Modeling and Analysis of Real Time Control Systems: A Cruise Control System Case Study

Anthony Spiteri Staines
University of Malta
Malta

1. Introduction

Real time control and embedded systems provide important services such as manufacturing control, temperature control, cruise control in cars and planes, monitoring and regulation of various parameters.

The importance of these systems is often overlooked and only when failure occurs is that one notices them. The user has limited control over the operations, which are normally managed by a real time controller. Ideally the user should be presented with predictable behaviour. These systems differ greatly in complexity when compared with other traditional software applications. Micro processors are produced and used in embedded applications. Given the ever increasing integration of dedicated components new issues are created. Embedded real time systems need to be produced at a lower cost but simultaneously quality, reliability and safeness have to be increased. The design time should be reduced.

In some embedded real time environments the computing element is a component within the system. The i) operational environment, ii) performance and iii) interfacing capabilities, have a great influence on the system. Safety and failsafe mechanisms need considerable attention at the analysis and design stages. Performance criteria imply understanding how fast a system reacts to events. Response speeds are time critical, varying in range from a few milliseconds to seconds. Performance estimation is more difficult if there are asynchronous tasks which have the possibility of different order execution. For control systems complex control law computations might be required, where the sampling period needs accurate estimation (Liu, 2000).

The design of i) hardware parts, ii) software and the iii) actual system are separate tasks requiring proper staged coordination. The importance of the analysis and design stages is sometimes neglected. The implementation of the physical components should be left to the end. Several advances in modern programming languages and real time programming serve to simplify coding. Hardware and software technologies available at the time along with the temporal requirements and safeness are important aspects (Gomaa, 1996; Williams, 2006).

In literature we are presented with many methods that serve for modeling and a better comprehension of these types systems. Unfortunately there is no single method or approach that caters for all the issues involved. Each technique, notation or method has its own special capability being specific for certain problems. To this end a loosely structured

approach is suggested. Different techniques and methods should be considered and used according to their relevance. In this work a case study of a typical cruise control system is presented, illustrating these concepts. The model is shown using different notations, offering different views. A place transition Petri net models the dynamic behaviour of the system. The Petri net is validated using invariant theory. It is converted to a task graph. Finally it is explained how to optimize and estimate the system's performance.

2. Analysis and Design Implications

Real time systems have special needs that necessitate the employment of special analysis and design methods. These methods differ from traditional software engineering approaches. In traditional software i) good software, ii) correct specifications, iii) modularization are important aspects. Real time concepts add on these requirements other concerns such as: i) correct timing, ii) safeness, iii) well identified state space and iv) viewpoints for different stakeholders.

Various software methods, methodologies and notations, along with all the formal notations, have been created to deal with these issues like i) ambivalence and ambiguity, ii) complexity issues. To make things worse, these systems have both i) functional and ii) non-functional requirements where the system is composed of i) functional specifications, ii) performance specifications, iii) interfaces and iv) constraints. Requirements engineering or requirements elicitation serve to deal with the main problems associated with the i) analysis and design stages of software and hardware development. Requirements engineering is important for hard systems, mission critical and embedded technologies. Requirements engineering tries to bridge the gap between the desired system and the real world. Requirements elicitation is not limited to the use of a particular method, methodology or notation.. Using modern case tools, the viewpoints of different stakeholders can thus be met. It is possible to develop very good system notations representing what is required. Use of schematic diagrams, animation prototyping, block diagrams (Maruyama, 2001) all make sense. Various formalisms like logics, temporal logics, abstract state machines, automata, Petri nets, calculus, calculus of communicating systems (CCS), communicating sequential processes (CSP), algebras and process algebras, formal languages like Z, Vienna development method (VDM), B, Haskell, language of temporal ordering specifications (LOTOS) and task graphs etc. have been created and used to express different views and aspects of real time design process. Formal methods definitely help towards producing better models because in the design process more thinking and reasoning is applied. However they normally focus on specific aspects. CSPs and CCS focus on communication issues, Z and VDM are used to represent the system using schemas. One problem is that most formal methods do not offer proper visualization. Some formal methods have limited CASE tool support. Formal representation can be difficult to understand, time consuming to produce and to amend.

Structured real time methods and notations are based on visual diagrams. Some examples are: Controlled Requirements Expression (CORE) (Mullery, 1979), Hierarchical Object Oriented Design (HOOD), Jackson System Design (JSD), Modular Approach To Software Construction Operation and Test (MASCOT), Design Approach for Real Time (DARTS), Concurrent Approach for Real Time (CODARTS), Real Time Object Oriented Modeling (ROOM), the Unified Modeling Language (UML)and UML-Real Time (UML-RT) (Bennett et

al., 2005; Cooling, 1995; Gomaa, 1996, 2001; Roques, 2005). All these serve to capture different system properties in an informal or semi-formal approach. The UML contains a repository of some notations that are common to most of these methods. Research has been devoted to formalize these diagrams (Bennett et al., 2005; Saldhana et al., 2001).
For designing real time embedded systems a loosely coupled approach is suggested and presented in Figure 1. Informal, semi formal and formal techniques are used at different stages. To start off, informal techniques are used and as more understanding of the system is gained it is possible to use formal techniques.
I) Sources of information are use cases, case tools etc. II) for initial requirements expression information can be collected using different methods or notations e.g. JSP, UML activity diagrams, block diagrams, control flow diagrams etc. III) the requirements obtained in ii) are developed further. IV) Models obtained in ii) and iii) are represented formally using automata, Petri nets, Control flow graphs, etc. The models are analyzed and checked for i) consistency, ii) correctness, iii) completeness , iv) states and v) timing issues.
Figure 1 indicates the complete requirements engineering process which is suited for developing embedded applications. At the initial requirements expression stage II it is more convenient to use informal notations. Simple diagrams and notations can be selected for use at stage I and II. Stages II and III can be combined. Any diagram from UML, JSD, MASCOT, ROOM, DARTS etc. are all valid. Ideally all the models used should be convenient and easy to read by both the engineer and the system stakeholders. At the detailed requirements expression stage III structured techniques and notations are useful. At the model analysis stage IV special checking rules are needed as different issues and problems might arise. Formal models guarantee error free development, it is better to use them at stage IV.

