mapping*: what part of the behavior should be implemented on which of the selected components. The mapping of the cruise controller is presented in Figure 7 using different hashed circles. For example, all the dark gray processes are mapped on the TCM electronic control unit.

The designer also has to decide on the execution order of the processes, which is called *scheduling*. In each ECU, there is a software module called *scheduler*, which is responsible for the activation and interruption of processes. The decisions on the activation of processes are taken based on *scheduling information*, which is produced off-line by *scheduling tools*. There are several types of scheduling information, depending on the architecture of the ECUs. Thus, for event-triggered systems, it is common to give *priorities* to processes, where a higher priority process can interrupt a lower priority process. For time-triggered systems, the scheduling information consists of *schedule tables*, which specify the start time and the duration of processes. Similar information exists for the scheduling of messages.

As mentioned earlier in section 2.2, the scheduling of the tasks in this distributed environment is a big challenge. It becomes more complicated when nodes have several functioning modes and they are dependent on the other nodes' execution. The designer has also to consider the synchronization of the different clocks needed for ensuring a deterministic scheduling.

In our case, we consider time-triggered systems, where the scheduling information consists of schedule tables. These schedule tables are produced by the scheduling tools proposed in [1].

The following table presents the schedule table for the cruise controller. Each process or communication, has a start time, a worst-case execution time, and executes on a given resource.

Process/Arc	Starting time	WCET	Resource
PR1	0	0	P6
PR2	0	12	P1
PR28	0	12	P2
ARC2	12	1	B1
PR3	13	7	P2
PR4	20	10	P2
ARC4	30	2	B1
PR5	32	5	P3
ARC18	37	2	B1
PR29	30	10	P2
ARC33	40	1	B1
PR30	41	7	P1
PR31	48	6	P1
PR17	39	15	P4
PR6	37	18	P3
ARC35	54	1	B1
ARC6	55	4	B1
PR18	54	6	P4
PR19	60	13	P4
PR7	59	14	P5
ARC7	73	4	B1
PR12	73	7	P5
ARC21	77	4	B1
PR8	77	8	P3
PR9	85	5	P3
PR13	80	11	P5
ARC14	91	2	B1
PR20	91	5	P5
PR10	90	10	P3
ARC22	100	4	B1
PR11	100	6	P3
PR14	106	5	P3
PR22	96	17	P5

ARC15	111	2	B1
PR15	113	8	P1
PR23	113	9	P5
PR21	104	20	P4
ARC26	122	4	B1
ARC16	126	2	B1
PR24	126	5	P4
ARC27	131	2	B1
PR16	128	11	P3
PR25	133	6	P5
ARC28	139	2	B1
PR26	141	5	P4
ARC29	146	2	B1
PR27	148	5	P2
PR32	153	0	P2

4 Modeling of Automotive Electronic Architectures

4.1 Introduction

Modeling of hardware and/or software architectures is done using architecture description languages (ADL). An ADL is a language that provides features for modeling hardware and software system's conceptual architecture. In other way, it is a high-level description of the overall interconnection structure of architecture [9].

ADLs provide a concrete syntax and a conceptual framework for characterizing architectures. It's underlying semantic theory is typically subsumed by the particular conceptual framework e.g., conditional process graph, Petri nets, finite state machines, etc.

ADLs provide formal modeling notations for architectural elements, such as software components and connectors. It also provides development tools that operate on architectural specifications, such as configuration and constraints. It reduces cost of development. It increases potential for commonality between different members of a closely related product family.

Many ADLs have been developed for modeling architectures both within a particular domain and as a general-purpose architecture modeling languages.

Our main interest was to address the issue of interoperability, with a focus on the exchange of automotive electronics requirements. For this purpose, instead of using an existing ADL, we have decided to define a simple architecture definition language used to model the automotive electronic systems. Our simple ADL has an XML representation that makes the models portable among manufacturers and suppliers.

4.2 Extensible Markup Language

The eXtensible Markup Language (XML) is a meta-markup language that defines a syntax used to define other domain-specific, semantic, structured markup languages.

It is a set of rules for defining semantic tags that identifies the different parts of documents [11].

XML includes the Document Type Definition (DTD) language in which tags can be defined. It shows the syntax of document by defining the elements and attributes.

Many areas like chemistry, music, math etc use XML to develop their own domain specific mark-up languages. In this way, professionals exchange notes, data and information without bothering about whether or not the receiving end person has the particular proprietary software that was used to develop the data. They can also send these data documents to the people not belonging to their profession with the confidence that the receiving people would not find any problem in reading these documents.

4.2.1 The Merit of XML

XML is a web standard, which is platform independent. This flexibility allows XML applications to exchange data contents across platforms and applications without modification. XML is available through the web browsers without licensing or royalty fees, which is also a key reason in its popularity. Open standards for XML have facilitated its quick adoption for application interoperability. Moreover, its wide availability assures to users and vendors that XML is a viable long-term technology in which it is safe and beneficial to invest time and resources thus ensuring its multi vendor support.

Another benefit of XML is the facility to create flexible data structures. Data structure for XML provides flexibility, data reuse, advanced views, Internet appliances, metadata, and intelligent agents. The data structures for XML incorporate a data exchange format that other applications can use to identify the contents through self-discovery. Meta-data allows the underlying data of XML document to stay longer as contemporary applications, so that it can be usable for future applications.

XML is very flexible because of its user definable data structure, which are flexible and extensible. It allows users to define new needed tags and attributes. It permits Data Structures to support virtually any level of simplicity or complexity of the captured data. XML's flexible data structure also encourages data reuse. It allows different subsets of the same data to be sent to multiple users in an automated way.

XML provides data validation as part of its definition through the DTD of an XML document. Due to this property, the XML documents are easy to identify and remove errors and designers are not required to incorporate the error management routines to identify invalid data. A validating parser reads a DTD and checks whether a document follows the rules specified by the DTD. If it does, the parser passes the data along to the XML application (such as web browser or a database). If parser finds mistakes it reports the error.

For the developing our XML models in this thesis, we have used a validating parser provided by Brown University's Scholarly Technology Group at http://www.stg.brown.edu/service/xmlvalid to validate the XML files we have produced [11].

4.3 Modeling of Cruise Controller using XML

We are defining four models to describe the Cruise Controller functionality. These models are Behavioral model, Architectural model, Scheduling model and Mapping model. Referring to conditional Process Graph (Figure 7), there are 32 processes from PR1 through PR32 execution on 5 Nodes or Processors namely CEM, ABS, ETM, ECM and TCM as described in Section 3.1.

The brief description of Architectural model, Behavioral model, Mapping model and Scheduling model is given below. Refer to Appendix I for details of these models.

ARCHITECTURAL MODEL

Below is the brief description of Architectural model in XML. Detailed Model is shown in Appendix I.A.

We are defining DTD for the Architecture of Cruise Controller in first stage. In DTD, we are taking components such as Sensors, Actuators, Channels and Nodes as main elements. The important information required to define for Sensors and Actuators in our model is their location in the car, which we have defined in x and y coordinates. This location is considered in meters. The names of Sensors and Actuator are given in the attributes of their respective elements.

For Channels, we are interested in their type and bandwidth capacity. We have considered two types of protocol for communication of signals on channel namely

Time Triggered Protocol (TTP) and Control Area Network Protocol (CAN). The bandwidth is defined in Megabytes. Channel name and Id are given as attributes. For Node, we are interested in its position, the name and type of processor on which it is running and the memory available for its functioning.

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE ARCHITECTURE_MODEL[
    <!ELEMENT ARCHITECTURE_MODEL (SENSOR+, ACTUATOR+,
  CHANNEL+, NODE+)>

    <!ELEMENT SENSOR (S_Position)>
    <!ATTLIST SENSOR Name CDATA #REQUIRED
       Id CDATA #REQUIRED>
    <!ELEMENT S_Position (#PCDATA)>
    <!ATTLIST S_Position axis CDATA #REQUIRED
         unit CDATA #REQUIRED>

    <!ELEMENT ACTUATOR (A_Position)>
    <!ATTLIST ACTUATOR Name CDATA #REQUIRED
              Id CDATA #REQUIRED>
    <!ELEMENT A_Position (#PCDATA)>
    <!ATTLIST A_Position axis CDATA #REQUIRED
         unit CDATA #REQUIRED>

    <!ELEMENT CHANNEL (C_Type, BW)>
    <!ATTLIST CHANNEL Name CDATA #REQUIRED
             Id CDATA #REQUIRED>
    <!ELEMENT C_Type (#PCDATA)>
    <!ELEMENT BW (#PCDATA)>
    <!ATTLIST BW unit CDATA #REQUIRED>

    <!ELEMENT NODE (N_Position, Processor+, Memory+)>
    <!ATTLIST NODE Name CDATA #REQUIRED
          Id CDATA #REQUIRED>
    <!ELEMENT N_Position (#PCDATA)>
    <!ATTLIST N_Position axis CDATA #REQUIRED
         unit CDATA #REQUIRED>
    <!ELEMENT Processor (P_Type)>
    <!ATTLIST Processor Name CDATA #REQUIRED>
    <!ELEMENT P_Type (#PCDATA)>
    <!ELEMENT Memory (#PCDATA)>
    <!ATTLIST Memory unit CDATA #REQUIRED>
]>
```

```
<!DOCTYPE ARCHITECTURE_MODEL (View Source for full doctype...)>
- <ARCHITECTURE_MODEL>
  - <SENSOR Name="PR1" Id="PR1A">
      <S_Position axis="x, y" unit="m">0, 0</S_Position>
    </SENSOR>
  - <ACTUATOR Name="PR1A" Id="PR1A">
      <A_Position axis="x, y" unit="m">0, 0</A_Position>
    </ACTUATOR>
  - <CHANNEL Name="B1" Id="B1">
      <C_Type>TTP</C_Type>
      <BW unit="kbps">256</BW>
    </CHANNEL>
  + <CHANNEL Name="B2" Id="B2">
  - <NODE Name="CEM" Id="P1">
      <N_Position axis="x, y" unit="m">1.6,
       1.1</N_Position>
    - <Processor Name="AMD">
        <P_Type>I</P_Type>
      </Processor>
      <Memory unit="KB">128</Memory>
    </NODE>

  ..........
  ..........

