Towards an Application of Model-Based Codesign: An Autonomous, Intelligent Cruise Controller

S. Schulz and J.W. Rozenblit
Dept. of Electrical and Computer Engineering
The University of Arizona
Tucson, Arizona 85721-0104
USA

{sschulz|jr@ece.arizona.edu}

K. Buchenrieder
Siemens AG
ZFE T SE 5
Otto-Hahn-Ring 6
81739 Munich
Germany
buchen@zfe.siemens.de

Abstract

In this paper, the domain of automotive safety is addressed. A specific application, i.e., autonomous, intelligent cruise controller, is selected. Model-based techniques which facilitate implementation independent specification and design of such a system are discussed. A set of systems requirements, the underlying object and behavioral models are given. In conclusion, postulates are discussed for the physical realization of the presented application.

1. Introduction

In the last decade, we have witnessed a sizable growth in the application of electronics in automotive industry. Initial applications ranged from electronic fuel injection devices to motor control units. With the advent of more powerful microprocessors, a new generation of devices is emerging. They are intended to improve safety of the automobile and its environment. A comparison of the automotive traffic with its counterparts, i.e., air and rail transportation shows that the automobile-based mode of transportation is the least controlled and structured. Therefore, it is imperative that passive and active devices be developed that provide driver assistance and serve as collision deterrents.

In this paper, we categorize the majors aspects of automotive safety, select a specific application, i.e., an autonomous, intelligent cruise controller, and propose model-based techniques for a realization of such a device.

2. Model-based codesign

A multitude of codesign techniques and methodologies are employed in academic and commercial environments [7,8,9,12,18,19]. A common trend appears to be emerging in which a clear shift in design and development paradigms is occurring. Initial approaches would foster immediate partitioning into hardware (HW) and software (SW) components, pursue HW and SW development threads in isolation from each other, and often place a stronger emphasis on hardware than software [1,11,12]. Those trends stem from certain misconceptions regarding the design of heterogeneous systems, namely, the beliefs that HW and SW can be developed separately, that inadequacies in the hardware components can be compensated by software revisions, and that the integration of subsystems should be postponed until the end of the design cycle. Numerous authors point to the deficiencies of the traditional codesign frameworks [1,2,5,9,12]. They strongly advocate a process that fosters the integration of the HW and SW perspectives. Thus, a unified representation is needed for modeling a system independently of its implementation in hardware or software. A number of modeling representations and formalisms have been proposed in the literature and applied to heterogeneous systems design [14]. They include, but are not limited to, *data flow diagrams, finite state machines, Petri nets, specialized algebras* and *object oriented techniques.*

Our position is that the formal specification techniques are of limited effectiveness without a systematic modeling methodology guiding their use throughout the codesign process. With Siemens Personnel, we are

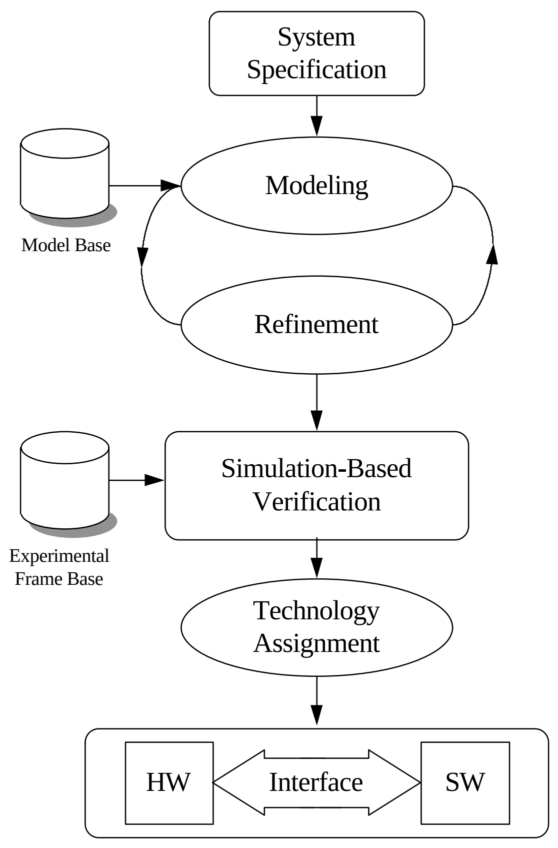


Figure 1. Model-Based Codesign

advocating an advanced codesign methodology in which abstract hardware/software models can be developed and validated prior to technology assignment. This approach is illustrated in Figure 1. The benefits of using an abstract system specification as a unifying HW/SW representation are: a) late partitioning, b) stepwise refinement fostering component reuse regardless of their technology, and c) the availability of a reference architecture in which components can be specified at multiple abstraction levels (e.g. system, instruction set, register-transfer, logic, or circuit level in the case of hardware, and OS, applications, utilities, etc. in the case of software components.