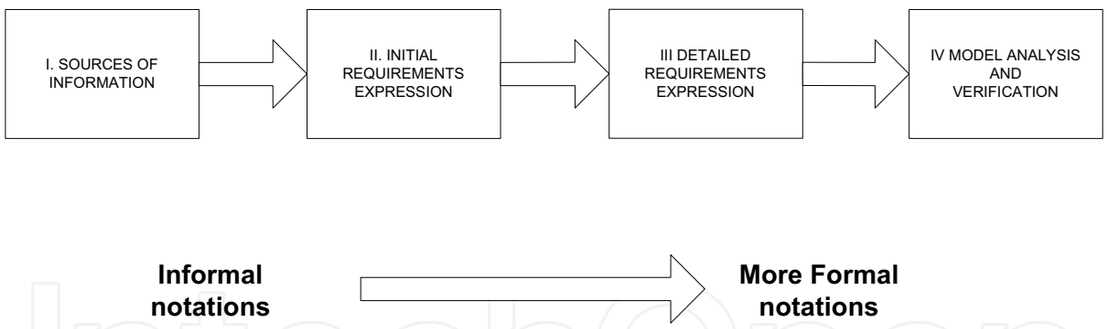

Fig. 1. Semi Structured Approach for Real Time Control Systems Design

For Stage IV Petri nets and task graphs are used. Petri nets are a convenient formalism for behavior modeling, experimentation, visualization and reasoning about real time system properties. They have over three decades of coverage. Petri nets support concurrency, synchronization and resource sharing, formally and diagrammatically. They share important similarities with UML activity diagrams and other notations. E.g. UML 2 activity diagrams are based on Petri net semantics according to the UML. Fundamental modeling concept Petri net diagrams are used to model software functioning at a high level. Higher order nets like colored Petri nets have been widely used to represent and analyze communication protocols and networking problems. Unlike other formalisms Petri nets have a sort of dual identity in the sense that they can be represented using formal languages and also graphical representation. Many UML diagrams like sequence, activity and state

charts all have been successfully translated into Petri nets. Both the structural and dynamic properties of Petri nets can be represented and analyzed. There are several classes of Petri nets ranging from Elementary nets to Object oriented nets and Colored Petri nets. Various CASE tools support Petri net modeling e.g. CPN Tools, HPsim, ExSpect, etc. Petri net structures can be supported or translated into other formalisms. Simple place transition Petri nets are easily converted into task graphs for other forms of analysis and optimization (Brusey & Mc Farlane, 2005; Cortes et al., 1999; Desel & Kindler, 2001; Hanzakel, 1997; Jensen & Rozenberg, 1991; Saldhana et al., 1998; Van Hee, 1991; Van Hee et al., 1994; Yamalidou et al., 1996).

3. Cruise Control System Case Study

3.1 Basic Description- Sources of Information Stage

A modern cruise control system can be classified as digital control. This type of system is characterized as an embedded real time system having a number of sensors and actuators where the actual real time system functions as a digital controller. In simple terms there is a 'read in' of values from the input sensors that depicts the recurrent system state. This is compared to the desired state or reference state and a computation is performed to obtain the 'adjustment value'. This is known as a control law computation which is also one of the most time consuming tasks, carried out after reading-in the sensor values. Behavior is predictable. A typical cruise control system is composed of several components or classes interacting amongst one another in real time. Some components have high computational requirements. Communication between the 'actors' could require the support of adequate protocols and communication channels. Some parts of this system clearly exhibit closed loop highly cyclical behavior typical of real time conrollers. In the cruise control, user data and sensor data is read in. The input data is compared with the desired speed. This comparison is used to compute the output adjustment value for the actuator. The actuator automatically performs the adjustment. This real time system also functions as a controller. Two types of tasks can be identified i) periodic tasks and ii) user initiated tasks. In a system like cruise control, these two types of tasks form part of the main activities. The the basic algorithm for the cruise control system is given below (Gomaa, 1996, 2001; Liu, 2000).

Set Timer to interrupt periodically in a period (T)
at each interrupt do
1) Sensor Scan process (
GPS,UI,Brake,Accel,Engine)
2) Get current speed
3) Compute control values
4) Update parameters
5) Send adjustment value to throttle
enddo

Steps 3 and 4 have the most significant time requirements. This is because the final computation is based on these values. The reading in sensor inputs tasks 1 and 2 are composed of subtasks. They can be carried out in any order. It is possible to execute these tasks concurrently. Tasks 3 and 4 have precedence constraints requiring certain ordering.

The current speed measured from the wheel rotation is compared with the desired speed and the data from the other sensors. The computed adjustment value is sent to the throttle actuator. Normally this would be i) reduce speed or ii) increment speed or iii) maintain current speed. Tasks 4 and 5 can be executed concurrently. The system behavior can be classified as deterministic, exhibiting a repeated pattern behavior. Sensor data is read to obtain accurate estimates of state variables to be monitored and controlled. Input values are used to compute an adjustment value. Parts of the system can be scheduled differently.