  + <NODE Name="TCM" Id="P5">
  </ARCHITECTURE_MODEL>
```

BEHAVIORAL MODEL

Below is the brief description of Behavioral model in XML. Detailed model is shown in Appendix I.B.

First we are defining DTD for Behavioral model of CC. We are considering two main elements i.e., Process and Arc, for the Behavioral model. There are 32 processes in total. For each process, we are interested in the worst-case execution time (WCET), its memory utilization. Sensors and actuators for signaling, if applicable, to and from the process are also considered.

There are 35 Arcs in our behavioral model. These arcs represent the flow of process from one process to the successor process. The sending process is source and the receiving one is destination. There is also an element to show the time delay (propagation delay) between source and destination for each Arc. The delay time is measured in ms.

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE BEHAVIOURAL_MODEL[
    <!ELEMENT BEHAVIOURAL_MODEL (PROCESS+, ARC+)>

    <!ELEMENT PROCESS (WCET, Memory,
            Sensor*, Actuator*)>
    <!ATTLIST PROCESS Name CDATA #REQUIRED
           Id CDATA #REQUIRED>
    <!ELEMENT WCET (#PCDATA)>
    <!ATTLIST WCET unit CDATA #REQUIRED>
    <!ELEMENT Memory (#PCDATA)>
    <!ATTLIST Memory unit CDATA #REQUIRED>
    <!ELEMENT Sensor (#PCDATA)>
    <!ELEMENT Actuator (#PCDATA)>

    <!ELEMENT ARC (Src, Dest, Delay)>
    <!ATTLIST ARC Name CDATA #REQUIRED
       Id CDATA #REQUIRED>
    <!ELEMENT Src (#PCDATA)>
    <!ELEMENT Dest (#PCDATA)>
    <!ELEMENT Delay (#PCDATA)>
    <!ATTLIST Delay unit CDATA #REQUIRED>
]>

- <BEHAVIOURAL_MODEL>
  - <PROCESS Name="PR1" Id="PR1">
      <WCET unit="ms">0</WCET>
      <Memory unit="KB">1</Memory>
      <Sensor>Driver</Sensor>
      <Sensor>Cruise Controller</Sensor>
    </PROCESS>
  - <PROCESS Name="PR2" Id="PR2">
      <WCET unit="ms">12</WCET>
      <Memory unit="KB">10</Memory>
    </PROCESS>

  .....
  .....
```

```
- <PROCESS Name="PR32" Id="PR32">
    <WCET unit="ms">0</WCET>
    <Memory unit="KB">1</Memory>
    <Actuator>Driver</Actuator>
    <Actuator>Cruise Controller</Actuator>
  </PROCESS>

- <ARC Name="ARC1" Id="ARC1">
    <Src>PR1</Src>
    <Dest>PR2</Dest>
    <Delay unit="ms">0</Delay>
  </ARC>
- <ARC Name="ARC2" Id="ARC2">
    <Src>PR2</Src>
    <Dest>PR3</Dest>
    <Delay unit="ms">1</Delay>
  </ARC>

.....
.....

- <ARC Name="ARC35" Id="ARC35">
    <Src>PR31</Src>
    <Dest>PR32</Dest>
    <Delay unit="ms">1</Delay>
  </ARC>
</BEHAVIOURAL_MODEL>
```

MAPPING MODEL

Representation of mapping model in XML is shown below. Detailed mapping model is shown in Appendix I.C.

The Mapping Model of Cruise Controller in XML has two elements i.e., process and node. There are 48 processes executing on 6 different nodes. This shows the part of the functionality (i.e., Process) to be implemented on which of the selected component (i.e., Node or Resource).

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE MAPPING [
    <!ELEMENT MAPPING (MAP+)>
    <!ELEMENT MAP (Node, Process*)>
    <!ELEMENT Node (#PCDATA)>
```

```
    <!ELEMENT Process (#PCDATA)>
]>

- <MAPPING>
  - <MAP>
      <Node>P1</Node>
      <Process>PR1</Process>
      <Process>PR2</Process>
      <Process>PR30</Process>
      <Process>PR31</Process>
    </MAP>
  - <MAP>
      <Node>P2</Node>
      <Process>PR3</Process>
      <Process>PR4</Process>
      <Process>PR27</Process>
      <Process>PR28</Process>
      <Process>PR29</Process>
      <Process>PR32</Process>
    </MAP>
  .....
  .....

  - <MAP>
      <Node>B1</Node>
      <Process>ARC2</Process>
      <Process>ARC4</Process>
      <Process>ARC6</Process>
      <Process>ARC7</Process>
      <Process>ARC14</Process>
      <Process>ARC15</Process>
      <Process>ARC16</Process>
      <Process>ARC18</Process>
      <Process>ARC21</Process>
      <Process>ARC22</Process>
      <Process>ARC26</Process>
      <Process>ARC27</Process>
      <Process>ARC29</Process>
      <Process>ARC31</Process>
      <Process>ARC33</Process>
      <Process>ARC35</Process>
    </MAP>
  </MAPPING>
```

SCHEDULING MODEL

Below is the brief description of Scheduling model in XML. Detailed Scheduling model is shown in Appendix I.D.

The Scheduling Model for Cruise Controller in XML has Slot element. These slot elements represent different processes and Arcs. There is an element of Start time for each Slot in milliseconds, when the process starts. There is another element of Duration showing the time to execute this process. The Resources element shows the machine on which the process is in execution.

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE SCHEDULE[
    <!ELEMENT SCHEDULE (SLOT+)>
    <!ELEMENT SLOT (Start, Duration, Resource)>
    <!ATTLIST SLOT Id CDATA #REQUIRED>
    <!ELEMENT Start (#PCDATA)>
    <!ATTLIST Start unit CDATA #REQUIRED>
    <!ELEMENT Duration (#PCDATA)>
    <!ATTLIST Duration unit CDATA #REQUIRED>
    <!ELEMENT Resource (#PCDATA)>
]>

- <SCHEDULE>
  - <SLOT Id="PR1">
      <Start unit="ms">0</Start>
      <Duration unit="ms">0</Duration>
      <Resource>P6</Resource>
    </SLOT>
  - <SLOT Id="PR2">
      <Start unit="ms">0</Start>
      <Duration unit="ms">12</Duration>
      <Resource>P1</Resource>
    </SLOT>
  - <SLOT Id="PR28">
      <Start unit="ms">0</Start>
      <Duration unit="ms">12</Duration>
      <Resource>P2</Resource>
    </SLOT>
  - <SLOT Id="ARC2">
      <Start unit="ms">12</Start>
      <Duration unit="ms">1</Duration>
      <Resource>B1</Resource>
    </SLOT>
  - <SLOT Id="PR3">
```

```
    <Start unit="ms">13</Start>
    <Duration unit="ms">7</Duration>
    <Resource>P2</Resource>
  </SLOT>

.....
.....

- <SLOT Id="ARC29">
    <Start unit="ms">146</Start>
    <Duration unit="ms">2</Duration>
    <Resource>B1</Resource>
  </SLOT>
- <SLOT Id="PR27">
    <Start unit="ms">148</Start>
    <Duration unit="ms">5</Duration>
    <Resource>P2</Resource>
  </SLOT>
- <SLOT Id="PR32">
    <Start unit="ms">153</Start>
    <Duration unit="ms">0</Duration>
    <Resource>P2</Resource>
  </SLOT>
</SCHEDULE>
```

5 Requirements Analysis and Modeling

5.1 Background

Requirements engineering (RE) is a process of finding the degree to which system meets the intended goals of stakeholders and documenting these in a form that is useful for analysis, communication and subsequent implementation. It is not a simple process because of the conflicting and varying interests of different stakeholder parties (including paying customer, users and developers) [12].

RE can be characterized as a branch of system engineering as it encompasses a system level view. It cannot function in isolation from the system in which it is embedded. It plays an important role in the management of changes in system development. It is performed in many contexts including market-driven product development and development for specific customer, eventually developing a broader market. It also plays an important role in calculation of feasibility of project and associated risk needed to be taken. Risk should be re-evaluated regularly throughout the development lifetime of a system. RE starts with the identification of a suitable process and the selection of methods and techniques for the various RE activities. The high level boundaries where the final delivered system will fit into the current operational environment are to find out what problem needs to be solved. Eliciting high-level goal is not only a crucial activity in early development process but also continues as development proceeds, as high-level goals (such as business goals) are refined into lower level goals (such as technical operational goals).

Modeling is the development of abstract ideas that are amenable to implementation, which is a main part of RE. There are many different types of modeling. Sometimes modeling is done on Enterprise level to capture the purpose of a system by considering the behavior of the organization in which the system has to operate. Data modeling shows what information the system will need to represent and how it corresponds to the real world phenomena being represented. Behavioral Modeling involves the modeling of dynamic or functional behavior of stakeholders and systems, both existing and required. Domain Modeling provides an abstract description of the world in which the focused system will operate.

RE also facilitates the effective and useful communication among all stakeholders for the discovered and specified requirement by ensuring that they can be read, analyzed, re-written and validated.

5.2 Types of Requirements

The following are the constraints of the automotive electronics field, as outlined in [3]. These constraints lead to requirements on the automotive electronic systems designed and developed for today's vehicles.

Costs
The design and development of a new vehicle model requires a major qualified human efforts and it takes 4 years. According to [3], 95% of the total cost of producing a new vehicle model is related to manufacturing and marketing and the only remaining 5% is related to the design and development. Therefore, the industry is always interested in technologies that will reduce manufacturing costs without losing the quality. Computer controlled multiplexing network is a good example of such technologies that replaced the copper wiring harness. We have already seeing its benefits in the Networking section above.

Reliability
An automobile is composed of a number of sophisticated subsystems, each of these having hundreds of components, which makes it complex product. We can say that today's automobiles are highly reliable products in spite of its complexity. Successful application of computer technology in the automotive industry should not decrease its reliability. Recent experience with computer controlled functions and multiplexing networks inside vehicles has shown that it is possible to achieve this goal.

Maintainability
Industries like avionics have highly skilled technicians, which follow strict maintenance rules and regulations. However, the qualification of a typical maintenance technician in an automotive industry is not very high. Thus a vehicle should be easy to maintain. The architecture of the automotive system is designed keeping in view the requirement of easy maintainability. From this point of view, architecture should be decomposable into so-called Field Replaceable Units (FRUs). Faults should be easily detectable by diagnostic tools, and it should be easy to replace the faulty FRU.

Safety
Safety is a very important issue in the automotive industry. The goal of automotive manufacturers is to reduce the number of incidents to less than one incident per billion cars per year[3].

There is a big potential to include onboard computer controlled safety mechanisms. As most of the traffic accidents are caused by human error, these mechanisms assist the human driver in critical situations and hopefully reduce the number of accidents.

The mechanical system underlying the electronic control system provides sufficient safety level to operate the car. However, the fall back to the inferior performance of the mechanical system is a safety risk. This risk increases if the driver gets used to the high performance computerized system. The implementation of fault-tolerant computer systems is also becoming cheaper than the corresponding combined computer/mechanical system. Therefore, the redundant mechanical system will be replaced by a fault-tolerant computerized system in the near future.

Security
The security of an automotive system is considered as an integral part of its architecture. The increasing number of vehicle thefts forced automotive manufacturers to introduce theft avoidance systems. These systems includes from very simple schemes to cryptographic based systems. The other important issue is the integrity checks of the control system installed onboard vehicles. Security measures have to be taken to prevent this type of incidents, as unauthorized modifications of control operations are possible by replacing the parts of the system (e.g., the ROMs or EPROMs).

Comfort
To increase the degree of comfort of a vehicle, additional items such as sophisticated audio/video systems, telephone systems, and other electronic comfort functions can be installed. The increasing number of vehicle users, which are willing to pay for increased comfort has prompted the automotive companies to offer increased comfort facilities as extra items.

Legal constraints
Automobiles have a profound influence on our society. Environmental concerns including safety matters forced legislators to enforce rules and regulations on the automotive industry. Use of electronic control functions is virtually mandatory on

every vehicle because of the strict requirements of many of these regulations (e.g., California's OBDII, Europe's proposed EOBD and EURO OBDII).