The above methodology, has been described in detail in [1]. Here, we briefly summarize its major tenets. Our basic supposition is that models serve as design blueprints for developing system. The modeling phase denotes the definition of the system's components (its object model) and associated behaviors. This phase results in the system's model that is subject to a validation and stepwise refinement process. A validated model is simulated in a specific set of experimental conditions (called *experimental frames)* to verify its adherence to the initial requirements, constraints, and design objectives. Technology assignment is then carried out from the verified model specification.

Our approach lends itself well to the automotive application at hand. The car industry is increasingly reducing design cycles and is keen saving costs. At the same time, the electronic components used in automobiles become highly complex embedded systems. The traditional, separate design of hardware and software and software components is difficult to justify, especially in view of the sophistication of the functional characteristics of the components being built.

In the following section, we define a set of requirements for our proposed application of the model-based codesign approach.

3. Requirements for automotive safety electronic systems

We begin with a set of high level requirements that automotive electronics should fulfill. Cost and schedules are general design and realization process constraints. Reductions in weight and space are required in order to produce fuel efficient vehicles and to conform to stricter environmental pollution laws. The electronics should be minimized in weight and size while providing maximum safety. It has to withstand extreme operating conditions of a car, e.g., extreme fluctuations in temperature and relative humidity.

A safety device should be able to detect and suggest solutions to handle hazardous driving conditions. It should interface with standard diagnostic stations which are used for detailed error analysis. An emphasis must be placed on a user friendly interface that could update a driver about the current status of the vehicle and its environment.

Additionally, we perceive requirements that apply to specific subsystems. As indicated before, we are focusing on a model-based codesign of a unit called autonomous, intelligent cruise control unit (AICC). The AICC system can be seen as an extension of the regular cruise control, not only keeping a fixed speed, but also adapting to the speed of the vehicle ahead. It controls the relative speed between two vehicles traveling in the same lane. Furthermore, it asserts longitudinal elements of control but no lateral control. Although the system is autonomous, meaning it does not rely on communication between vehicles, the driver remains in full control since he or she can override the device, e.g., by braking.

The circuitry of a safety unit must satisfy real-time constraints. It should not fail in emergency situations. A standard cruise control nowadays does not represent a true real-time design problem. However, the design of an AICC system must take into account large amounts of data, especially from the vision sensors, and process such data in a timely manner so that safety requirements are met.

In the design of AICC, it is necessary to guarantee that the safety distance is kept within a small margin of error under normal traffic conditions. In our design, this refers to the vehicles in front and back. The device should differentiate between obstacles and moving objects, warn the driver and suggest or take appropriate actions.

We elaborate on the AICC specifications in Section V. First we provide a general object model that allows us to structure the domain of automotive safety aspects and to select a subset of elements that can constitute the autonomous, intelligent cruise controller.

4. Object model definition

Our modeling approach is to first focus on the structural aspects of a domain in which a particular system is being developed. We accomplish this by defining an object model [18] for the domain, by selecting an instance of this domain model that underlies the structure of the system under development, and then by defining behavioral specifications. This methodology has been described in detail in [1].

Domain object models are represented using the system entity structures --- a tree like diagrams that reflect a hierarchy of object decompositions and taxonomies [15,17]. A high level view of various aspects of automotive safety is shown in Figure 2. (The double vertical lines reflect a taxonomy of components. A single line that denotes a decomposition.)

The SES of Figure 2 was specified with a focus on safety systems using micro-electronic devices. First, four broad approaches were taken to divide the domain of *automotive safety* into a *systems* aspect, an aspect considering *collision avoidance*, a *crash safety* aspect and the *micro-electronics* aspect.