3.2 Initial requirements expression and Representation Stages

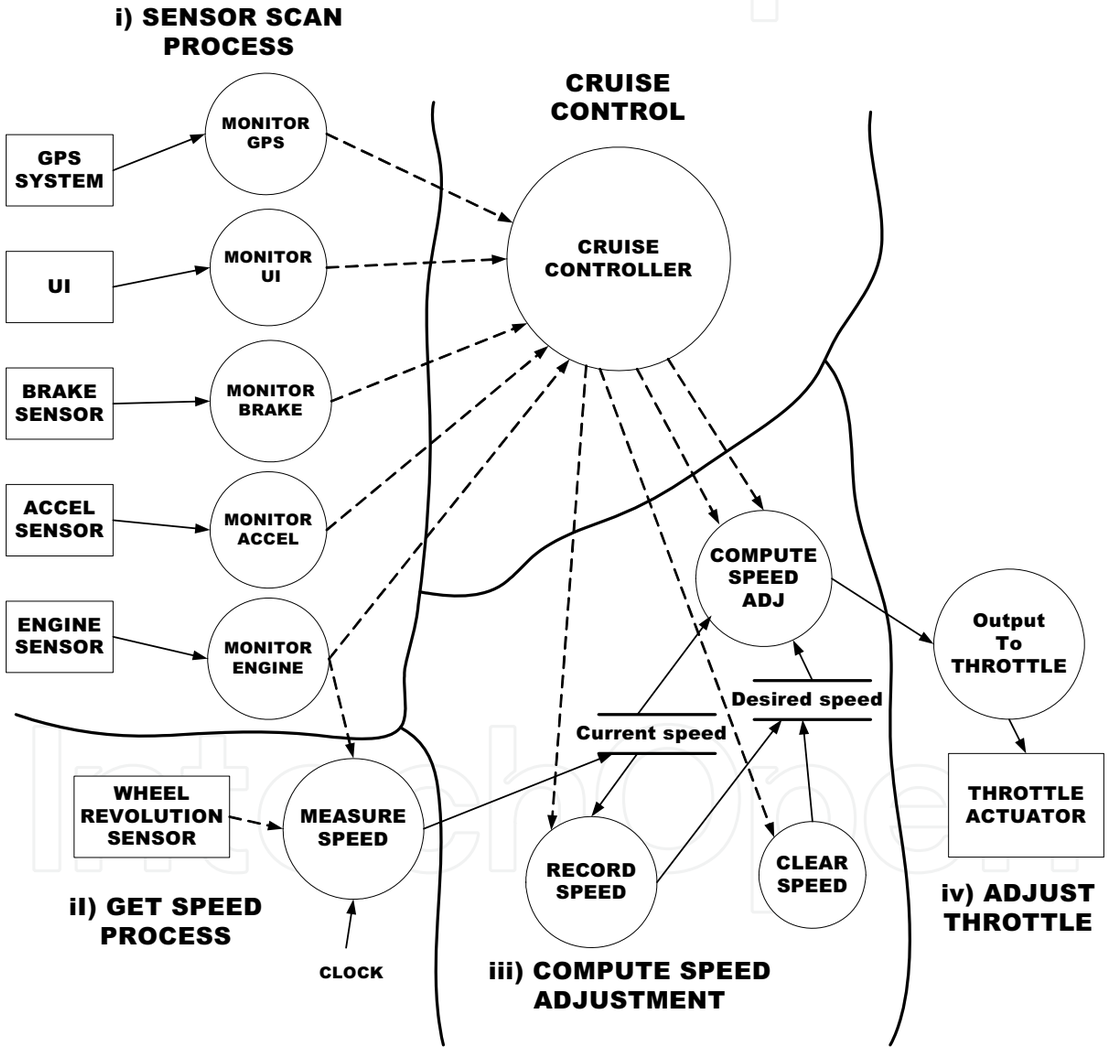

Fig. 2. Cruise Control Flow Data Flow Diagram adapted from (Kramer & Magee, 1997)

The initial and detailed requirements expression stages II and III are combined. The diagrams used are from the previously suggested methods.

The diagram in Figure 2 (Kramer & Magee, 1997) illustrates the main cruise control system events. This diagram is commonly used in real time structured analysis and design methods like JSD and DARTS (Gomaa, 1996). Extensions to data flow diagrams are used to add details for event flows and control transformations like discrete, continuous, triggered, enable/disable etc. The diagram has been partitioned into four main sections to clearly illustrate all four main tasks. All activities are controlled and synchronized by the cruise controller.

Figure 2 depicts the top-level network diagram for the cruise control using MASCOT notation. This diagram can be used to obtain a full system template with bindings and interfaces e.g.

Server Disp out:Digital_out(ow=USER..op);

Server Spsensor-in: Analog-in1(s1w=CC1.s1p); etc.

The subsystem components e.g. speed control, user interface, etc. can also be identified. Component coupling is rigorously enforced. The diagram can be decomposed further. UML interaction diagrams can describe the communication processes for the cruise control.

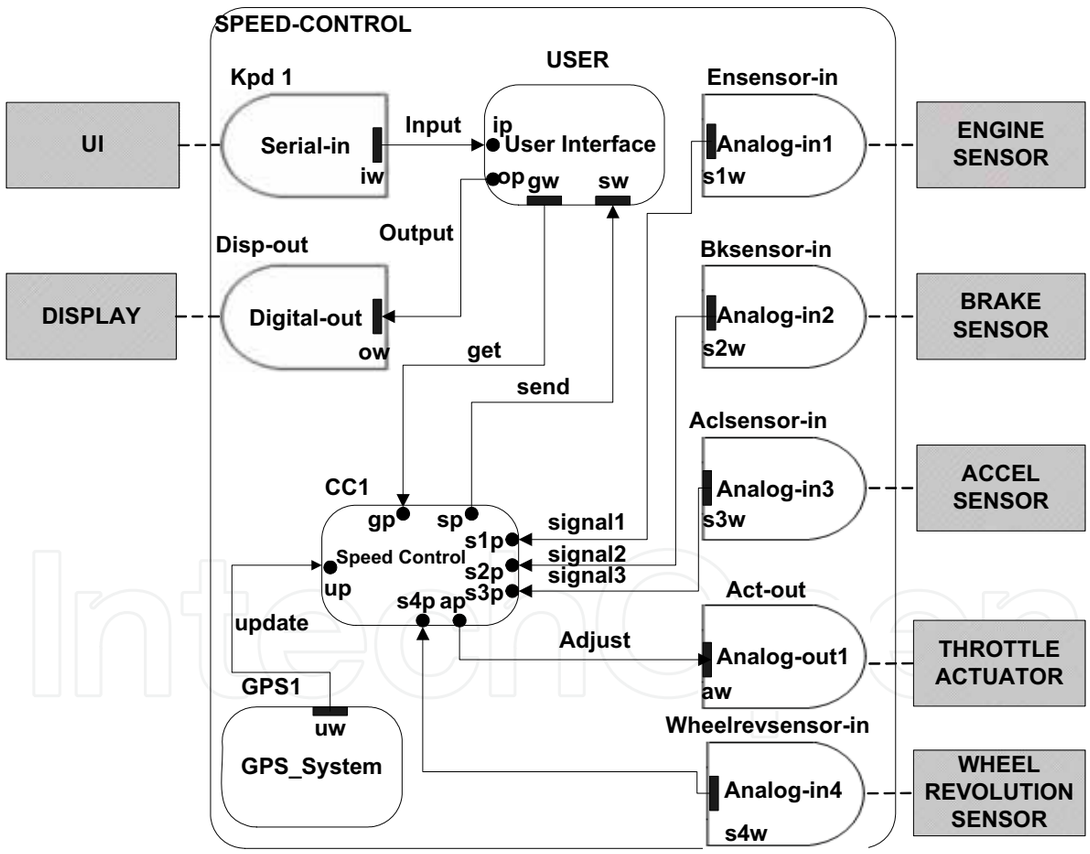

Fig. 3. Cruise System MASCOT Network Diagram

The cruise control System UML activity diagram is presented in Figure 4. This shows all the tasks and their order. The activity diagram can be converted into a Petri net.