5.2.1 Requirements on the Cruise Controller

We have defined four XML files for modeling of cruise controller in section 4.3. Now we will discuss the requirements needed to analyze cruise controller model. We will define different rules to check if the cruise controller models fulfill these rules. Each rule will be based on certain requirements.

Schedule

1. The cruise controller should execute within 100 ms.
   (This means that the last process in the schedule should finish by 100 ms)
2. There should be no overlap between two processes on the same processor, or between two messages on the same bus.
3. The precedence order should be respected by the schedule.
   (If PR1 is the predecessor of PR2, it should not execute after it)
4. The duration of execution on a processor should be the same as the one in the specification.

Mapping

5. The sum of the memory of all processes mapped on a node should not exceed the total memory of that node (processor).
6. No process should be mapped on more than one node.
7. Every process should be mapped on one node.
8. A process should be mapped only on the potential nodes.

Application

9. Every process except the first one should have a predecessor.
10. Every process except the last one should have a successor.

If the Cruise Controller model defined in XML follows above rules and gives the correct results then in this case, our CC model is working properly. If any of the condition does not meet and gives error then it means that the CC model defines has some errors. These errors should be checked and removed immediately.

5.3 Capturing Requirements using XML Technologies.

Discussion about existing XML technologies, with an explanation about why the following ones were chosen.

5.4 XLinkit

Xlinkit is a rule-based hyperlink generation and consistence checking language. It considers as an input XML documents sets and gives as output Xlink linkbases that have semantically relevant links between elements of those XML documents [14].

It is important for related information to be linked in order to support navigation for an effective hypertext. The construction of links manually is difficult, time consuming and prone to errors. It is required to automate the process of link generation. Thanks to Xlinkit that generates automated hyperlink, which are stored in linkbase, for a large data volumes with global consistency of distributed documents. Xlinkit uses full benefits of the semantic richness of distributed XML documents by enabling the specifications of rules that relate the document types.

A number of powerful languages for linking and elements extraction have been developed until now. XPath provides the formulation of queries on XML documents, which results a set of elements. Xlink provides linking functionality for XML data. Unlike HTML, it can form links between more than two endpoints, which means any element in the XML file can act as a link. XLinks do not have to be stored in the documents that have to be linked i.e., we can assemble a set of links in the linkbase. When combining with XPointer or XPath, in addition to document level, Xlink is able to link between elements contained within the documents.

XPath allows us to retrieve elements from XML documents and group them into sets of DOM nodes. For example, the XPath query /schedule/SCHEDULE/ SLOT/WCET/text() retrieve the set of all slot WCETs. The function $S[[p]]_x$ select the pattern p from the document with context node x. The pattern is always relative to the context node. If a specific node has to be retrieved with an absolute index, the root node / can be used as the context node.

Our requirement set in previous section says that the duration of execution on a processor should be the same as the one in the specification. This means for all

slots in the Schedule, there must be some process or arc with the same Id in behavioral model. So first of all we will establish the sets, which we are dealing. The set of all WCETs in Schedule can be retrieved by applying the query *S*[[/schedule/SCHEDULE/SLOT/WCET]] to the Schedule model presented in Appendix I.D. The set of all Durations/Delays in Behavior for process or arc can be retrieved by using the query *S*[[/behavior/BEHAVIOURAL_MODEL/PROCESS/ Duration]] or *S*[[/behavior/BEHAVIOURAL_MODEL/ARC/ Delay]] to the Behavior model presented in Appendix I.B.

We can translate above natural language rules into our language. Let 'p' be the set of all Slots in the Schedule model. Let 'a' and 'c' be the set of all Processes and Arcs respectively in the Behavioral model. We can write as:

$$\forall p \in P(\exists a \in A\ (S[[@Id]]_p = S[[@Id]]_a))$$
$$\forall p \in P(\exists c \in C\ (S[[@Id]]_p = S[[@Id]]_c))$$

Currently the language is very simple. It corresponds to the constructs allowed in general first order logic with the following limitations:

We are working on DOM nodes, which are always finite, the only predicates allowed are equality and inequality and no functions are allowed. Under these conditions, a truth table for a particular rule with its sets of nodes can be established in finite time. Thus our evaluation function will always terminate.

The rules encoded in XML provide users the same editing tool environment that they would be using for processing their data documents. The formulation of our query in XML is as follows

```
<consistencyruleset>
  <globalset id="schedule" xpath="/SCHEDULE/SLOT"/>
  <globalset id=
         "process_behaviour" xpath="/BEHAVIOURAL_MODEL/PROCESS"/>
  <globalset id="arc_behaviour" xpath="/BEHAVIOURAL_MODEL/ARC"/>

<consistencyrule id="r1">
  <description>
    The WCET in Schedule model must be equal to corresponding
Duration/Delay time in Behavioral model.
  </description>

<forall var="a" in="$schedule">
```

```
  <or>
      <exists var="p" in="$process_behaviour">
       <and>
          <equal op1="$a/@Id" op2="$p/@Id"/>
          <equal op1="$a/Duration/text()" op2="$p/WCET/text()"/>
        </and>
      </exists>

      <exists var="c" in="$arc_behaviour">
       <and>
          <equal op1="$a/@Id" op2="$c/@Id"/>
          <equal op1="$a/Duration/text()" op2="$c/Delay/text()"/>
        </and>

      </exists>
  </or>
</forall>
</consistencyrule>
</consistencyruleset>
```

Please refer to Appendix II.A for the complete rule file in XML.

5.4.1 Link Generation

There are two types of links: consistent and inconsistent links. If the relationship set out in the rule holds, the link established between the elements is consistent otherwise the link generated is inconsistent.

A rule consists of a formula, which has an equality comparison or a qualifier or operator that may consist of further formulae. We achieve set of links by evaluating a rule by defining a recursive strategy. The locators contained in a link will be derived from the current assignment of qualifier variables in the formula when the link is being composed. A formula that recursively consist of subformula takes the links created by the subformulae and appends its own locators and links.

TABLE: SOURCE AND DESTINATION NODE SETS

Schedule set (P)	Behavior set (A or C)
SLOT (Id=”PR1”)	PROCESS(Id=”PR1”)
SLOT (Id=”PR2”)	PROCESS(Id=”PR2”)
SLOT (Id=”PR28”)	PROCESS(Id=”PR3”)

SLOT (Id=”ARC2”)	PROCESS(Id=”PR4”)
SLOT (Id=”PR3”)	PROCESS(Id=”PR5”)
SLOT (Id=”PR4”)	PROCESS(Id=”PR6”)
	……
	ARC(Id=”ARC1”)
	ARC(Id=”ARC2”)
	ARC(Id=”ARC3”)
	ARC(Id=”ARC4”)

Going through our rules:

$$\forall p \in P(\exists a \in A\ (\mathcal{S}[[@Id]]_p = \mathcal{S}[[@Id]]_a))$$
$$\forall p \in P(\exists c \in C\ (\mathcal{S}[[@Id]]_p = \mathcal{S}[[@Id]]_c))$$

The forall evaluation binds p to the first entry in the set, P_1 (the Slot with Id “PR1” and pass it to the existential qualifier evaluation, which iterates through sets A and C and perform an equality comparison for each entry. For example consider the case of Slot at P_4 (the slot with Id “ARC2”), the behavior at C_2 is true, so we use C_2 as a locator and use a link of the form consistent [C_2]. The result is the set of links

{consistent[P_4,C_2]}

If there is a slot with Id (say x), which is not defined in behavior, the existential qualifier will return false and produce an empty set of links. The universal qualifier uses this information to store the link inconsistent[P_x].

5.4.2 Document Management

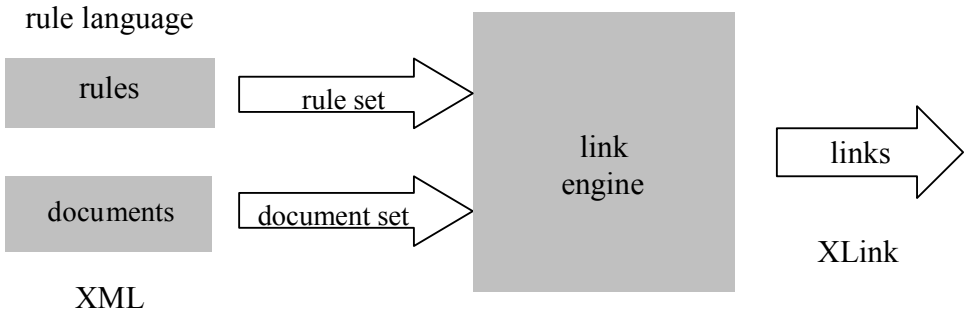


Figure 9. Rule engine architecture

The link generation engine always takes a set of documents and a set of rules as input shown in Figure 9.

The document set file, which links the two data files in our query, is shown below. The Document elements instruct the system to include an XML document into the Set.

```
<DocumentSet name="BehaviorDoc">

<Description>
         Doc related to the Cruise Controller environment
</Description>

<DocFile href="http://www.ida.liu.se/~x01syeak/
xlinkit/schedule.xml"/>

<DocFile href="http://www.ida.liu.se/~x01syeak/
xlinkit/Behaviors.xml"/>

</DocumentSet>
```

The rules are included similarly and arranged in a rule set as shown in the next listing. The Rule file element is used to add rule from a file. Using the xpath attribute to pick up specific rule can do further precision. A Set element can be used in a rule set to include further rule sets, similar to the document set mechanism.

```
<RuleSet name="Cruise Controller">
  <Description>
         Rules related to the CC environment
  </Description>
  <RuleFile
         href="http://www.ida.liu.se/~x01syeak/xlinkit/rule4.xml"
         xpath="/consistencyruleset/consistencyrule" />
</RuleSet>
```

Refer to Appendix II.B for complete details of Document set file and Rule Set file.

5.4.3 Applications

Xlinkit has a various applications in enterprise data integration and content management. It is very useful to create lightweight portals across heterogeneous

data resources, for business rules. Content aggregation services, which bring together XML feeds, can use xlinkit to provide value-added linking wholly automatically [14].

It can also be useful in other areas, which have large volumes of distributed and potentially inconsistent data.

We have seen that in our case xlinkit is only applicable to the rule no. 4 of section 5.2.1. For the rest of the rules we do not need to check consistency among different databases so we will not use xlinkit.

5.5 XQuery

Previously XML database designers were focusing on developing efficient storage methods for XML data, but nowadays the focus is shifting towards creating powerful query and retrieval methods. There are many query languages (including XQL, XML-QL, Quilt) for XML data has been developed. XML queries are performed in two ways. Some are applied on data that is physically represented as XML, which may be either persistent or transient data. Others are applied on XML views of foreign data sources. Because query languages have traditionally been designed for specific kinds of data, most existing proposals for XML query languages are robust for particular types of data sources but weak for other types.

XQuery is a query language that uses the structure of XML intelligently to express queries across all these kinds of data. Therefore it is designed to be broadly applicable across all types of XML data sources. It is the first query language to receive industry-wide attention and support. It is currently being developed by the W3C XML Query Working Group [16].