An autonomous intelligent cruise control acts much like a regular cruise control. The distinguishing feature is the "safe distance maintenance" to the car in front and back, detection of cars in the blindspots and, in the future, keeping the car in the appropriate lane. More sophisticated versions would work with a location system which could be realized using a Global Positioning System or an Inertial Navigation System. The first alternative has disadvantages in cities or tunnels, or other places where the reception of radio

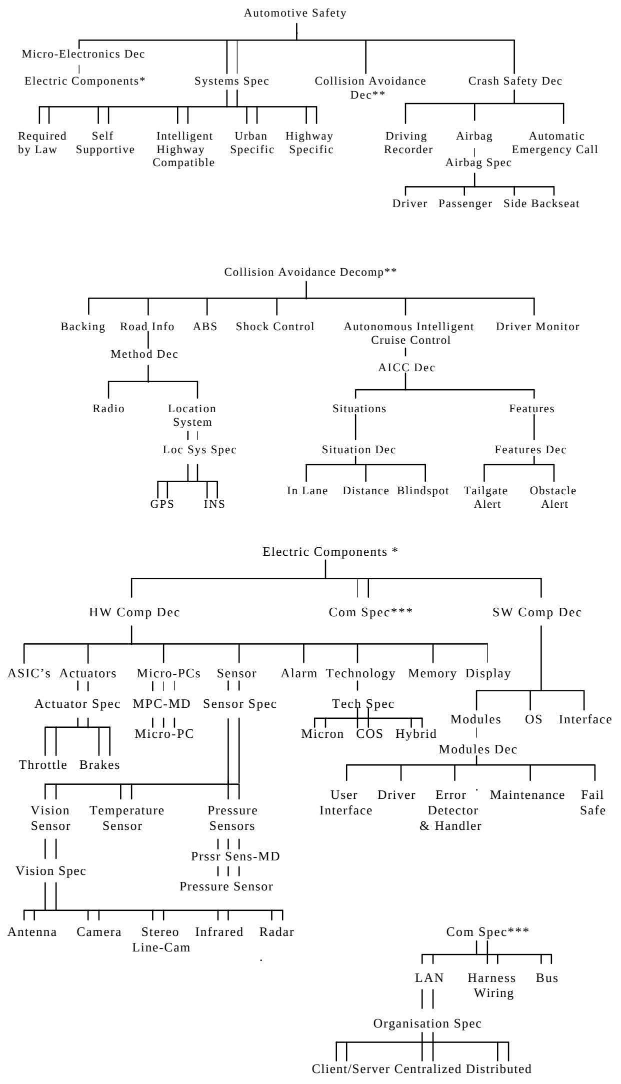


Figure 2. System Entity Structure of the Domain

waves is difficult. For more detailed descriptions of these subsystems, we refer the reader to [3,4,6,13].

The selection of an instance from the domain object model is done in our approach using a knowledge-based technique called *pruning* [17]. In this process, design parameters instantiate the components offered by taxonomies in the system entity structure tree and ensure the coupling of elements identified in the decompositions.

Recall, that our focus is the design of an AICC. Let us assume a safety system with a high budget, high intelligence, medium room and highway as the primary environment. At the *Collision Avoidance,* we select *Autonomous Intelligent Cruise Control* from among the other nodes. For the *Hardware* aspect *ASIC's*, *Actuators*, a *Micro-P*ro*C*essor, a *Sensor*, *Technology*, a *Display* and *Memory* are selected. At the *Actuator* node only the throttle actuator is added since *Collision Avoidance* is *AICC* and the most important reason being that the brake is needed to disable the cruise control. From the *Sensor* node we select the *Vision Sensor* and *Radar*. The *Technology* is *Micron* for all of the devices.

On the *Software* side for the *Micro-PC* an *O*perating *S*ystem, an *interface*, and multiple *components* are chosen. They consist of one *driver* per *sensor* and *actuator*, *maintenance* as well as a *fail-safe*, *error detector & handler* and *user interface* modules. Since the budget is not constrained at the *Communications* aspect, we select a *bus* for the inter-device communication. The SES tree shown in Figure 3 reflects the hierarchy of AICC's components.

Through the above object model instantiation process, we assure that some of the requirements are met prior to the behavioral specification phase. Consider the following design aspects: from a cost standpoint, there are no restrictions. Thus any technology can be selected. Room or size are mostly a hardware problem and therefore constrain the selection of that specific branch in the decision tree. Dealing with real-time needs to be addressed on the software side. An appropriate operating system must implemented based on a priority scheme. Different actions have a different level of importance. In case of the AICC updating the sensor readings should always run at the highest priority. The modules split the task of monitoring. For example, the error detection routine detects if sensors are communicating properly.