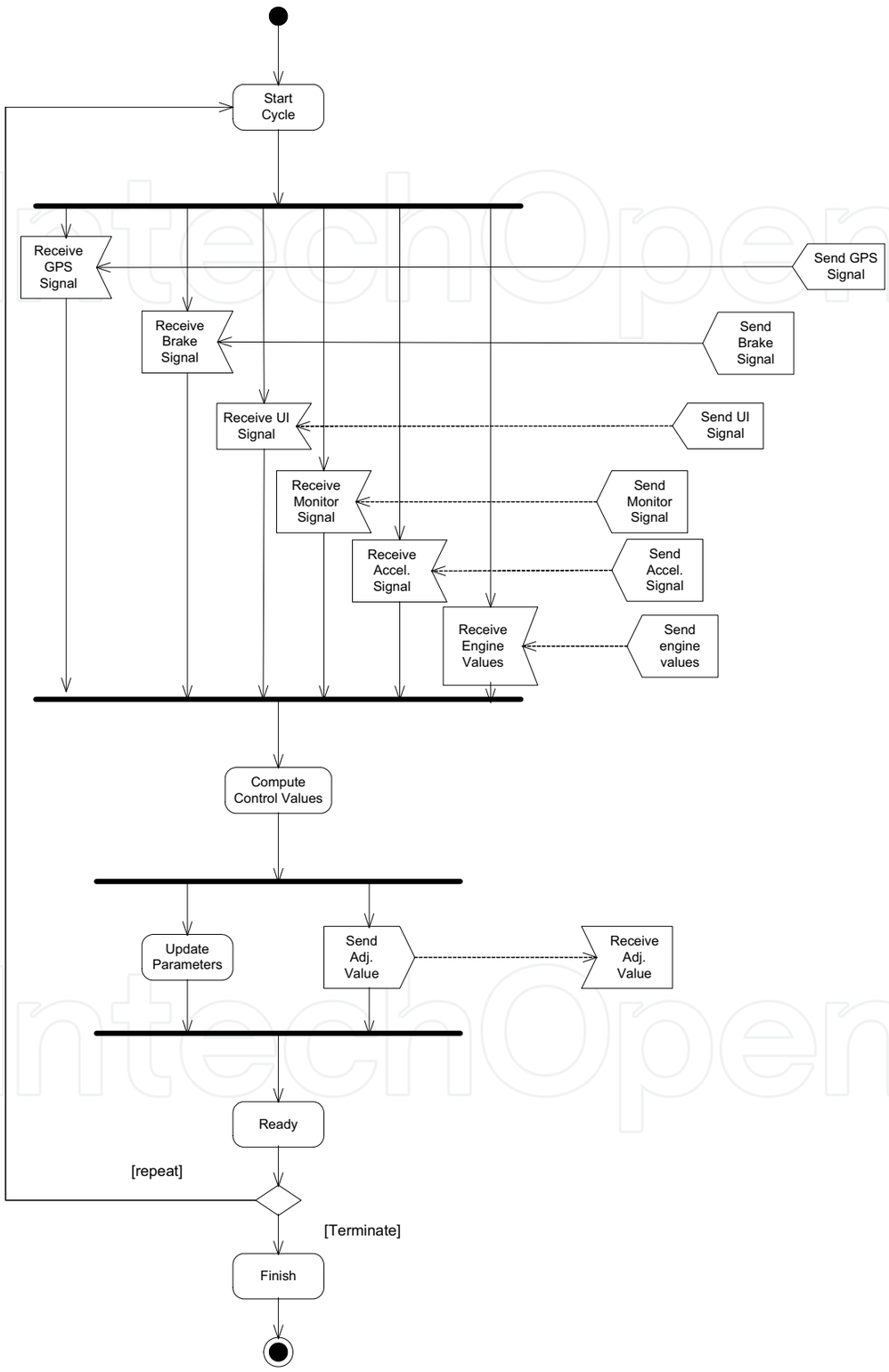

Fig. 4. Cruise Control System UML Activity Diagram

4. Model Analysis and Verification Stage - Petri Net and Task Graph Modeling

4.1 Constructing the Petri Net

As stated for analysis of the models Petri net structures and task graphs are used. Constructing the Petri net is a relatively simple process. The algorithm in 3.1 or the UML activity diagram shown in Figure 4 can be used. The following steps explain the conversion process: i) add a dummy source transition (node) at the top ii) add a dummy sink transition at the bottom (end) iii) the tasks in the algorithm are placed in sequence between the source and the sink node. Transitions for parallel tasks are placed next to each other, iv) places are added to join the transitions. For parallel processing a fork point and a join point is used to join the input and output of the concurrent tasks.

Practically speaking the Petri net represents the possible task execution sequence and it is similar to a task graph (Abdeddaim et al., 2003; Hanzakel, 1997). The Petri net is both a visual and formal executable specification that is easy to understand.

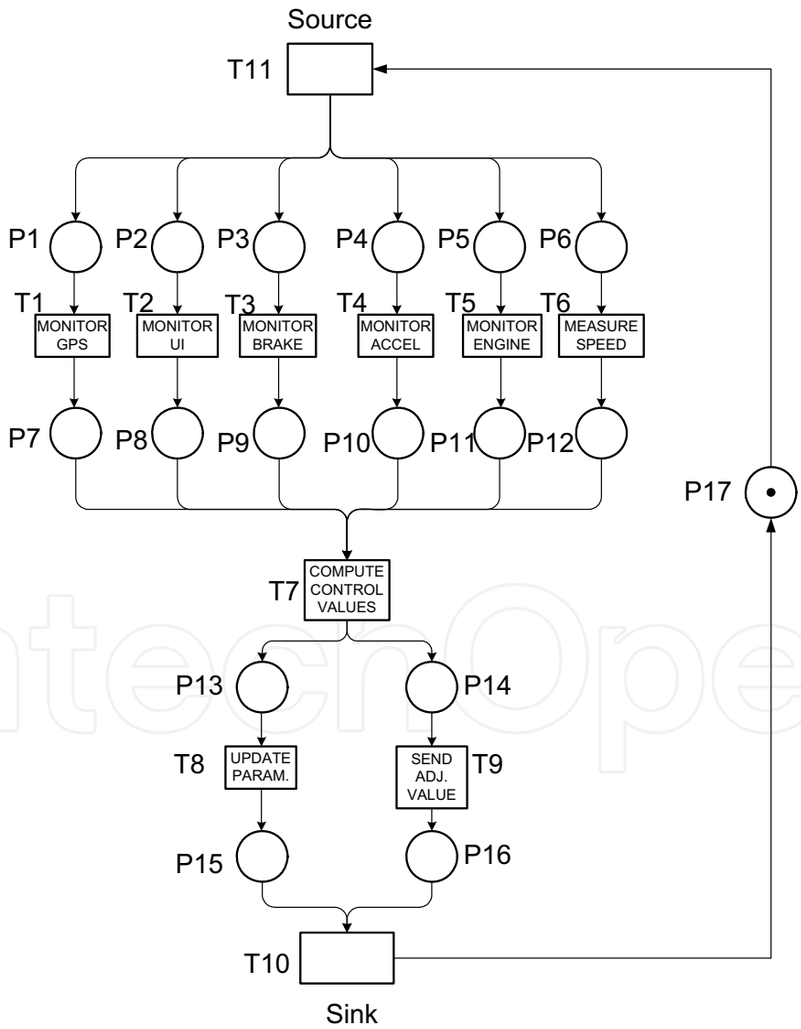

Fig. 5. Cruise Control Initial Petri Net

4.2 Task Parallelism

Parallelism (Hanzakel, 1997; Liu, 2000) implies detecting computations that can be carried out in parallel. If more than one processor is available, tasks can be executed in parallel. Petri nets and task graphs expand this possibility. E.g. Figure 5 shows that tasks T1..T6 and T8,T9 can be executed in parallel.