Industry experts expect applications of XQuery for XML and XML databases similar to applications of SQL for relational data and relational database systems. XQuery provides a vendor independent, powerful and easy-to-use method for query and retrieval of XML data. XQuery is designed to be a very simple and easily implementable language. The queries used in Xquery are concise and easily understood. The dominating feature of Xquery is that it is flexible enough to query a broad spectrum of XML information sources, including both databases and documents. The Query Working Group has identified a requirement for both a human-readable query syntax and an XML-based query syntax. XQuery is designed to meet the first of these requirements. XQuery is derived from an XML

query language called Quilt. Quilt which in turn borrowed features from several other languages, including XPath, XQL, XML-QL, SQL, and OQL.

XQuery uses the data model, which is based on XPath. It considers each XML document as a tree of nodes. The data model, in addition to handling documents, is also designed to work on well-formed documents parts known as fragments, collection of documents or collection of fragments.

XQuery is a functional language where each query is an expression. There are 7 types of expressions in XQuery: path expressions, element constructors, FLWR expressions, expressions involving operators and functions, conditional expressions, quantified expressions and expressions that test or modify data types. The various expressions can be used together both sequentially and nested [15]. We will see few expression in the following subsection which are more important.

5.5.1 Language Concepts

XQuery Path Expressions

A path expression locates nodes within a tree, and returns a sequence of distinct nodes in document order. A path expression is always evaluated with respect to an evaluation context.

In our requirement 1 of section 5.2.1, we are referring to worst case execution time of process Name “PR1” in the document named "behvavior.xml" as shown below.

```
document("behavior.xml")//BEHAVIOURAL_MODEL
          /PROCESS[@Name = "PR1"]/WCET
```

XQuery Element Constructors

XQuery element constructor consists of a start tag and an end tag, enclosing an optional list of expressions that provide the content of the element.

The following expression creates a Process element that contains attributes, subelements, and text:

```
<Process>
          <duration > $w </duration>,
          <memory> $m </memory>
```

```
</Process>
```

where we will initialize our query by defining variables $w and $m are variables defines for WCET and Memory tuples respectively in the file behavioral.xml.

XQuery FLWR Expressions

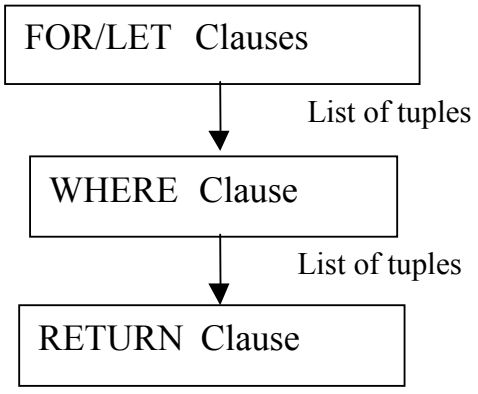
The name "FLWR", pronounced "flower", stands for the keywords for, let, where, and return clauses. XQuery provides a FLWR expression for iteration and for binding variables to intermediate results (see Figure 10). This kind of expression is often useful for computing joins between two or more documents and for restructuring data.

Example: find all Processes having WCET greater than 10

```
FOR $x IN document("behavior.xml")//Process
WHERE $x/WCET > 10
RETURN $x/@Name
```

FOR and LET clauses generate a list of tuples of bound expressions, keeping the document order. WHERE clause applies a predicate, eliminating some of the tuples. RETURN clause is executed for each surviving tuple, which generates an ordered list of outputs.

Although for and let both bind variables, the manner in which variables are bound is quite different. In a let clause, the variable is bound directly to the expression,



Instance of XQuery data model

Figure 10. The ”Flower” expression.

and it is bound to the expression as a whole. Consider the following query:

```
let $x := document("behavior.xml")//Process
return <result> $x </result>
```

The variable $x is bound to the expression (<Process/>). There is no ‘for’ clauses, so the ‘let’ clause generates one tuple that contains the variable binding of $x. The return clause is invoked for this tuple, creating the following output:

```
<result>
  <Process>...</Process>
  <Process>...</Process>
  <Process>...</Process>
  ...
</result>
```

Now we are considering a similar query which contains a ‘for’ clause instead of a ‘let’ clause:

```
for $x IN document("behavior.xml")//Process
return <result> $x </result>
```

The variable $x is associated with the expression (<Process/>), from which the variable bindings of $x will be drawn. When only one expression is present, the Cartesian product is equivalent to the sequence of values returned by that expression. In this example, the variable $x is bound as many times as many are the processes in Behavior file; One tuple is generated for each of these variable bindings, and the return clause is invoked for each tuple, creating the following output:

```
<result> <Process>...</Process></result>
<result> <Process>...</Process></result>
<result> <Proces >...</Process></result>
```

Conditional Expressions

XQuery supports a conditional expression based on the keywords if, then, and else. The expression following the ‘if’ keyword is called the test expression, and the expressions following the ‘then’ and ‘else’ keywords are called result expressions.

The first step in processing a conditional expression is to find the effective boolean value of the test expression. The value of a conditional expression is defined as follows: If the effective boolean value of the test expression is true, the value of the first ("then") result expression is returned. If the effective boolean value of the test expression is false, the value of the second ("else") result expression is returned.

```
For $h in document("behavior.xml")//Process
Return        <PROCESS>
                        $h/WCET,
                        If ($h/WCET < 8)
                        THEN "ok"
                        ELSE "Time Lapse"
              </PROCESS> SORTBY (Process)
```

5.5.2 XQuery Engines

QuiP

For the purpose of this thesis, we are using Software AG's Quip interface to execute queries for Cruise Control Model. Quip is a prototype of XQuery, the W3C XML Query Language. It is a Graphical user interface for writing queries and viewing results. It is easy to learn and use the language. Online help including syntax diagrams for XQuery and a large number of examples including W3C usecase queries are available on web site. It is supported by Linux and windows operating systems.

5.5.3 Writing Requirements using XQuery

Now we will use XQuery to implement rules defined in section 5.2.1. The first rule requires that "The cruise controller should execute within 100 ms. It means that the last process in the schedule should finish by 100 ms. This rule can be written using XQuery in the following way:

```
<results>{
  let $a := document("data/schedule.xml")//SLOT,
       $b := max(for $c in
                 document("data/schedule.xml")//SLOT/Start
                 return
                  int(string-value($c))),
      $d := $a[Start = $b],
      $e := $a[Start = $b]/Start,
      $f := $a[Start = $b]/Duration,
```

```
        $g := $b+$f

        return
            <SLOT Id={$d/@Id}>{
            $e, $f}
            {<WCET> {$g} </WCET>
            ,if ($g>100)
            then
            <Fault> Time Lapsed </Fault>
            else "WCET ok"
            }</SLOT>
}</results>
```

It is clear from the XQuery syntax shown above that it is similar to SQL. This shows its simplicity and ease in use.

The output of the previous query, if applied to the cruise controller model, is:

```
<results>
 <SLOT Id="PR32">
  <Start unit="ms">153</Start>
  <Duration unit="ms">0</Duration>
  <WCET>153</WCET>
  <Fault>Time Lapsed</Fault>
  </SLOT>
</results>
```

Since the last process in the Schedule is unable to finish within 100ms, therefore fault indication is given as “Time Lapsed”.

Similarly we applied the XQuery to find out the result required for query no. 5 of section 5.5.1, which says, “The sum of the memory of all processes mapped on a node should not exceed the total memory of that node (processor)”.

Our query will be of the following form:

```
for
  $map in document("data/sweb/mapping.xml")//MAP,
  $node in document("data/sweb/architecture.xml")//:
    NODE[@Id = $map/@Resource]
         let $proc := document("data/sweb/behaviour.xml")//
         PROCESS[@Id = $map/Process]
```

```
return
<processor
  Name={$node/@Name}
  Id={$node/@Id}
  HasMemory={$node/Memory/text(),$node/Memory/@unit}
  MemoryUsedByScheduledProcesses=
 {sum($proc/Memory),$node/Memory/@unit}
>
{
 for $process in $proc
 return
   <process
     Name={$process/@Name}
     Id={$process/@Id}
     Memory={$process/Memory/text(),
       $process/Memory/@unit}
   />
 sortby(int(substring-before(@Memory,"K")))
}
</processor>
sortby(int(substring-after(@Id,"P")))
```

The output of the previous query, if applied to the cruise controller model, is the following:

```
<?xml version="1.0"?>
<quip:result xmlns:quip=
      "http://namespaces.softwareag.com/tamino/quip/">
   <processor Name="CEM" Id="P1" HasMemory="128KB"
      MemoryUsedByScheduledProcesses="20KB">
      <process Name="PR1" Id="PR1" Memory="1KB"/>
      <process Name="PR31" Id="PR31" Memory="4KB"/>
      <process Name="PR30" Id="PR30" Memory="5KB"/>
      <process Name="PR2" Id="PR2" Memory="10KB"/>
   </processor>
   <processor Name="ABS" Id="P2" HasMemory="256KB"
      MemoryUsedByScheduledProcesses="25KB">
      <process Name="PR32" Id="PR32" Memory="1KB"/>
      <process Name="PR27" Id="PR27" Memory="2KB"/>
      <process Name="PR3" Id="PR3" Memory="2KB"/>
      <process Name="PR4" Id="PR4" Memory="5KB"/>
      <process Name="PR29" Id="PR29" Memory="7KB"/>
      <process Name="PR28" Id="PR28" Memory="8KB"/>
   </processor>
   <processor Name="ETM" Id="P3" HasMemory="128KB"
```

```
         MemoryUsedByScheduledProcesses="40KB">
         <process Name="PR14" Id="PR14" Memory="2KB"/>
         <process Name="PR11" Id="PR11" Memory="2KB"/>
         <process Name="PR9" Id="PR9" Memory="2KB"/>
         <process Name="PR8" Id="PR8" Memory="3KB"/>
         <process Name="PR10" Id="PR10" Memory="4KB"/>
         <process Name="PR16" Id="PR16" Memory="7KB"/>
         <process Name="PR6" Id="PR6" Memory="10KB"/>
         <process Name="PR5" Id="PR5" Memory="10KB"/>
      </processor>
      <processor Name="ECM" Id="P4" HasMemory="256KB"
         MemoryUsedByScheduledProcesses="19KB">
         <process Name="PR26" Id="PR26" Memory="2KB"/>
         <process Name="PR24" Id="PR24" Memory="2KB"/>
         <process Name="PR21" Id="PR21" Memory="2KB"/>
         <process Name="PR19" Id="PR19" Memory="2KB"/>
         <process Name="PR18" Id="PR18" Memory="3KB"/>
         <process Name="PR17" Id="PR17" Memory="8KB"/>
      </processor>
      <processor Name="TCM" Id="P5" HasMemory="128KB"
         MemoryUsedByScheduledProcesses="30KB">
         <process Name="PR22" Id="PR22" Memory="1KB"/>
         <process Name="PR20" Id="PR20" Memory="2KB"/>
         <process Name="PR25" Id="PR25" Memory="3KB"/>
         <process Name="PR13" Id="PR13" Memory="5KB"/>
         <process Name="PR12" Id="PR12" Memory="5KB"/>
         <process Name="PR23" Id="PR23" Memory="7KB"/>
         <process Name="PR7" Id="PR7" Memory="7KB"/>
      </processor>
   </quip:result>
```

5.6 CommonRules

CommonRules [18] is an intelligent rule-based system, which focuses on the flexible and consistent connectivity between the logic (business rules) with the underlying data (business objects) through the Java features. CommonRules provides efficient XML interoperability and prioritized conflict handling capabilities.