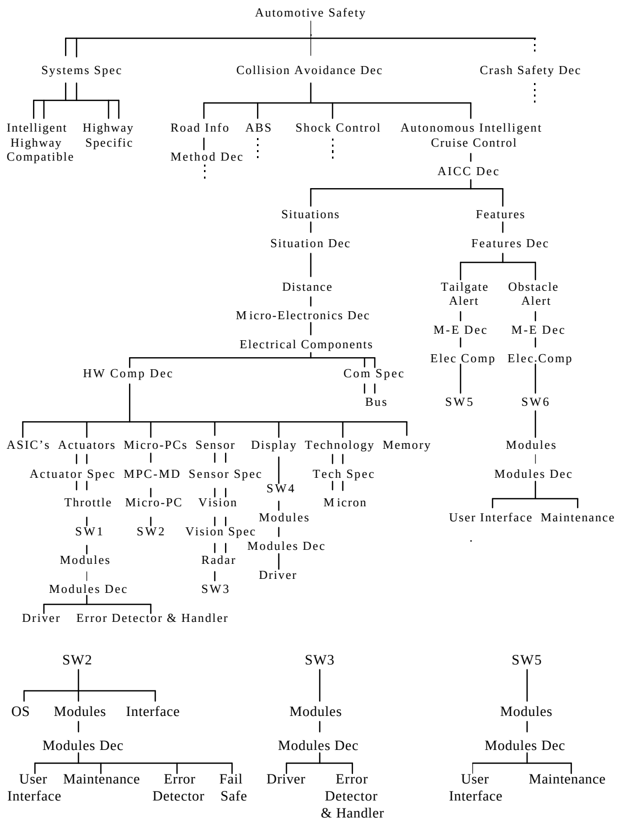


Figure 3. Pruned System Entity Structure for AICC

User friendliness can be accomplished through special user interface routines. They could run at a lower priority to convert technical data to easily comprehensible information. In addition, optimized hardware should be used to support of input and output devices. At the right point in time, the display should provide a sufficient amount of information to the driver. He or she should be able to monitor and comprehend a variety of control functions. To properly address this, complexity has to be minimized and functionality must be maximized.

Meeting real-time constraints without loosing functionality requires that a careful consideration be given as to which parts of the components are written as software routines and which are implemented using Application Specific Integrated Circuits. From our point of view, a model-based design approach to hardware/software codesign is beneficial because we can solve this problem with early simulation and guarantee a relatively short design cycle. In applying our codesign techniques, we have to refine each of the software modules and provide a library of ASIC circuits.

5. Behavior specification

In Figure 4, we show a high level block diagram of the AICC. We now specify the AICC' s behavioral model. The specifications are derived from the following assumptions and informal specifications.

The intelligent cruise control is operated using five buttons: ON/OFF, RESUME/ACC, COAST, OBSTACLE ALERT and TAILGATE ALERT. To prevent the misuse of the cruise control, it cannot be turned on or activated if the speed of the car is less than 35 mph or if the car is currently in reverse or neutral. Tapping the ON/OFF button once turns on the cruise control, and doing that twice turns it off. Once the cruise control is turned off, it enters a disabled state.

When the COAST button is tapped, the cruise control is enabled and the desired speed is maintained if the distance to the vehicles in the front is more than or equal

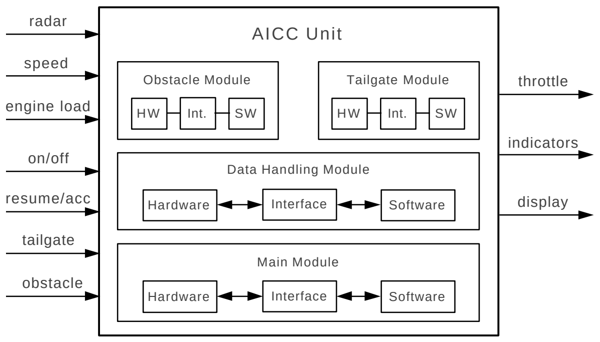


Figure 4. High Level Block Diagram for the AICC

to the required safety distance. If the safe speed is less than the desired speed, the desired speed is saved, the car adjusts the throttle to reach the safety distance and then attempts to resume the desired speed. This accomplishes the goal of keeping the safety distance from vehicles in the front at all times. During the enabled state, power is decreased if the speed exceeds 2 mph above the currently requested speed or increased if it drops 2 mph below that mark. The currently requested speed is either the safe or the desired speed. Pressing the COAST button in the enabled mode results in reducing the desired speed. This results in deceleration if the car is traveling at the desired speed.