4.3 Cruise Control Directed Graph

The cyclic directed graph similar to a task graph for the cruise control system is simply obtained by ignoring all the places in the Petri net and replacing the transitions with nodes. This is possible because each place holds exactly one token i.e. it is a 1-safe Petri net. This graph can be reduced into a directed acyclical graph (DAG) by removing e17.

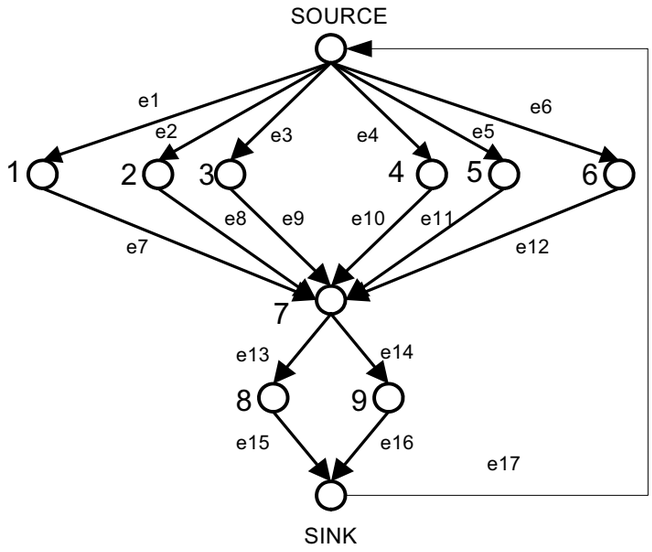

Fig. 6. Cruise Control Directed Graph

4.4 Redrawing the Graph for Two Available Processors

The tasks or transitions in Figure 5 would have to be executed sequentially if a single processor is assumed to be available. If more than one processor is available the whole scenario will change. The vertices in Figure 6 indicate which tasks can be carried out in parallel. It is evident that if all T1..T6 had to be executed completely in parallel, six processors are required. Normally we can assume just two processors are available. The directed graph needs to be redesigned to reflect this. The problem is to find an optimum solution to redistribute/schedule concurrent tasks. The following algorithm can be used for this purpose.

Identify all critical tasks
Identify all parallel tasks
Add tasks in order to a processor

If (critical task) then
    If (time (P_a) = time (P_b)) add task to P_a or P_b

If (time (P_a) > time (P_b)) add task to P_a
If (time (P_a) < time (P_b)) add task to P_b
Set time for P_a , P_b = max time

If (parallel task) then
If time (P_a) + newtask < time (P_b) + newtask
Add task to P_a
If time (P_b) + newtask < time (P_a) + newtask
Add task to P_b
If (time (P_a) + newtask = time (P_b) + newtask add task to P_a or P_b

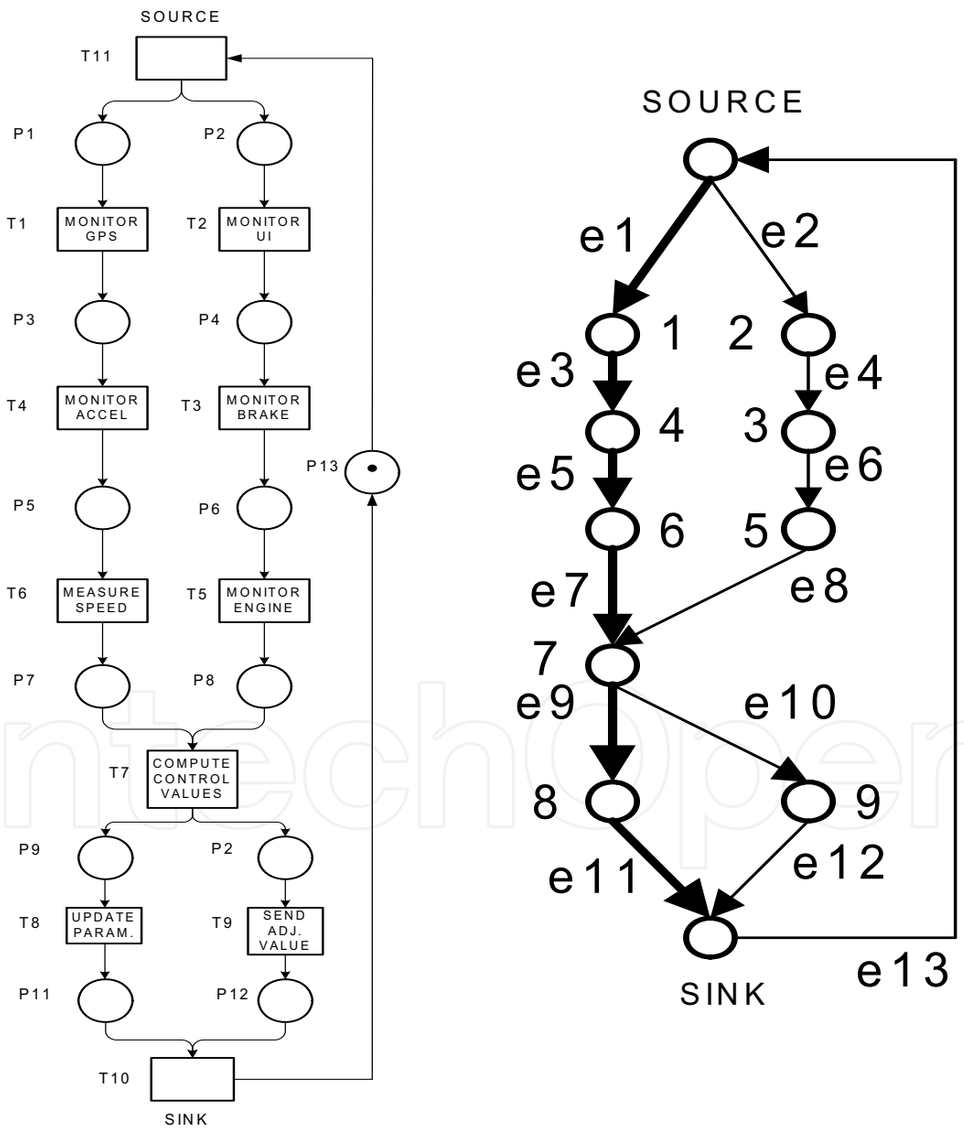

Fig. 7. Cruise Control a) Petri Net and b) Directed Graph for Two Processors

If the given times for tasks are: T1, T6, T9 = 20ms, T2,T5 = 10ms, T3,T4 =15ms , T7= 40ms and T8= 25ms the result is as shown in Figure 7 a) and b). In Figure 7 b) the critical path or longest cycle is shown in bold. This is the sequence of events on processor A, T1-T4-T6-T7-T8. The sequence of events on processor B is T2-T3-T5-T9.