CommonRules makes use of specific form of logic system, call Situated Courteous Logic Program (CLP) which provides a set of semantically rich, human like language with a flexible and powerful mapping system to allow linking of any external attached procedures (Java classes, functions, databases, legacy data

through wrapper) and object instances to the rule set. Thus allowing the ruleset, which expresses the business logic to execute directly through a processing engine.

This approach of application development replaces the traditional cycle of software development and thus narrowing down the role of software developer to simply build the business objects (such as database access, legacy data conversion...etc). While the business executives with the help of a rule technician can specify the business requirement in the form of business rules. Testing of business logic and data can be done separately. Business logic can be changed or modified without changing the underlying business objects. Applications development using this approach, are easy to implement, maintain and modify. It also allows maximum portability since the business logic does not depend on the underlying data.

It is so simple to implement CommonRules that business management people, mostly non-programmer, can easily amend the executable business rules incrementally at run-time. For example if a company policy is to give some percent discount on purchasing goods to loyal customers as an incentives. Now another law comes in addition that late payment customers will not get discount. Let's assume these two rules, which work separately conflicts for a particular customer who is loyal but also late in payments. The CommonRule detects the conflict automatically and will raise the alarm for this customer. A sales manager can easily resolve the conflict by incorporating at run-time a further rule specifying that the first rule has priority over the second. The new rule set will now automatically cover all customers with similar conditions.

Below is a simple example of how to use ComonRules:

```
/*  bookstore pricing ruleset:

A. "5 percent discount if have a loyal spending history"

B. "10 percent discount if have a big spending history"

C. "5 percent discount if have store charge card"

D. "10 percent discount if member of the Platinum Club"

E. "lose your discount if you have late-payment history in the
last year"

F. otherwise, no discount
```

```
priorities:
B is higher than all the others;
D is higher than A, C;
E is higher than A, C, D;
B is higher than E;
F is lower than all the others.

*/

<steadySpender>
  if
    shopper(?Cust) and  spendingHistory(?Cust, loyal)
  then
    giveDiscount(percent5, ?Cust);

<bigSpender>
   if
      shopper(?Cust) and spendingHistory(?Cust, big)
   then
      giveDiscount(percent10, ?Cust);

<storeCard>
   if
      shopper(?Cust) and hasChargeCard(?Cust, store)
   then
      giveDiscount(percent5, ?Cust);

<platinumClub>
   if
      shopper(?Cust) and  memberPlatinumClub(?Cust)
   then
      giveDiscount(percent10, ?Cust);

<slowPayer>
   if
      slowToPay(?Cust, last1year)
   then
      giveDiscount(percent0, ?Cust);

<presumeOrdinary>
   if
      shopper(?Cust)
```

```
   then
      giveDiscount(percent0, ?Cust);

<emptyLabel>
   if
      inModule(?X, discounting) and notEquals(?X, bigSpender)
   then
      overrides(bigSpender,?X);

<emptyLabel>
   if
      inModule(?X, discounting) and notEquals(?X, presumeOrdinary)
   then
      overrides(?X,presumeOrdinary);

<emptyLabel>
   overrides(slowPayer, steadySpender);
<emptyLabel>
   overrides(slowPayer, storeCard);
<emptyLabel>
   overrides(slowPayer, platinumClub);
<emptyLabel>
   overrides(platinumClub, steadySpender);
<emptyLabel>
   overrides(platinumClub, storeCard);

<emptyLabel>
   inModule(steadySpender, discounting);
<emptyLabel>
   inModule(bigSpender, discounting);
<emptyLabel>
   inModule(storeCard, discounting);
<emptyLabel>
   inModule(storeCard, memberships);
<emptyLabel>
   inModule(platinumClub, discounting);
<emptyLabel>
   inModule(platinumClub, memberships);
<emptyLabel>
   inModule(slowPayer, discounting);
<emptyLabel>
   inModule(slowPayer, payments);
<emptyLabel>
   inModule(presumeOrdinary, discounting);

MUTEX
```

```
    giveDiscount(?X, ?Cust) and giveDiscount(?Y, ?Cust)
GIVEN
    notEquals(?X, ?Y);


/* some facts for various cases of shoppers  */

shopper(ann);

shopper(cal);
spendingHistory(cal, loyal);

shopper(meg);
spendingHistory(meg, loyal);
spendingHistory(meg, big);

shopper(tim);
spendingHistory(tim, loyal);
memberPlatinumClub(tim);

shopper(zoe);
hasChargeCard(zoe, store);
slowToPay(zoe, last1year);
spendingHistory(zoe, big);
```

6 Conclusions and Future Work

In this thesis we have addressed the problems of interoperability and modeling of the requirements in the automotive electronics field.

We have used the XML language to define a simple ADL language. The language captures the behavior, architecture, mapping and scheduling information of automotive systems. Such a representation makes the models portable among the manufacturers and suppliers in the automotive field. The ADL has been successfully used to model a vehicle cruise controller.

Starting from the constraints of the automotive electronics field, we have identified several types of requirements. We have then tried to match the semantics of the types of requirements identified with existing XML technologies.

Thus, xlinkit has been used to model requirements that can be easily expressed as set operations, XQuery for those that can be captured using relational-database constructs, while CommonRules is suited for requirements that can be expressed using a logic-programming language. Several requirements placed on the cruise controller have been successfully modeled using the mentioned XML technologies.

The advantage of this approach is that it formalizes the requirements capturing process, hopefully leading to formal verification and the elimination of errors. More importantly for our thesis, it improves the interoperability of tools used by the automotive electronic manufacturers and suppliers by shipping the models at the same time with the requirements using standard XML technologies.

Our approaches have been validated through an industrial case study representing a vehicle cruise controller. The cruise controller has been modeled using our ADL, and the requirements placed on the cruise controller have been modeled using xlinkit, XQuery and CommonRules.

Ideas for future work include:

- An analysis of the relationship between this work and the semantic web. For example, what are the advantages of using an ontology language as the basis for an ADL.

- What are the challenges of developing a more complex and useful ADL for automotive electronics?
- An analysis of how difficult actually is to express requirements using our approach for more complex systems. How easy is for the designers to use this approach?
- What are the changes if the soon-to-be-a-standard RuleML is used instead of CommonRules?

References

1. P. Pop, “Scheduling and Communication Synthesis for Distributed Real-Time Systems”, Licentiate Thesis No. 832, Linköping Studies in Science and Technology, Linköping, Sweden, 2000.

2. H. Kopetz, “Real-Time Systems-Design Principles for Distributed Embedded Applications”, Kluwer Academic Publishers, 1998.

3. H. Kopetz, “Automotive Electronics: Present State and Future Prospects”, Technical University of Vienna, 1995.

4. J. Axelsson, “Analysis and Synthesis of Heterogeneous Real-Time Systems”, Dissertation No. 502, Linköping Studies in Science and Technology, Linköping, Sweden, 1997.

5. A. Lindbom, “Functional Requirement Description- Functional Area: Engine, Sub-function Cruise Control”, Issue P1, 1998.

6. G. Leen and D. Hefffernan, “In-Vehicle Networks: Expanding Automotive Electronic Systems”, PEI Technologies and University of Limerick, IEEE paper.

7. P. Eles, A. Doboli, P. Pop, and Z. Peng, Scheduling with Bus Access Optimization for Distributed Embedded Systems, IEEE Transactions on VLSI Systems, Volume: 8, Issue: 5, October 2000, Pages 472-491.

8. OSEK/VDX Operating System Specification, Version 2.2, www.osek-vdx.org

9. N. Medvidovic and R. N. Taylor, “A Framework for Classifying and Comparing Architecture Description Language”, Department of information and Computer Science, University of California, Irvine, USA.

10. W. O. Cesario, L. Gauthier, D. Lyonnard, G. Nicolescu and A. A. Jerraya, “An XML-based Meta-model for the Design of Multiprocessor Embedded Systems”, TIMA Laboratory, Grenoble, France

11. E. R. Harold, “XML Bible”, 1999.

12. B. Nuseibeh and S. Easterbrook, “Requirement Engineering: A Roadmap”, Department of Computing, Imperial College, London, UK.

13. “Web Services Made Easier- The Java APIs and Architecture for XML”, A Technical White Paper, Sun Microsoft Inc., USA.

14. C. Nentwick, W. Emmerich and A. Finkelstein, “xlinkit: links that make sense”, Department of Computer Science, University College London, London WCIE 6BT.

15. “XQuery 1.0: An XML Query Language- W3C Working Draft 30 April 2002”.

16. B. Bakker and I. Widarto, “An Introduction to Xquery”.

17. H. Boley, “The Rule Markup Language: RDF-XML Data Model, XML Schema Hierarchy, and XSL Transformations”, DFKI GmbH.

18. Common Rules, http://www.alphaworks.ibm.com/tech/commonrules

Appendix I

A. Architectural Model

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE ARCHITECTURE_MODEL[
	<!ELEMENT ARCHITECTURE_MODEL (SENSOR+, ACTUATOR+,
                                              CHANNEL+, NODE+)>

	<!ELEMENT SENSOR (S_Position)>
	<!ATTLIST SENSOR Name CDATA #REQUIRED
                                                        Id CDATA #REQUIRED>
	<!ELEMENT S_Position (#PCDATA)>
	<!ATTLIST S_Position axis CDATA #REQUIRED
                                                        unit CDATA #REQUIRED>

	<!ELEMENT ACTUATOR (A_Position)>
	<!ATTLIST ACTUATOR Name CDATA #REQUIRED
                                                        Id CDATA #REQUIRED>
	<!ELEMENT A_Position (#PCDATA)>
	<!ATTLIST A_Position axis CDATA #REQUIRED
                                                        unit CDATA #REQUIRED>

	<!ELEMENT CHANNEL (C_Type, BW)>
	<!ATTLIST CHANNEL Name CDATA #REQUIRED
                                                        Id CDATA #REQUIRED>
	<!ELEMENT C_Type (#PCDATA)>
	<!ELEMENT BW (#PCDATA)>
	<!ATTLIST BW unit CDATA #REQUIRED>

	<!ELEMENT NODE (N_Position, Processor+, Memory+)>
	<!ATTLIST NODE Name CDATA #REQUIRED
                                                        Id CDATA #REQUIRED>
	<!ELEMENT N_Position (#PCDATA)>
	<!ATTLIST N_Position axis CDATA #REQUIRED
                                                        unit CDATA #REQUIRED>
	<!ELEMENT Processor (P_Type)>
	<!ATTLIST Processor Name CDATA #REQUIRED>
	<!ELEMENT P_Type (#PCDATA)>
	<!ELEMENT Memory (#PCDATA)>
	<!ATTLIST Memory unit CDATA #REQUIRED>
]>

