Subject:	Ph.D. dissertation submitted in partial fulfillment of the requirements for obtaining the academic degree *Doctor technicae* (short Dr. techn., in German: Doktor der technischen Wissenschaften) at the *Faculty of Natural Sciences* at the University of Salzburg
Title:	Modeling of Real-Time Software Systems based on Logical Execution Time—TDL Extensions, Simulation, Tools
Ph.D. Candidate:	Andreas Naderlinger
Secondary Grader:	Pieter J. Mosterman Senior Research Scientist, *Design Automation*, MathWorks Adjunct Professor, *School of Computer Science*, McGill University
Grade:	1 (Sehr Gut)

1 Topic

The general area of the dissertation is in the design of embedded systems; physical systems with reactive computational functionality. More specifically, the domain is *digital control systems* more so than digital signal processing. This is an important distinction as the requirements for the different domains can be significantly at odds with one another.

Given the proliferation of embedded computing power in modern engineered systems, this is clearly an area of distinct importance. The work has implications at a software level, system level, and potentially societal level, although the latter probably would require a more substantive treatment of distributed behavior. Still, the challenges met in embedded software, especially in terms of system integration, are critical in enabling the engineering of even today's systems, as evidenced, for example, by the repeated delays in the delivery of the F-22 [5].

Modern design of digital control system (e.g., [4]) follows a layered approach that typically starts off with requirements capture, from this high-level specifications are distilled, at a system level. The high-level specification is then transformed into detailed specifications at a subsystem level, from which the subsystem components can be implemented. Given the extensive use of computation in engineered systems, and because of its versatility as universal *system integrator*, nowadays these subsystems often comprise large amounts of software.

This dissertation addresses part of the apparent impedance mismatch when a (sub)system specification is grafted onto a software implementation: whereas software is designed to implement untimed logic functionality, a substantive part of the (sub)system specification imposes temporal constraints. The mismatch complications are exacerbated by the typical software stack put in place to efficiently exploit a platform-based design approach [3]. Whereas the

underlying hardware adheres strictly to physical timing, the typical tasked software abstraction dissociates the physical time of the hardware from the application software that implements the (sub)system specification.

A general time-triggered approach reinstates time at the application level by compartmentalizing logic tasks into temporal segments of physical time during which they can be active. In this dissertation, the time-triggered notion as developed for network communication has been adopted for execution based on previous work initiated by Thomas Henzinger *et al.* The essential aspect of the methodology to bridge the tasked software abstraction separates the *execution* time of a software task from its *scheduled* time of completion. The obvious benefit of this is determinism and, because of the separation of concerns, the ease and, consequently, correctness of design.

2 Summary of the Dissertation

In this dissertation, the time triggered execution is developed as an effective integrated element of Model-Based Design [1]. An intermediate layer between timing *requirements* and logic *specifications* of subsystems is introduced by means of the Timing Definition Language (TDL). This layer is a formalization of temporal behavior of modules that allow run-time task switching. The timing effects, such as introduced delays, are automatically derived and functional simulation of the temporal behavior with such effects included is facilitated.

Specification of the intermediate layer is supported by a software tool that allows convenient definition of the TDL elements as well as a platform definition and a mapping between the two. Moreover, the software tool is integrated with the *de facto* standard in embedded control system design, MATLAB and Simulink software by MathWorks. It leverages *call backs* and a Java Application Programming Interface (API) to allow conveniently moving back and forth between applications and keeping changes between the different applications consistent.

The TDL specification can be translated into a Simulink equivalent that allows various forms of simulation and from which code can be generated. Function-Call Subsystems are the pivotal semantic construct of the Simulink implementation to facilitate the tasking abstraction of the software stack, whereas the scheduling that implements the timing requirements is implemented by S-Functions that serve to initiate the function calls.

The Simulink Merge block is used to facilitate different tasks writing to the same variable. This ensures that data dependencies are observed, and only requires mutually exclusive writes,

which the tasked specification can provide. In addition to the Merge block Rate-Transition blocks are further employed to serve data storage purposes. Experiments with the Data Store Memory block were performed for comparison which mainly showed syntactic advantages but drawbacks in potential semantic complications.

Real-Time Workshop Embedded Coder (RTW-EC) software by MathWorks is employed to generate code from the task functionality that is specified in the separate Function-Call Subsystems. A wrapper for this RTW-EC code allows it to be integrated with the TDL code.

3 Contributions

TDL establishes a structured framework to specify temporal behavior and study the timing effects. This contrasts with the typical approach in Simulink where such behavior is specified at an individual block level. The benefit is a systematic separation of concerns. The drawback of previous work was the atomicity of tasks. This dissertation mitigates (and eliminates in some respect) the problems because of corresponding causal dependencies by *splitting* the execution of a task into an output and an update stage. Such separation of functionality has a solid foundation in the general control system engineering discipline as manifested, for example, by the S-function API of Simulink.