5. Verification of the Petri Nets

5.1 Incidence Matrix Properties

An incidence, flow or change matrix C_{ij} is a special matrix representing the ordered input flows and output flows of the Petri net. For this matrix $1 \le i \le m, 1 \le j \le n$ and C = input flows – output flows. The incidence matrix is important for expressing basic structural properties of the net. A transition might have exactly one input and output flow, if for every row there are non zero values which sum up to zero. I.e. for a row i, $\sum_{k=1}^{n} a_{jk} = 0$.

5.2 Invariants

There are several classes of invariants (Clarisio et al., 2005; Sankaranarayana et al., 2004; Van Hee, 1994, Yamalidou et al., 1996; Zhou & Venkatesh, 1999). Simple linear invariants that interpret the structural and firing properties of the net are used. A vector $v \in Z^m$ is by definition a P-invariant **iff** $v^t \cdot C = 0$ for a given Petri net. For the Petri net $v^t \cdot M = v^t \cdot M' + v^t.C.\vec{s}$, where M = initial marking, M' = next marking, C = incidence matrix and $\vec{s}$ =firing vector. $v^t \cdot M = v^t \cdot M'$ for all reachable markings denoting that the weighted sum of tokens remains constant or unchanged. A vector $y \in Z^m$ is by definition a T-invariant **iff** $Cy = 0$ for a given Petri net denoting a repetitive firing cycle.

Analyzing the Petri nets in Figure 5 and Figure 7a) the following results are obtained. This data was obtained using the Dnanet Petri net tool (Attieh et al., 1995).

```
1 1 0 0 0 0 0 0 0 0 0 0    (main.p1)      1 1 0 0    (main.p1)
0 0 1 0 0 0 0 1 0 0 0 0    (main.p2)      0 0 1 1    (main.p2)
0 0 0 1 0 0 0 0 1 0 0 0    (main.p3)      1 1 0 0    (main.p3)
0 0 0 0 1 0 0 0 0 1 0 0    (main.p4)      0 0 1 1    (main.p4)
0 0 0 0 0 1 0 0 0 0 1 0    (main.p5)      1 1 0 0    (main.p5)
0 0 0 0 0 0 1 0 0 0 0 1    (main.p6)      0 0 1 1    (main.p6)
1 1 0 0 0 0 0 0 0 0 0 0    (main.p7)      1 1 0 0    (main.p7)
0 0 1 0 0 0 0 1 0 0 0 0    (main.p8)      0 0 1 1    (main.p8)
0 0 0 1 0 0 0 0 1 0 0 0    (main.p9)      1 0 1 0    (main.p9)
0 0 0 0 1 0 0 0 0 1 0 0    (main.p10)     0 1 0 1    (main.p10)
0 0 0 0 0 1 0 0 0 0 1 0    (main.p11)     0 1 0 1    (main.p11)
0 0 0 0 0 0 1 0 0 0 0 1    (main.p12)     1 1 1 1    (main.p12)
1 0 1 1 1 1 1 0 0 0 0 0    (main.p13)     1 0 1 0    (main.p13)
0 1 0 0 0 0 0 1 1 1 1 1    (main.p14)
1 0 1 1 1 1 1 0 0 0 0 0    (main.p15)
0 1 0 0 0 0 0 1 1 1 1 1    (main.p16)
1 1 1 1 1 1 1 1 1 1 1 1    (main.p17)
```

Fig. 8. Place Invariants for the Petri Nets in Figure 5 and Figure 7 a)

```
1     (main.t1)
1     (main.t2)
1     (main.t3)
1     (main.t4)
1     (main.t5)
1     (main.t6)
1     (main.t7)
1     (main.t8)
1     (main.t9)
1     (main.t10)
1     (main.t11)
```

Fig. 9. Place Invariants for the Petri Nets in Figure 5 and Figure 7 a)

5.3 Other Behavioral and Structural Properties

The marking graph, consisting of all the possible markings is used for constructing the reachability tree and testing for deadlock.

Invariants are used for further analysis. i) Bounded and conservative behavior is denoted by $\exists v > 0, v^T C = 0$,ii) Repetitive behavior by $\exists y > 0, Cy \geq 0$ and iii) Consistent behavior by $\exists y > 0, Cy = 0$. Other issues like home states, cyclic behavior, deadlock and boundedness can be properly interpreted from the invariant results. Petri net test suites and CASE tools like Dnanet, etc. are useful for further checking. The results of the reachability graphs and invariants are presented in tables 1 and 2.

PETRI NET	REACHABILITY MARKING GRAPH	CONNECTED COMPONENTS	T- INVARIANTS	P-INVARIANTS
Fig. 5	69 unique markings	1 strongly	identical	not identical
Fig. 7a)	21 unique markings	1 strongly	identical	not identical

Table 1. Reachability and Invariant Comparison for Petri Nets in Figure 5 and 7a)

PETRI NET	DEADLOCK POSSIBLE	BOUNDED	CYCLIC BEHAVIOUR	HOME STATES
Fig. 5	NO	YES	YES	YES
Fig. 7a)	NO	YES	YES	YES

Table 2. Other Structural and Behavioural Properties for Petri Nets in Figure 5 and 7a)

5.4 Interpretation of the Petri Net Properties

The Petri net models for the cruise control system have been successfully validated. They are both conflict free, deadlock free and do not have unwanted states. The most strongly connected component is T7. This task is the most critical and important task. The transition invariant analysis shows that even though the algorithm and firing cycle is modified, the basic properties and execution remain unchanged. I.e. both models are formally correct and valid. The structures have a relatively small reachability tree. The two processor Petri net

shown in Figure 7a) has only 21 unique markings compared with the one of Figure 5 having 69. This is indicative that if more processors are introduced, the overall system state space is increased and becomes more difficult to handle. More synchronization overhead is necessary to coordinate and control process communication. On the other hand, by reducing the parallel tasks in the system the complexity is reduced so there is less switching overhead. The number of parallel places or the concurrent tasks in the directed task graph, indicate the total number of processors required for parallel task execution.

6. Task Scheduling and Optimization

6.1 Two Processor Task Scheduling

One difficulty in parallel systems is load balancing. The graphical result for the Petri net in Figure 7a) and task graph in Figure 7b) is shown in Figure 10. This is obtained from the algorithm in section 4.4. Processor I is fully utilized and processor II has a utilization of approximately 46 % only when directly compared with processor II. Processor II is utilized for 55ms only in a 120 ms cycle when Processor I is utilized 100% of the time. Adding another processor can reduce the time of processor I, but the task iii) compute control values, has precedence constraints hence all the preceding tasks must have completed.