<ARCHITECTURE_MODEL>

	<SENSOR Name="PR1" Id="PR1A">
		<S_Position axis="x, y" unit="m">0, 0 </S_Position>
	</SENSOR>

	<ACTUATOR Name="PR1A" Id="PR1A">
		<A_Position axis="x, y" unit="m">0, 0 </A_Position>
	</ACTUATOR>
```

```
        <CHANNEL Name="B1" Id="B1">
               <C_Type>TTP</C_Type>
               <BW unit="kbps">256</BW>
        </CHANNEL>

        <CHANNEL Name="B2" Id="B2">
               <C_Type>CAN</C_Type>
               <BW unit="kbps">512</BW>
        </CHANNEL>

        <NODE Name="CEM" Id="P1">
               <N_Position axis="x, y" unit="m">1.6, 1.1  </N_Position>
               <Processor Name="AMD">
                      <P_Type>I</P_Type>
               </Processor>
               <Memory unit="KB">128</Memory>
        </NODE>

        <NODE Name="ABS" Id="P2">
               <N_Position axis="x, y" unit="m">0.4, 1.4 </N_Position>
               <Processor Name="Pentium">
                      <P_Type>III</P_Type>
               </Processor>
               <Memory unit="KB">256</Memory>
        </NODE>

        <NODE Name="ETM" Id="P3">
               <N_Position axis="x, y" unit="m">0.8, 1.4 </N_Position>
               <Processor Name="AMD">
                      <P_Type>II</P_Type>
               </Processor>
               <Memory unit="KB">128</Memory
        </NODE>

        <NODE Name="ECM" Id="P4">
               <N_Position axis="x, y" unit="m">1.2, 0.8 </N_Position>
               <Processor Name="Pentium">
                      <P_Type>III</P_Type>
               </Processor>
               <Memory unit="KB">256</Memory
        </NODE>

        <NODE Name="TCM" Id="P5">
               <N_Position axis="x, y" unit="m">0.4, 0.8 </N_Position>
               <Processor Name="Pentium">
                      <P_Type>III</P_Type>
               </Processor>
               <Memory unit="KB">128</Memory>
        </NODE>

</ARCHITECTURE_MODEL>
```

B. Behavioral Model

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE BEHAVIOURAL_MODEL[
        <!ELEMENT BEHAVIOURAL_MODEL (PROCESS+, ARC+)>
```

```
        <!ELEMENT PROCESS (WCET, Memory,
                                Sensor*, Actuator*)>
        <!ATTLIST PROCESS Name CDATA #REQUIRED
                                Id CDATA #REQUIRED>
        <!ELEMENT WCET (#PCDATA)>
        <!ATTLIST WCET unit CDATA #REQUIRED>
        <!ELEMENT Memory (#PCDATA)>
        <!ATTLIST Memory unit CDATA #REQUIRED>
        <!ELEMENT Sensor (#PCDATA)>
        <!ELEMENT Actuator (#PCDATA)>

        <!ELEMENT ARC (Src, Dest, Delay)>
        <!ATTLIST ARC Name CDATA #REQUIRED
                        Id CDATA #REQUIRED>
        <!ELEMENT Src (#PCDATA)>
        <!ELEMENT Dest (#PCDATA)>
        <!ELEMENT Delay (#PCDATA)>
        <!ATTLIST Delay unit CDATA #REQUIRED>
]>

<BEHAVIOURAL_MODEL>
        <PROCESS Name="PR1" Id="PR1">
                <WCET unit="ms">0</WCET>
                <Memory unit="KB">1</Memory>
                <Sensor>Driver</Sensor>
                <Sensor>Cruise Controller</Sensor>
        </PROCESS>

        <PROCESS Name="PR2" Id="PR2">
                <WCET unit="ms">12</WCET>
                <Memory unit="KB">10</Memory>
        </PROCESS>

        <PROCESS Name="PR3" Id="PR3">
                <WCET unit="ms">7</WCET>
                <Memory unit="KB">2</Memory>
        </PROCESS>

        <PROCESS Name="PR4" Id="PR4">
                <WCET unit="ms">10</WCET>
                <Memory unit="KB">5</Memory>
        </PROCESS>

        <PROCESS Name="PR5" Id="PR5">
                <WCET unit="ms">5</WCET>
                <Memory unit="KB">10</Memory>
                <Sensor>Speed</Sensor>
                <Sensor>Acceleration</Sensor>
        </PROCESS>

        <PROCESS Name="PR6" Id="PR6">
                <WCET unit="ms">18</WCET>
                <Memory unit="KB">10</Memory>
        </PROCESS>

        <PROCESS Name="PR7" Id="PR7">
                <WCET unit="ms">14</WCET>
                <Memory unit="KB">7</Memory>
        </PROCESS>

        <PROCESS Name="PR8" Id="PR8">
                <WCET unit="ms">8</WCET>
```

```
        <Memory unit="KB">3</Memory>
</PROCESS>

<PROCESS Name="PR9" Id="PR9">
        <WCET unit="ms">5</WCET>
        <Memory unit="KB">2</Memory>
</PROCESS>

<PROCESS Name="PR10" Id="PR10">
        <WCET unit="ms">10</WCET>
        <Memory unit="KB">4</Memory>
</PROCESS>

<PROCESS Name="PR11" Id="PR11">
        <WCET unit="ms">6</WCET>
        <Memory unit="KB">2</Memory>
</PROCESS>

<PROCESS Name="PR12" Id="PR12">
        <WCET unit="ms">7</WCET>
        <Memory unit="KB">5</Memory>
</PROCESS>

<PROCESS Name="PR13" Id="PR13">
        <WCET unit="ms">11</WCET>
        <Memory unit="KB">5</Memory>
</PROCESS>

<PROCESS Name="PR14" Id="PR14">
        <WCET unit="ms">5</WCET>
        <Memory unit="KB">2</Memory>
</PROCESS>

<PROCESS Name="PR15" Id="PR15">
        <WCET unit="ms">8</WCET>
        <Memory unit="KB">5</Memory>
</PROCESS>

<PROCESS Name="PR16" Id="PR16">
        <WCET unit="ms">11</WCET>
        <Memory unit="KB">7</Memory>
</PROCESS>

<PROCESS Name="PR17" Id="PR17">
        <WCET unit="ms">15</WCET>
        <Memory unit="KB">8</Memory>
</PROCESS>

<PROCESS Name="PR18" Id="PR18">
        <WCET unit="ms">6</WCET>
        <Memory unit="KB">3</Memory>
</PROCESS>

<PROCESS Name="PR19" Id="PR19">
        <WCET unit="ms">13</WCET>
        <Memory unit="KB">2</Memory>
</PROCESS>

<PROCESS Name="PR20" Id="PR20">
        <WCET unit="ms">5</WCET>
        <Memory unit="KB">2</Memory>
</PROCESS>
```

```
<PROCESS Name="PR21" Id="PR21">
       <WCET unit="ms">20</WCET>
       <Memory unit="KB">2</Memory>
</PROCESS>

<PROCESS Name="PR22" Id="PR22">
       <WCET unit="ms">17</WCET>
       <Memory unit="KB">1</Memory>
</PROCESS>

<ROCESS Name="PR23" Id="PR23">
       <WCET unit="ms">9</WCET>
       <Memory unit="KB">7</Memory>
</PROCESS>

<PROCESS Name="PR24" Id="PR24">
       <WCET unit="ms">5</WCET>
       <Memory unit="KB">2</Memory>
</PROCESS>

<PROCESS Name="PR25" Id="PR25">
       <WCET unit="ms">6</WCET>
       <Memory unit="KB">3</Memory>
</PROCESS>

<PROCESS Name="PR26" Id="PR26">
       <WCET unit="ms">5</WCET>
       <Memory unit="KB">2</Memory>
</PROCESS>

<PROCESS Name="PR27" Id="PR27">
       <WCET unit="ms">5</WCET>
       <Memory unit="KB">2</Memory>
       <Actuator>Speed</Actuator>
       <Actuator>Acceleration</Actuator>
</PROCESS>

<PROCESS Name="PR28" Id="PR28">
       <WCET unit="ms">12</WCET>
       <Memory unit="KB">8</Memory>
</PROCESS>

<PROCESS Name="PR29" Id="PR29">
       <WCET unit="ms">10</WCET>
       <Memory unit="KB">7</Memory>
</PROCESS>

<PROCESS Name="PR30" Id="PR30">
       <WCET unit="ms">7</WCET>
       <Memory unit="KB">5</Memory>
</PROCESS>

<PROCESS Name="PR31" Id="PR31">
       <WCET unit="ms">6</WCET>
       <Memory unit="KB">4</Memory>
</PROCESS>

<PROCESS Name="PR32" Id="PR32">
       <WCET unit="ms">0</WCET>
       <Memory unit="KB">1</Memory>
       <Actuator>Driver</Actuator>
```

```
        <Actuator>Cruise Controller</Actuator>
</PROCESS>

<ARC Name="ARC1" Id="ARC1">
        <Src>PR1</Src>
        <Dest>PR2</Dest>
        <Delay unit="ms">0</Delay>
</ARC>

<ARC Name="ARC2" Id="ARC2">
        <Src>PR2</Src>
        <Dest>PR3</Dest>
        <Delay unit="ms">1</Delay>
</ARC>

<ARC Name="ARC3" Id="ARC3">
        <Src>PR3</Src>
        <Dest>PR4</Dest>
        <Delay unit="ms">0</Delay>
</ARC>

<ARC Name="ARC4" Id="ARC4">
        <Src>PR4</Src>
        <Dest>PR5</Dest>
        <Delay unit="ms">2</Delay>
</ARC>

<ARC Name="ARC5" Id="ARC5">
        <Src>PR5</Src>
        <Dest>PR6</Dest>
        <Delay unit="ms">0</Delay>
</ARC>

<ARC Name="ARC6" Id="ARC6">
        <Src>PR6</Src>
        <Dest>PR7</Dest>
        <Delay unit="ms">4</Delay>
</ARC>

<ARC Name="ARC7" Id="ARC7">
        <Src>PR7</Src>
        <Dest>PR8</Dest>
        <Delay unit="ms">4</Delay>
</ARC>

<ARC Name="ARC8" Id="ARC8">
        <Sc>PR8</Src>
        <Dest>PR9</Dest>
        <Delay unit="ms">0</Delay>
</ARC>

<ARC Name="ARC9" Id="ARC9">
<Src>PR9</Src>
        <Dest>PR10</Dest>
        <Delay unit="ms">0</Delay>
</ARC>

<ARC Name="ARC10" Id="ARC10">
        <Src>PR10</Src>
        <Dest>PR11</Dest>
        <Delay unit="ms">0</Delay>
</ARC>
```

```
<ARC Name="ARC11" Id="ARC11">
       <Src>PR11</Src>
       <Dest>PR14</Dest>
       <Delay unit="ms">0</Delay>
</ARC>

<ARC Name="ARC12" Id="ARC12">
       <Src>PR7</Src>
       <Dest>PR12</Dest>
       <Delay unit="ms">0</Delay>
</ARC>

<ARC Name="ARC13" Id="ARC13">
       <Src>PR12</Src>
<Dest>PR13</Dest>
<Delay unit="ms">0</Delay>
</ARC>

<ARC Name="ARC14" Id="ARC14">
       <Src>PR13</Src>
       <Dest>PR14</Dest>
       <Delay unit="ms">2</Delay>
</ARC>

<ARC Name="ARC15" Id="ARC15">
       <Src>PR14</Src>
       <Dest>PR15</Dest>
       <Delay unit="ms">2</Delay>
</ARC>

<ARC Name="ARC16" Id="ARC16">
       <Src>PR15</Src>
       <Dest>PR16</Dest>
       <Delay unit="ms">2</Delay>
</ARC>

<ARC Name="ARC17" Id="ARC17">
       <Src>PR16</Src>
       <Dest>PR27</Dest>
       <Delay unit="ms">0</Delay>
</ARC>

<ARC Name="ARC18" Id="ARC18">
       <Src>PR5</Src>
       <Dest>PR17</Dest>
       <Delay unit="ms">2</Delay>
</ARC>

<ARC Name="ARC19" Id="ARC19">
       <Src>PR17</Src>
       <Dest>PR18</Dest>
       <Delay unit="ms">0</Delay>
</ARC>

<ARC Name="ARC20" Id="ARC20">
       <Src>PR18</Src>
       <Dest>PR19</Dest>
       <Delay unit="ms">0</Delay>
</ARC>

<ARC Name="ARC21" Id="ARC21">
```

```
        <Src>PR19</Src>
        <Dest>PR20</Dest>
        <Delay unit="ms">4</Delay>
</ARC>

<ARC Name="ARC22" Id="ARC22">
        <Src>PR20</Src>
        <Dest>PR21</Dest>
        <Delay unit="ms">4</Delay>
</ARC>

<ARC Name="ARC23" Id="ARC23">
        <Src>PR21</Src>
        <Dest>PR24</Dest>
        <Delay unit="ms">0</Delay>
</ARC>

<ARC Name="ARC24" Id="ARC24">
        <Src>PR20</Src>
        <Dest>PR22</Dest>
        <Delay unit="ms">0</Delay>
</ARC>

<ARC Name="ARC25" Id="ARC25">
        <Src>PR22</Src>
        <Dest>PR23</Dest>
<Delay unit="ms">0</Delay>
</ARC>

<ARC Name="ARC26" Id="ARC26">
        <Src>PR23</Src>
        <Dest>PR24</Dest>
        <Delay unit="ms">4</Delay>
</ARC>

<ARC Name="ARC27" Id="ARC27">
        <Src>PR24</Src>
        <Dest>PR25</Dest>
        <Delay unit="ms">2</Delay>
</ARC>

<ARC Name="ARC28" Id="ARC28">
        <Src>PR25</Src>
        <Dest>PR26</Dest>
        <Delay unit="ms">2</Delay>
</ARC>

<ARC Name="ARC29" Id="ARC29">
        <Src>PR26</Src>
        <Dest>PR27</Dest>
        <Delay unit="ms">2</Delay>
</ARC>

<ARC Name="ARC30" Id="ARC30">
        <Src>PR27</Src>
        <Dest>PR32</Dest>
        <Delay unit="ms">0</Delay>
</ARC>

<ARC Name="ARC31" Id="ARC31">
        <Src>PR1</Src>
        <Dest>PR28</Dest>
```

```
            <Delay unit="ms">1</Delay>
        </ARC>

        <ARC Name="ARC32" Id="ARC32">
            <Src>PR28</Src>
            <Dest>PR29</Dest>
            <Delay unit="ms">0</Delay>
        </ARC>

        <ARC Name="ARC33" Id="ARC33">
            <Src>PR29</Src>
            <Dest>PR30</Dest>
            <Delay unit="ms">1</Delay>
        </ARC>

        <ARC Name="ARC34" Id="ARC34">
            <Src>PR30</Src>
            <Dest>PR31</Dest>
            <Delay unit="ms">0</Delay>
        </ARC>

        <ARC Name="ARC35" Id="ARC35">
            <Src>PR31</Src>
            <Dest>PR32</Dest>
            <Delay unit="ms">1</Delay>
        </ARC>


</BEHAVIOURAL_MODEL>
```