A specific advantage of the execution splitting is support for control of plants that have causal relations between input and output, so called *direct feedthrough*. In the *state space* formulation of control theory, this corresponds to a nonzero D matrix.

A further contribution is the introduction of fine grained control over task scheduling by creating time *slots*. This dissociates the periodicity of a task from its execution time and further enables an efficient implementation of *asynchronous* tasks. The latter are aperiodic tasks and would generally interfere with the timing analysis of a TDL specification that is based on periodicity and worst case execution time. This dissertation puts forward constraints on interaction between periodic and aperiodic tasks as well as their priorities so as to support a form of asynchronicity. An important aspect of this is the execution time of such aperiodic tasks being their worst case execution time and disallowing more than one instance of a currently scheduled task.

An additional contribution is in the area of software engineering so as to facilitate the integration of a Java application with MATLAB and Simulink. Exception handling has become a critical part of software engineering for business applications, and this work performs a sys-

tematic study of the nature and ground for exceptions in the corresponding Java application, with the aim of software refactoring.

The resulting software application enables another contribution; the mapping of functional specification elements onto architectural elements of a given platform. A number of platform models are included with 'plug ins' for code generation.

4 Future Work

The dissertation includes a discussion of future work. What follows are particularly important elements to be reinforced and to be additionally identified. Where the TDL framework introduces an effective and useful separation of concern, it also creates another layer in the overal design of a system. The result is a compartmentalization of optimizations, which is currently a particulary important topic to resolve for *Cyber-Physical Systems*. Future work may concentrate on how to facilitate *cross-layer optimizations* in a structured manner.

Performance in general is an important topic for future work to further the quality of the work to attain industrial grade. Along these lines, robustness is crucial, for example, in the face of exceptional scenarios. Schedule overruns may result from these and future work may attempt to tackle such phenomena in the TDL framework.

Another component of the industrial grade objective pertains to the handling of aynchronous events. These are quite common in industrial applications and it would serve the future of the work documented in this dissertation well if asynchronous events could be treated on equal footing as synchronous behavior. This includes asynchronous event support for a dense time base and, more importantly, high priority over synchronous tasks.

In addition to support for digital control systems, the domain of this dissertation, future work may specifically address requirements of *digital signal processing*. Key elements that must be addressed there are: (i) the task splitting, which may require a state reconstruction that does not work well for complicated signal processing algorithms; and (ii) the vast amounts of data that signal processing operates on, which necessitates prudent use of memory resources. This puts forward the potential investigation in future work of the trade with quality of service.

Finally, it could be interesting to study how the use of worst case execution times (WCET) has an effect on nominal performance [2]. It could well be that the use of a different measure than WCET could lead to superior average behavior.

5 Grading

The dissertation is very well written with clear and concise language. Figures are clear as well and prudently used. The referenced work provides a good landscape of the abundance of work in the general area of the dissertation. The contributions of the research are well motivated, clearly presented, and innovative while maintaining a practicable perspective. This is corroborated by publications in proceedings of esteemed peer reviewed fora. The significance of the contributions in moving the software engineering of digital control systems forward is substantially corresponding to doctoral grade work. Moreover, solid software design and engineering is demonstrated by the implemented tool support for automization. In conclusion, the overall quality of the work is high and deserving of my grade:

Sehr gut (1)

References

[1] Jon Friedman and Jason Ghidella. Using model-based design for automotive systems engineering – requirements analysis of the power window example. In *Proceedings of the SAE 2006 World Congress & Exhibition*, pages CD–ROM: 2006–01–1217, Detroit, MI, April 2006.

[2] Pierre G. Paulin, Olivier Benny, Michel Langevin, Youcef Bouchebaba, Chuck Pilkington, Bruno Lavigueur, David Lo, Vincent Gagne, and Michel Metzger. MPSoC platform mapping tools for data-dominated applications. In Gabriela Nicolescu and Pieter J. Mosterman, editors, *Model-Based Design for Embedded Systems*, pages 179–206. CRC Press, Boca Raton, FL, 2009. ISBN 978-1-420-06784-2.

[3] Katalin Popovici and Ahmed Jerraya. Programming models for MPSoC. In Gabriela Nicolescu and Pieter J. Mosterman, editors, *Model-Based Design for Embedded Systems*, pages 231–258. CRC Press, Boca Raton, FL, 2009. ISBN 978-1-420-06784-2.

[4] Bill Potter. Use of the mathworks tool suite to develop do-178b certified code. In *ERAU / FAA Software Tools Forum*, Daytona Beach, Florida, 2004.

[5] Michael Sullivan. TACTICAL AIRCRAFT–F/A-22 and JSF acquisition plans and implications for tactical aircraft modernization. Technical Report GAO-05-519T, United States Government Accountability Office, April 2005.