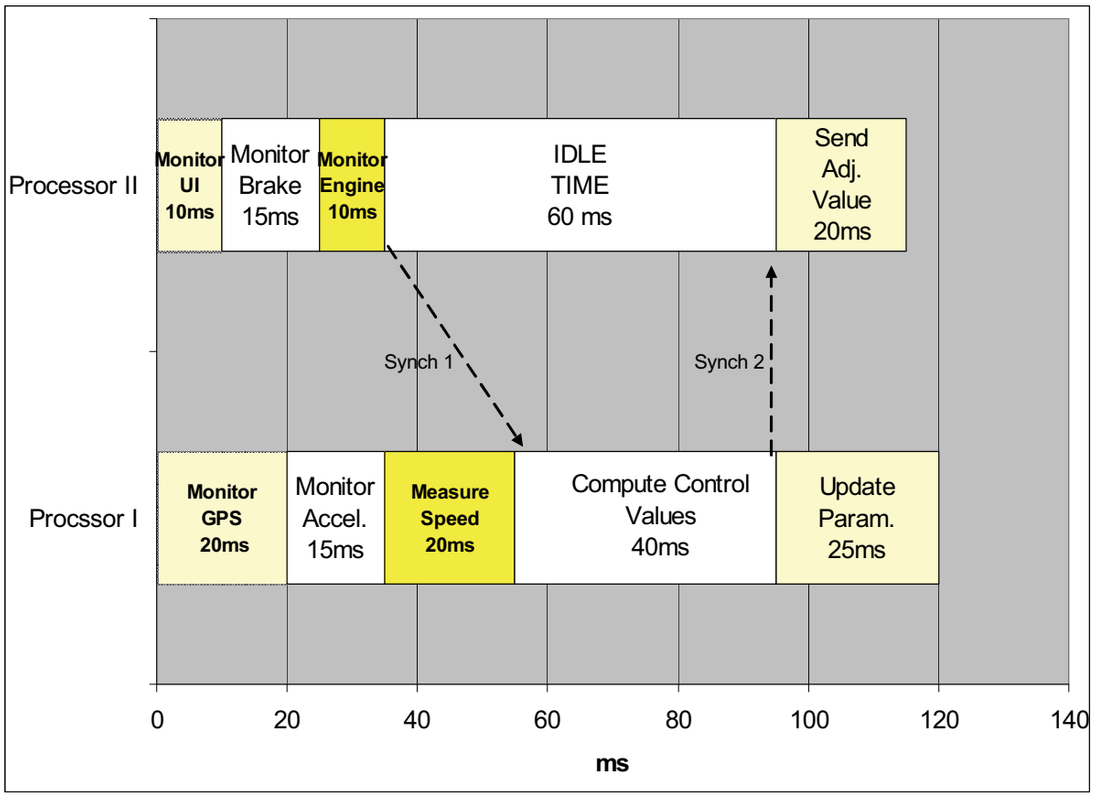

Fig. 10. Task Scheduling for Two Processors

$$utilratio = \frac{T_{p1}}{T_{p2}} \tag{1}$$

Equation (1) defines the processor utilization ratio which is a very simple measure. If Processor I is used for 120ms and Processor II for only 55ms then we get a value of 2.18 which implies that processor I is used 2.18 times more than processor II. I.e. the load is not evenly distributed. The inverse of the utilization ratio yields a value of 0.458 confirming that Processor II is utilized 45.8% of the time that Processor I is utilized. It is not really possible to improve this because there are precedence constraints and communication delays which is normal for multiprocessing.

6.2 Improvement Factor of Two Processors vs a Single Processor

Two or more processors offer an improvement over having just a single processor. The improvement needs to be measurable.

$$I = \frac{T_{Seq} - T_{parallel}}{T_{seq}} \tag{2}$$

Equation (2) defines the improvement factor of parallel processing over sequential processing. I = improvement factor, *Tseq* = total time taken to complete tasks with a single processor. *Tparallel* = total time taken to complete tasks when there is more than one processor. If *Tseq*= 175ms for the model in fig 5. and *Tparallel*= 120ms for fig 7a) then we get I=0.314, i.e. adding one extra processor reduces the time of sequential processing by 31.4%. The improvement that is obtained having two processors is again constrained by precedence constrains and communication or synchronization requirements which cannot normally be changed. It must be pointed out that the improvement depends on the timing of the individual tasks. With other timings it is possible to have completely different results and even no improvement.

7. Conclusion

The main aim of this work has been to present a semi structured approach for the analysis and design of real time embedded and control systems. The main steps involved are i) sources of information, ii) initial requirements expression, iii) detailed requirements expression and iv) model analysis and verification. In this context, the example of a basic cruise control system has been taken. This example has been used to show some different diagrammatic notations and formal modeling using place transition Petri nets and task graphs. It is not possible to represent embedded and control systems without diagrams. There being many methods, it becomes difficult as to which diagrams should be used. In this work different notations from different methods have been used for the initial requirements expression and detailed requirements expression stages, i.e. stage II and III which can be combined. These show different views and capture more information. Ideally it could be better when designing these systems if more than one method is considered. To prove the correctness of system specifications, formal methods have to be used. When there

is a hard real time system, like a cruise control system, it is imperative to have a high degree of confidence that the system specifications and notations are correct. Once this has been done the design stage is better managed. On the other hand the use of formal notations can be time consuming and require expert knowledge.
A cruise control system has been studied and analyzed using different notations. MASCOT, UML activity diagrams etc. have been used initially. Other diagrams from JSD, DARTS, ROOM etc. could have been used. Finally for dynamic modelling a simple Petri net is built. Petri nets are converted into directed task graphs. Petri net theory is used for model optimization and proving the correctness of models. Useful properties like invariants, reachability tree, etc. can be easily obtained for deterministic systems. More detailed analysis based on graph theory, other Petri net properties and simulation techniques can be considered. Colored Petri nets and higher order nets provide other possibilities. The two processor Petri net model can be translated into ladder logic diagrams (LLD) useful for programmable logic controller (PLC) programming.
.If there are other timings or considering other communication and synchronization overheads not shown in this study, it is possible not to have any improvements and a completely different strategy might be required. This is because the actual analysis given in section 6 depends upon the order and timing of the tasks.