C. Mapping Model

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE MAPPING[
        <!ELEMENT MAPPING (MAP+)>
        <!ELEMENT MAP (Process*)>
        <!ATTLIST MAP Node CDATA #REQUIRED>
        <!ELEMENT Process (#PCDATA)>
]>

        <MAPPING>

        <MAP Node="P1">
            <Process> PR1 </Process>
            <Process> PR2 </Process>
            <Process> PR30 </Process>
            <Proess> PR31 </Process>
        </MAP>

        <MAP Node="P2">
            <Process> PR3 </Process>
            <Process> PR4 </Process>
            <Process> PR27 </Process>
            <Process> PR28 </Process>
            <Process> PR29 </Process>
            <Process> PR32 </Process>
        </MAP>

        <MAP Node="P3">
```

```
            <Process> PR5 </Process>
            <Process> PR6 </Process>
            <Process> PR8 </Process>
            <Process> PR9 </Process>
            <Process> PR10 </Process>
            <Process> PR11 </Process>
            <Process> PR14 </Process>
            <Process> PR16 </Process>
      </MAP>

      <MAP Node="P4">
            <Process> PR17 </Process>
            <Process> PR18 </Process>
            <Process> PR19 </Process>
            <Process> PR21 </Process>
            <Process> PR24 </Process>
            <Process> PR26</Process>
      </MAP>

      <MAP Node="P5">
            <Process> PR7 </Process>
            <Process> PR12 </Process>
            <Process> PR13 </Process>
            <Process> PR20 </Process>
            <Process> PR22 </Process>
            <Process> PR23 </Process>
            <Process> PR25</Process>
      </MAP>

      <MAP Node="B1">
            <Process> ARC2 </Process>
            <Process> ARC4 </Process>
            <Process> ARC6 </Process>
            <Process> ARC7 </Process>
            <Process> ARC14 </Process>
            <Process> ARC15 </Process>
            <Process> ARC16 </Process>
            <Process> ARC18 </Process>
            <Process> ARC21 </Process>
            <Process> ARC22 </Process>
            <Process> ARC26 </Process>
            <Process> ARC27 </Process>
            <Process> ARC29 </Process>
            <Process> ARC31 </Process>
            <Process> ARC33 </Process>
            <Process> ARC35 </Process>
      </MAP>

</MAPPING>
```

D. Schedule Model

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE SCHEDULE[
      <!ELEMENT SCHEDULE (SLOT+)>
      <!ELEMENT SLOT (Start, Duration, Resource)>
      <!ATTLIST SLOT Id CDATA #REQUIRED>
      <!ELEMENT Start (#PCDATA)>
      <!ATTLIST Start unit CDATA #REQUIRED>
      <!ELEMENT Duration (#PCDATA)>
```

```
	<!ATTLIST Duration unit CDATA #REQUIRED>
	<!ELEMENT Resource (#PCDATA)>
]>

<SCHEDULE>

	<SLOT Id="PR1">
		<Start unit="ms">0</Start>
		<Duration unit="ms">0</Duration>
		<Resource>P6</Resource>
	</SLOT>

	<SLOT Id="PR2">
		<Start unit="ms">0</Start>
		<Duration unit="ms">12</Duration>
		<Resource>P1</Resource>
	</SLOT>

	<SLOT Id="PR28">
		<Start unit="ms">0</Start>
		<Duration unit="ms">12</Duration>
		<Resource>P2</Resource>
	</SLOT>

	<SLOT Id="ARC2">
		<Start unit="ms">12</Start>
		<Duraion unit="ms">1</Duration>
		<Resource>B1</Resource>
	</SLOT>

	<SLOT Id="PR3">
		<Start unit="ms">13</Start>
		<Duration unit="ms">7</Duration>
		<Resource>P2</Resource>
	</SLOT>

	<SLOT Id="PR4">
		<Start unit="ms">20</Start>
		<Duration unit="ms">10</Duration>
		<Resource>P2</Resource>
	</SLOT>

	<SLOT Id="ARC4">
		<Start unit="ms">30</Start>
		<Duration unit="ms">2</Duration>
		<Resource>B1</Resource>
	</SLOT>

	<SLOT Id="PR5">
		<Start unit="ms">32</Start>
		<Duration unit="ms">5</Duration>
		<Resource>P3</Resource>
	</SLOT>

	<SLOT Id="ARC18">
		<Start unit="ms">37</Start>
		<Duration unit="ms">2</Duration>
		<Resource>B1</Resource>
	</SLOT>

	<SLOT Id="PR29">
		<Start unit="ms">30</Start>
```

```
        <Duration unit="ms">10</Duration>
        <Resource>P2</Resource>
</SLOT>

<SLOT Id="ARC33">
        <Start unit="ms">40</Start>
        <Duration unit="ms">1</Duration>
        <Resource>B1</Resource>
</SLOT>

<SLOT Id="PR30">
        <Start unit="ms">41</Start>
        <Duration unit="ms">7</Duration>
        <Resource>P1</Resource>
</SLOT>

<SLOT Id="PR31">
        <Start unit="ms">48</Start>
        <Duration unit="ms">6</Duration>
        <Resource>P1</Resource>
</SLOT>

<SLOT Id="PR17">
        <Start unit="ms">39</Start>
        <Duration unit="ms">15</Duration>
        <Resource>P4</Resource>
</SLOT>

<SLOT Id="PR6">
        <Start unit="ms">37</Start>
        <Duration unit="ms">18</Duration>
        <Resource>P3</Resource>
</SLOT>

<SLOT Id="ARC35">
        <Start unit="ms">54</Start>
        <Duration unit="ms">1</Duration>
        <Resource>B1</Resource>
</SLOT>

<SLOT Id="ARC6">
        <Start unit="ms">55</Start>
        <Duration unit="ms">4</Duration>
        <Resource>B1</Resource>
        </SLOT>

<SLOT Id="PR18">
        <Start unit="ms">54</Start>
        <Duration unit="ms">6</Duration>
        <Resource>P4</Resource>
</SLOT>

<SLOT Id="PR19">
        <Start unit="ms">60</Start>
        <Duration unit="ms">13</Duration>
        <Resource>P4</Resource>
</SLOT>

<SLOT Id="PR7">
        <Start unit="ms">59</Start>
        <Duration unit="ms">14</Duration>
        <Resource>P5</Resource>
```

```
</SLOT>

<SLOT Id="ARC7">
       <Start unit="ms">73</Start>
       <Duration unit="ms">4</Duration>
       <Resource>B1</Resource>
</SLOT>

<SLOT Id="PR12">
       <Start unit="ms">73</Start>
       <Duration unit="ms">7</Duration>
       <Resource>P5</Resource>
</SLOT>

<SLOT Id="ARC21">
       <Start unit="ms">77</Start>
       <Duration unit="ms">4</Duration>
       <Resource>B1</Resource>
</SLOT>

<SLOT Id="PR8">
       <Start unit="ms">77</Start>
       <Duration unit="ms">8</Duration>
       <Resource>P3</Resource>
</SLOT>

<SLOT Id="PR9">
       <Start unit="ms">85</Start>
       <Duration unit="ms">5</Duration>
       <Resource>P3</Resource>
</SLOT>

<SLOT Id="PR13">
       <Start unit="ms">80</Start>
       <Duration unit="ms">11</Duration>
       <Resource>P5</Resource>
</SLOT>

<SLOT Id="ARC14">
       <Start unit="ms">91</Start>
       <Duration unit="ms">2</Duration>
       <Resource>B1</Resource>
</SLOT>

<SLOT Id="PR20">
       <Start unit="ms">91</Start>
       <Duration unit="ms">5</Duration>
       <Resource>P5</Resource>
</SLOT>

<SLOT Id="PR10">
       <Start unit="ms">90</Start>
       <Duration unit="ms">10</Duration>
       <Resource>P3</Resource>
</SLOT>

<SLOT Id="ARC22">
       <Start unit="ms">100</Start>
       <Duration unit="ms">4</Duration>
       <Resource>B1</Resource>
</SLOT>
```

```
<SLOT Id="PR11">
       <Start unit="ms">100</Start>
       <Duration unit="ms">6</Duration>
       <Resource>P3</Resource>
</SLOT>

<SLOT Id="PR14">
       <Start unit="ms">106</Start>
       <Duration unit="ms">5</Duration>
       <Resource>P3</Resource>
</SLOT>

       <SLOT Id="PR22">
       <Start unit="ms">96</Start>
       <Duration unit="ms">17</Duration>
       <Resource>P5</Resource>
</SLOT>

<SLOT Id="ARC15">
       <Start unit="ms">111</Start>
       <Duration unit="ms">2</Duration>
       <Resource>B1</Resource>
</SLOT>

<SLOT Id="PR15">
       <Start unit="ms">113</Start>
       <Duration unit="ms">8</Duration>
       <Resource>P1</Resource>
</SLOT>

<SLOT Id="PR23">
       <Start unit="ms">113</Start>
       <Duration unit="ms">9</Duration>
       <Resource>P5</Resource>
</SLOT>

<SLOT Id="PR21">
       <Start unit="ms">104</Start>
       <Duration unit="ms">20</Duration>
       <Resource>P4</Resource>
</SLOT>

<SLOT Id="ARC26">
       <Start unit="ms">122</Start>
       <Duration unit="ms">4</Duration>
       <Resource>B1</Resource>
</SLOT>

<SLOT Id="ARC16">
       <Start unit="ms">126</Start>
       <Duration unit="ms">2</Duration>
       <Resource>B1</Resource>
</SLOT>

<SLOT Id="PR24">
       <Start unit="ms">126</Start>
       <Duration unit="ms">5</Duration>
       <Resource>P4</Resource>
</SLOT>

<SLOT Id="ARC27">
       <Start unit="ms">131</Start>
```

```
            <Duration unit="ms">2</Duration>
            <Resource>B1</Resource>
      </SLOT>

      <SLOT Id="PR16">
            <Start unit="ms">128</Start>
            <Duration unit="ms">11</Duration>
            <Resource>P3</Resource>
      </SLOT>

      <SLOT Id="PR25">
            <Start unit="ms">133</Start>
            <Duration unit="ms">6</Duration>
            <Resource>P5</Resource>
      </SLOT>

      <SLOT Id="ARC28">
            <Start unit="ms">139</Start>
            <Duration unit="ms">2</Duration>
            <Resource>B1</Resource>
      </SLOT>

      <SLOT Id="PR26">
            <Start unit="ms">141</Start>
            <Duration unit="ms">5</Duration>
            <Resource>P4</Resource>
      </SLOT>

      <SLOT Id="ARC29">
            <Start unit="ms">146</Start>
            <Duration unit="ms">2</Duration>
            <Resource>B1</Resource>
      </SLOT>

      <SLOT Id="PR27">
            <Start unit="ms">148</Start>
            <Duration unit="ms">5</Duration>
            <Resource>P2</Resource>
      </SLOT>

      <SLOT Id="PR32">
            <Start unit="ms">153</Start>
            <Duration unit="ms">0</Duration>
            <Resource>P2</Resource>
      </SLOT>

</SCHEDULE>
```

Appendix II

A. xLinkit

Document Set

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE DocumentSet [
       <!ELEMENT DocumentSet (Description?, (Set | DocFile)*)>
       <!ATTLIST DocumentSet name CDATA #REQUIRED>
       <!ELEMENT Description (#PCDATA)>

       <!-- Both a set of xml files and a file are simply identified by an URL -->

       <!ELEMENT Set EMPTY>
       <!ATTLIST Set href CDATA #REQUIRED>
       <!ELEMENT DocFile EMPTY>
       <!ATTLIST DocFile href CDATA #REQUIRED
                                fetcher CDATA "FileFetcher">

]>

<DocumentSet name="BehaviorDoc">

       <Description>Doc related to the Cruise Controller environment</Description>

       <DocFile href="http://www.ida.liu.se/~x01syeak/xlinkit/ schedule.xml"/>

       <DocFile href="http://www.ida.liu.se/~x01syeak/xlinkit/ behavior.xml"/>

</DocumentSet>
```

Rule Set

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE RuleSet [
       <!ELEMENT RuleSet (Description?, (Set | RuleFile)*)>
       <!ATTLIST RuleSet name CDATA #REQUIRED>
       <!ELEMENT Description (#PCDATA)>

       <!-- Both a set of xml files and a file are simply identified by an URL -->

       <!ELEMENT Set EMPTY>
       <!ATTLIST Set href CDATA #REQUIRED>
       <!ELEMENT RuleFile EMPTY>
       <!ATTLIST RuleFile href CDATA #REQUIRED
        xpath CDATA "/consistencyruleset/consistencyrule">
]>

<RuleSet name="ScheduleRule">

      <Description>Rules related to the Cruise Controller environment</Description>
```

```
      <RuleFile href="http://www.ida.liu.se/~x01syeak/xlinkit/ rule4.xml"
      xpath="/consistencyruleset/consistencyrule"/>

</RuleSet>
```

Rule

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE consistencyruleset[
      <!ELEMENT consistencyruleset (globalset*, consistencyrule+)>
      <!ELEMENT globalset EMPTY>
      <!ATTLIST globalset id ID #REQUIRED
                  xpath CDATA #REQUIRED>
      <!ELEMENT consistencyrule (description?,linkgeneration?,forall)>
      <!ATTLIST consistencyrule d ID #REQUIRED>
      <!ELEMENT description (#PCDATA)>
      <!ELEMENT linkgeneration (consistent?,inconsistent?,
                              liminatesymmetry?)>
      <!ELEMENT consistent EMPTY>
      <!ATTLIST consistent status (on | off) "on">
      <!ELEMENT inconsistent EMPTY>
      <!ATTLIST inconsistent status (on | off) "on">
      <!ELEMENT eliminatesymmetry EMPTY>
      <!ATTLIST eliminatesymmetry status (on | off) "off">
      <!ELEMENT forall exists|forall|and|or|implies|not|equal|notequal|
                        same|ubset|intersect)?>
      <!ATTLIST forall var CDATA #REQUIRED
                        in CDATA #REQUIRED
                        mode (exhaustive | instance) "exhaustive">

      <!ELEMENT exists (exists|forall|and|or|implies|not|equal|notequal|
                        same|subset|intersect)?>
      <!ATTLIST exists var CDATA #REQUIRED
                        in CDATA #REQUIRED
                        mode (exhaustive | instance) "exhaustive">

      <!ELEMENT and (exists|forall|and|or|implies|not|equal|notequal|
                        same|subset|intersect)*>

      <!ELEMENT or (exists|forall|and|or|implies|not|equal|notequal|
                        same|subset|intersect)*>

      <!ELEMENT implies (exists|forall|and|or|implies|not|equal|notequal|
                        same|subset|intersect)*>

      <!ELEMENT not (exists|forall|and|or|implies|not|equal|notequal|
                        same|subset|intersect)>

      <!ELEMENT equal EMPTY>
      <!ATTLIST equal op1 CDATA #REQUIRED
                        op2 CDATA #REQUIRED>

      <!ELEMENT notequal EMPTY>
      <!ATTLIST notequal op1 CDATA #REQUIRED
                        op2 CDATA #REQUIRED>

      <!ELEMENT same EMPTY>
      <!ATTLIST same op1 CDATA #REQUIRED
                        op2 CDATA #REQUIRED>
```

```
       <!ELEMENT subset EMPTY>
       <!ATTLIST subset op1 CDATA #REQUIRED
                        op2 CDATA #REQUIRED
                        size CDATA "0">

       <!ELEMENT intersect EMPTY>
       <!ATTLIST intersect op1 CDATA #REQUIRED
                        op2 CDATA #REQUIRED
                        size CDATA "0">

]>

<consistencyruleset>

   <globalset id="schedule" xpath="/SCHEDULE/SLOT"/>
   <globalset id="process_behaviour" xpath="/BEHAVIOURAL_MODEL/PROCESS"/>
   <globalset id="arc_behaviour" xpath="/BEHAVIOURAL_MODEL/ARC"/>

      <consistencyrule id="r1">

       <description>
         The duration of execution on a processor should be the same as the

         one in specification
       </description>

       <forall var="a" in="$schedule">

          <or>
             <exists var="p" in="$process_behaviour">
                  <and>
                     <equal op1="$a/@Id" op2="$p/@Id"/>
                     <equal op1="$a/Duration/text()" op2="$p/WCET/text()"/>
                  </and>

              </exists>

              <exists var="c" in="$arc_behaviour">
                  <and>
                     <equal op1="$a/@Id" op2="$c/@Id"/>
                     <equal op1="$a/Duration/text()" op2="$c/Delay/text()"/>
                  </and>
              </exists>

          </or>

       </forall>

      </consistencyrule>

</consistencyruleset>
```

B. XQuery

Rule-9. Every process except the first one should have a predecessor.

```
<result>{

 for $src in distinct-values(document("data/sweb/behaviour.xml")//ARC/Src)
 let $dest := document("data/sweb/behaviour.xml")//ARC/Dest

       return
            if ($src=$dest)
            then ""
             else $src, "The process without precedence"

}</result>
```

The result is:

```
   <?xml version="1.0"?>
   <quip:result xmlns:quip="http://namespaces.softwareag.com/tamino/quip/">
      <result>
         <Src>PR1</Src>
         The process without precedence
      </result>
   </quip:result>
```

Rule-10. Every process except the last one should have a successor.

```
<result>{

 for $dest in distinct-values(document("data/sweb/behaviour.xml")//ARC/Dest)
 let $src := document("data/sweb/behaviour.xml")//ARC/Src

       return
            if ($src=$dest)
            then ""
            else $dest, "The process without successor"

}</result>
```

The result is:

```
   <?xml version="1.0"?>
   <quip:result xmlns:quip="http://namespaces.softwareag.com/tamino/quip/">
      <result>
         <Dest>PR32</Dest>
         The process without successor
      </result>
   </quip:result>
```