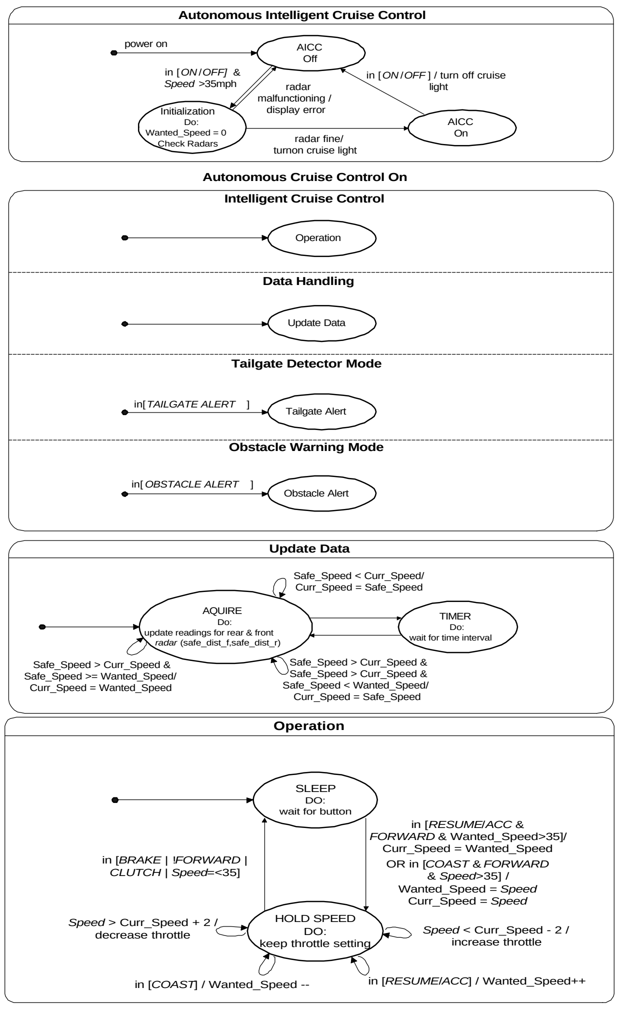
If the TAILGATE ALERT option is activated, the unit checks if the distance to the car in the rear is more than or equal to the safe distance. If the safe distance constraint is violated, the car flashes both rear indicator lights which has the intended effect of having the vehicle behind slow down or pass. To maintain safety distances two radar devices - one in the front and one in the back - are used.

RESUME/ACC resumes to previously set speed (if there is one) when the unit is in the disabled mode. Otherwise, this button increases the desired speed.

An OBSTACLE ALERT option warns a driver if an object ahead is traveling at less than 50% of the driver's car's current speed. This warning mechanism is also a function of the current distance. It is questionable if this application should be used during passing maneuvers on two-way roads.

The unit is reset to the disabled state by tapping the brake, engaging the clutch, or if the speed falls below 35 mph. The two options TAILGATE and OBSTACLE ALERT remain enabled regardless of the mode the cruise control is in. A statechart representation of the behavioral model in shown in Figure 5.

The control unit also obtains data from the internal sensors. The vision sensors are typically chosen to be radar units as shown in Figure 4. The data arrives according to a specified rate given by a specific protocol, e.g., a CAN bus used by a lot by automobile manufacturers today. From this data, information about the distance and relative speeds of the surrounding


Autonomous Intelligent Cruise Control
power on
AICC Off
in [ON/OFF] & Speed >35mph
radar malfunctioning / display error
in [ON/OFF] / turn off cruise light
Initialization Do: Wanted_Speed = 0 Check Radars
radar fine/ turnon cruise light
AICC On
Autonomous Cruise Control On
Intelligent Cruise Control
Operation
Data Handling
Update Data
Tailgate Detector Mode
in[TAILGATE ALERT]
Tailgate Alert
Obstacle Warning Mode
in[OBSTACLE ALERT]
Obstacle Alert
Update Data
Safe_Speed < Curr_Speed/ Curr_Speed = Safe_Speed
AQUIRE Do: update readings for rear & front radar (safe_dist_f,safe_dist_r)
TIMER Do: wait for time interval
Safe_Speed > Curr_Speed & Safe_Speed >= Wanted_Speed/ Curr_Speed = Wanted_Speed
Safe_Speed > Curr_Speed & Safe_Speed > Curr_Speed & Safe_Speed < Wanted_Speed/ Curr_Speed = Safe_Speed
Operation
SLEEP DO: wait for button
in [RESUME/ACC & FORWARD & Wanted_Speed>35]/ Curr_Speed = Wanted_Speed OR in [COAST & FORWARD & Speed>35] / Wanted_Speed = Speed Curr_Speed = Speed
in [BRAKE | !FORWARD | CLUTCH | Speed=<35]
HOLD SPEED DO: keep throttle setting
Speed > Curr_Speed + 2 / decrease throttle
Speed < Curr_Speed - 2 / increase throttle
in [COAST] / Wanted_Speed --
in [RESUME/ACC] / Wanted_Speed++

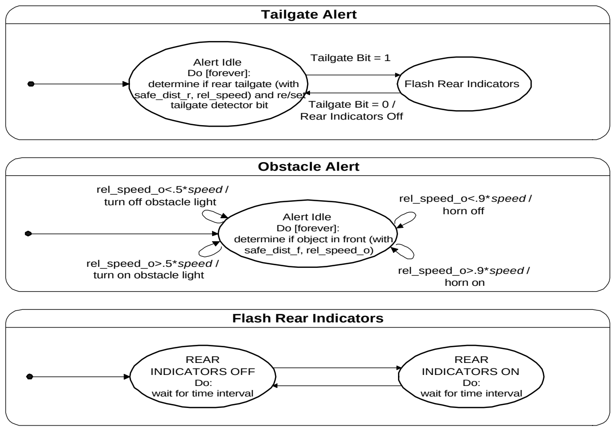


Figure 5. The Equivalent Statechart Representation

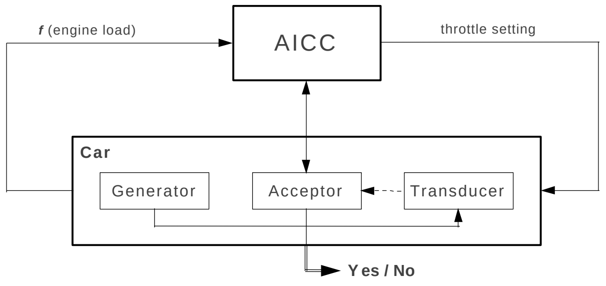


Figure 6. AICC with Experimental Frame

vehicles can be obtained. Other information can be retrieved from the speed sensor or sensors measuring the momentum of the car.

6. Experimental conditions

In our methodology, system models are verified using simulation. Experimental frames [16,21] are employed to define circumstances under which the model is simulated and observed. An experimental frame consists of a generator, an acceptor and a transducer. It is a means of instrumenting a simulation experiment with a mechanism that a) subjects the system's model to the simulated effects of the environment on the system (this is handled by a generator), b) collects and monitors the effects of the system on its environment by observing the system's model's outputs (managed by a transducer), and c) controls the simulation experiment through an acceptor module. For a schematic representation, please refer to Figure 6.

Figure 6 illustrates how the design is tested for its basic function of maintaining the speed within limits assuming that not interrupt events are generated (that is, no event that would disengage the AICC occurs). The car can be viewed as a environment that has various characteristics depending on the model. It is represented as an experimental frame. This means that the output function of the generator corresponds to an engine load per specific model specifications. The response of the AICC unit in form of the throttle setting is then evaluated by the transducer. Finally, to observe the behavior in the different states of the cruise control, e.g., OFF or CRUISE, and to test for correct state transitions, the acceptor is be used.

The AICC simulation model was developed using an house discrete event simulator DEVS-Java [21]. The model was decomposed in a manner that corresponds to the functional decomposition of the real AICC. A diagram of the model is shown in Figure 7. After building and testing the atomic models or components the model was then extended to the final coupled model. It consists of a state manager which keeps track of the state of the cruise control which is mostly influenced by the user, the data manager which obtains data from the functions and the sensors and distributes it to the components that request it, and finally the functions which each solve different calculations.

Since this model is being implemented on a single processor environment (Motorola 68HC11EVB) an additional component, the scheduler, was added to the AICC. This component was necessary to eliminate concurrent execution that can be achieved during the simulation, but which cannot be realized in a single processor implementation. We thus obtained a valid model which can be used to resolve timing and functionality issues, i.e. to determine the bottlenecks of the system.

Along with the model a general user interface (GUI) was developed to test the model of the AICC. It provides an interface which is common to a car. While the simulation is running the user can manipulate cruise control buttons, speed of the car with the cruise control and a profile of the car ahead with this interface. This puts the AICC in a closed-loop environment where the GUI is coupled to the model of an actual car.

7. Conclusions

This paper summarizes our current work to develop a testbed for the model-based codesign methodology. The autonomous, intelligent cruise control module is being realized using the techniques presented above. At the high level the system will be tested according to the specified requirements and then iteratively refined down to the lowest level where the design decisions about the implementation in hardware or software will be made.

Clearly, the technology assignment phase remains a critical issue. As we have pointed out in [1], technology assignment is not as limited as a target architecture bound partitioning scheme used in most existing systems today. This is because the implementation independence of the design persists up until this very assignment step. The components of the validated system model can be bound to modules of possibly different technology.

The interfaces and their synthesis are again a central point of technology assignment. In fact, the dual steps of partitioning and integration is now replaced by a stepwise refinement procedure based on an abstract behavioral model and the interface synthesis step. The problem as such has not, however, become easier. Interfaces are generated based on component interrelations derived from the refined model. Depending on the technology choice,

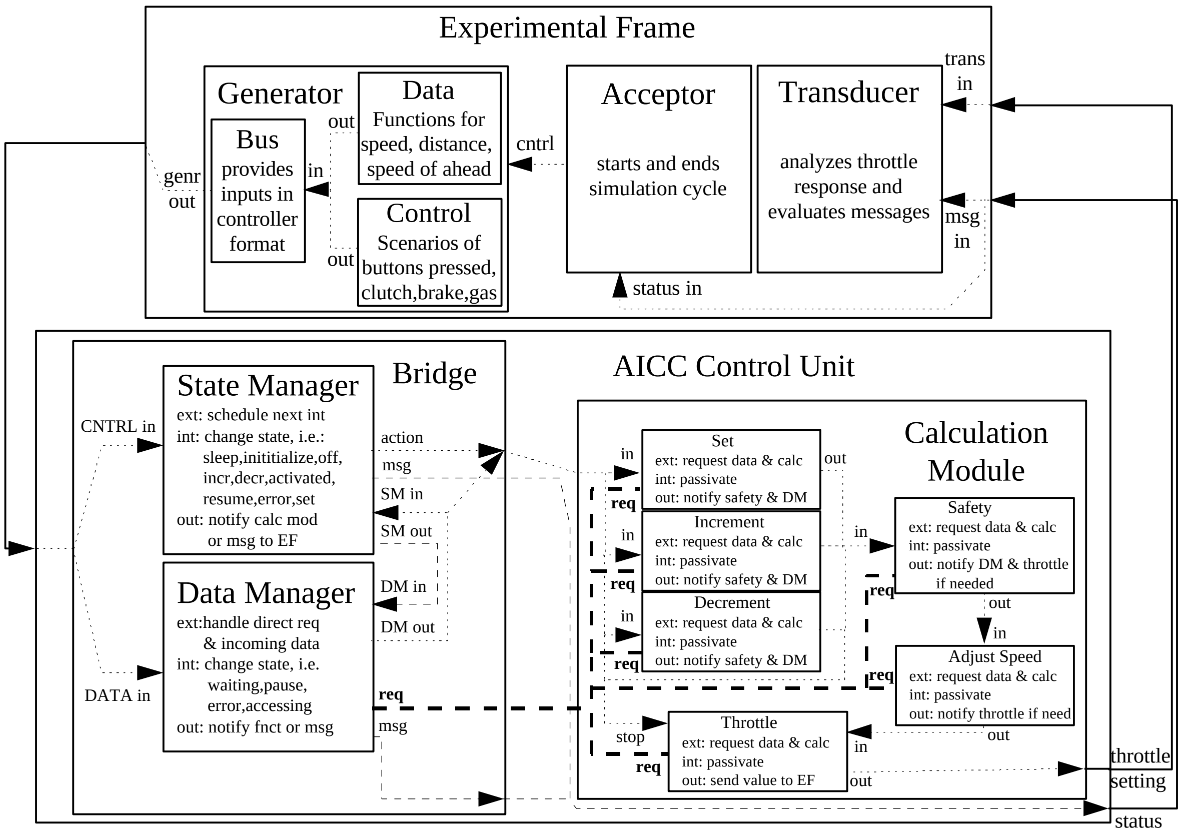


Figure 7. Diagram of the AICC Simulation Model

either signal exchange, interrupt, or other synchronization means are chosen. In our proposed codesign methodology, alternative designs can be evaluated with respect to various criteria, e.g., the allocation (binding) of behavioral models or functionality to action modules (HW, SW, interfaces). This assignment phase is guided by the performance estimation results obtained in the simulation step.

We are currently developing a model-based representation of the AICC module and its various subcomponents. A set of simulation experiments for virtual prototyping and the physical realization of the AICC is being used to validate its specifications. We are also designing backtracking search-based algorithms to facilitate the technology assignment process.

Acknowledgments

This work has been supported by Siemens AG, Central Research and Development Laboratories, Munich, Germany.

References

[1] K. Buchenrieder and J.W. Rozenblit. Codesign: An Overview. In J.W. Rozenblit and K. Buchenrieder (Eds) *Codesign: Computer-Aided Software/Hardware Engineering*, 1-16, IEEE Press, 1994.

[2] R. Bündgen and W. Küchlin. Term Rewriting for Hardware and Software Design. In J.W. Rozenblit and K. Buchenrieder (Eds) Codesign: Computer-Aided Software/Hardware Engineering, 19-40, IEEE Press, 1994.

[3] P. Carrea, A. Saroldi. A Prototype Vehicle for Integration of Driver Support Functions: The Alert Project. *Proceedings of the First World Congress on Applications of Transport Telematics and Intelligent Vehicle Highway Systems*, Vol.4, 2209-15, Paris, France, December 1994.

[4] K. Ehlers. Das Automobil - eine beispielhafte Anwendung von Mikroelektronik.. *Impulse,* No.9, VOLKSWAGEN AG, 5-12, Wolfsburg, Germany, December 1991.

[5] J. Forrest. Implementation Independent Descriptions Using Object-Oriented Approach.. In J.W. Rozenblit and K. (Eds) *Codesign: Computer-Aided Software/Hardware Engineering*, IEEE Press, 1994.

[6] F. Gnavi, R. Risser, M. Carrara. Result of the Application of the PROMETHEUS Traffic Safety Check-List for the Safety Evaluation of the Autonomous Intelligent Cruise Control. *Proceedings of the First World Congress on Applications of Transport Telematics and Intelligent Vehicle Highway Systems*, Vol.4, 2192-9, Paris, France, December 1994.

[7] R.K. Gupta and G. De Micheli, Hardware-Software Cosynthesis for Digital Systems, *IEEE Design and Test of Computers*, 10(3), 29-41, 1993.

[8] R. K. Gupta, N.C. Coelho Jr., and G. De Micheli. Program Implementation Schemes for Hardware-Software Systems, *IEEE Computer*, 27(1), 48-55, 1994.

[9] T.B. Ismail and A.A. Jerraya. Synthesis Steps and Design Models Codesign. *IEEE Computer*, 28(2), 44-53, February 1995.

[10] A. Kalavade and E.A. Lee. A Hardware-Software Codesign Methodology for DSP Applications. *IEEE Design and Test Computers*, 10(3), 16-28, 1993.

[11] A. Kern and A. Bazevicius. A Concurrent Hardware and Software Design Environment. *VLSI Systems Design*, 34-40, August 1994.

[12] S. Kumar, J.H. Aylor, B.W. Johnson, Wm. A. Wulf, and R. D. Williams. An Abstract Hardware/Software Model For Easy Performance Evaluation. *Proceedings of the 1995 IEEE Symposium and Workshop on Engineering of Computer Based Systems*, 299-306, Tucson, AZ, March 1995.

[13] A. Puasteri, A. Streit, J. Gardener. Inertial navigation and GPS compared for automatic vehicle locating systems. *ISATA Proceedings*, 901-5, Aachen, Germany, 1993.

[14] J.W. Rozenblit and K. Buchenrieder. *Codesign: Computer- Software/Hardware Engineering*. IEEE Press, 1994.

[15] J.W. Rozenblit and J. Hu. Integrated Knowledge Representation Management in Simulation Based Design Generation, *IMACS Journal of Mathematics and Computers in Simulation*, 34(3-4), 262-282, 1993.

[16] J.W. Rozenblit, J.W., Experimental Frame Specification Methodology for Hierarchical Simulation Modeling, *International Journal of General Systems*, 19(3), 317-336, 1991.

[17] J.W. Rozenblit, J.W. and Y.M. Huang, Rule-Based Generation of Model Structures in Multifaceted Modeling and System Design, *ORSA Journal on Computing*, 3(4), 330-344, 1991.

[18] J. Rumbaugh. *Object-Oriented Modeling and Design*. Prentice Hall, 1991.

[19] J. Staunstrup. Towards a Common Model of Software and Hardware Components. In J.W. Rozenblit and K. Buchenrieder (Eds) *Codesign: Computer-Aided Software/Hardware Engineering*, 117-127, IEEE Press, 1994.

[20] P. A. Subrahmanayam, Hardware-Software Codesign: Cautious Optimism for the future, *IEEE Computer*, 26(1), 84, 1993.

[21] B.P. Zeigler. *Object Oriented Simulation with Hierarchical Models*, Copyright by Author, 1995.