8. References

Abdeddaim, Y.; Kerbaa, A. & Maler, O. (2003). Task Graph Scheduling using Timed Automata*, International Parallel and Distributed Processing Symposium (IPDPS'03)*, pp. 237.2, Nice, April 2003, IEEE Computer Society, France

Attieh, A.; Brady, M. & Knottenbelt, W. (1995). DNAnet A Concurrent Systems Modelling Tool User Guide ,*Data Network Architectures Laboratory*, University of Cape Town, South Africa

Bennett, S.; Skelton, J. & Lunn, K. (2005). *Schaums Outline of UML 2nd ed.*, McGraw-Hill, ISBN: 978-0-077-10741-3, Europe

Brusey, J. & Mc Farlane, D. (2005). Designing Communication Protocols for Holonic Control Devices using Elementary Nets*, Second International Conference on Industrial Applications of Holonic and Multi-Agent Systems,* pp. 22-24, ISSN: 0302-9743, Copenhagen, August 2003, Springer-Verlag, Denmark

Clarisio, R.; Rodriguez-Carbonell, E. & Cortadella, J. (2005). Derivation of Non-structutal Invariants of Petri Nets using Abstract Interpretation*, 26th International Conference On Applications and Theory of Petri Nets (ICATPN),*pp. 188-207, ISBN 3-540-26301-2, Miami, June 2005, Florida, USA

Cooling, J. E. (1995). *Software Design for Real-Time Systems*, Chapman & Hall , ISBN: 978-0442311742, London

Cortes, L.A.; Eles, P. & Peng, Z. (1999). A Petri Net based Model for Heterogeneous Embedded Systems, 17*th* *IEEE NORCHIP Conference*, pp. 248-255, Oslo, November 1999, IEEE, Norway

Desel, J. & Kindler E. (2001). Petri Nets and Components Extending the DAWN Approach, *Workshop on Modelling of Objects, Components, and Agents (MOCA'01)*,pp. 21-36, Aarhus University, August 2001, CPN Group, Denmark

Gerhke, T.; Goltz, U. & Wehrheim , H. (1998). The Dynamic Models of UML: Towards a Semantics and its Application in the Development Process, *Technical Report Informatik-Bericht 11/98,* CiteSeerX.psu:10.1.1.43.3437, 1998, University of Hildesheim, Germany

Gomaa, H. (1996). *Software Design Methods for Concurrent and Real-Time Systems*, Addison-Wesley, ISBN: 978-0201525779, USA

Gomaa, H. (2001). *Designing Concurrent, Distributed, and Real-Time Applications with UML*, Addison- Wesley Professional , ISBN: 978-0201657937, USA & Canada

Hanzakel, Z. (1997). *Parallel Algorithms for Distributed Control - A Petri Net Based Approach*, PhD. Thesis, Czech Technical University (CTU), Prague 1997

Jensen, K. & Rozenberg, G. (1991). *High-Level Petri Nets: Theory and Application*, Springer-Verlag, ISBN:3-540-54125-x, London, UK

Kramer, J. & Magee, J. (1997). Exposing the Skeleton in the Coordination Closet, *Proceedings of the Second International Conference on Coordination Languages and Models*, pp. 18-31, ISBN: 3-540-63383-9,Berlin, September 1997, Springer-Verlag, Hiedleberg

Liu, J.W.S. (2000). *Real-Time Systems*, Pretence Hall, ISBN: 978-0130996510, New Jersey

Maruyama, K. (2001). Automated Method-Extraction Refactoring by Using Block-Based Slicing, *Proceedings of the 2001 symposium on Software reusability: putting software reuse in context*, pp. 31-40, ISBN:1-58113-358-8, Toronto, May 2001, ACM, Ontario, Canada

Mullery, G. P. (1979). CORE- A Method for Controlled Requirement Specification, *Proc. of the 4th Int. Conf. on Software Engineering*, pp. 126-135, Munich , September 1979, IEEE Computer Society, Germany

Roques, P. (2005). *UML in Practice: The Art of Modeling Software Systems Demonstrated through Worked Examples and Solutions*, Wiley, ISBN: 978-0-470-84831-9,

Saldhana, J. A.; Shatz, S.M. & Hu, Z. (2001). Formalization of Object Behavior and Interactions From UML Models, *International Journal of Software Engineering and Knowledge Engineering (IJSEKE),* Vol. 11,No 6., May 2001, pp. 31-40, ACM

Sankaranarayana,S.; Simpa, H. & Manna, Z. (2004). Petri Net Analysis using Invariant Generation, *In: Verification: Theory and Practice: Essays Dedicated to Zohar Manna on the Occasion of His 64th Birthday (LNCS)*, Vol. 2772 , pp. 682-701, Springer-Verlag, ISSN:978-3-540-21002-3, Heidelberg Germany

Van Hee, K.M.; Somers, L.J. & Voorhoeve, M. (1991). The EXSPECT Tool, *Proceedings of the 4th International Symposium of VDM Europe on Formal Software Development-Volume I: Conference Contributions - Volume I (LNCS)* , pp. 683-684, ISBN:3-540-54834-3, Springer-Verlag, London UK

Van Hee, K.M. (1994). *Information Systems Engineering A Formal Approach,* Cambridge University Press, ISBN:0-521-45514-6, Cambridge UK

Williams, R. (2006). *Real-Time Systems Development*, Elsevier, ISBN: 978-0-7506-6471-4, United Kingdom

Yamalidou , K. ; Moody, J. , Lemmon, M. & Antsaklis, P. (1996). Feedback Control of Petri Nets Based on Place Invariants, *Automatica (Journal of IFAC),* Vol. 32, no. 1, 1996,pp. 15-28, Pergamon Press, ISSN: 0005-1098, Tarrytown, NY, USA

Zhou, M. & Venkatesh, K. (1999). *Modeling, Simulation and Control of Flexible Manufacturing Systems- A Petri Net Approach*, World Scientific Publishing Company, ISBN: 978-9810230296, Singapore

Recent Advances in Technologies

Edited by Maurizio A Strangio

ISBN 978-953-307-017-9

Hard cover, 636 pages

Publisher InTech

Published online 01, November, 2009

Published in print edition November, 2009

The techniques of computer modelling and simulation are increasingly important in many fields of science since they allow quantitative examination and evaluation of the most complex hypothesis. Furthermore, by taking advantage of the enormous amount of computational resources available on modern computers scientists are able to suggest scenarios and results that are more significant than ever. This book brings together recent work describing novel and advanced modelling and analysis techniques applied to many different research areas.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Anthony Spiteri Staines (2009). Modeling and Analysis of Real Time Control Systems: A Cruise Control System Case Study, Recent Advances in Technologies, Maurizio A Strangio (Ed.), ISBN: 978-953-307-017-9, InTech, Available from: http://www.intechopen.com/books/recent-advances-in-technologies/modeling-and-analysis-of-real-time-control-systems-a-cruise-control-system-case-study

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen