Second Year ARTES Project Report

A4-9805: Real-Time Software for Versatility, Scalability and Reconfigurability in Complex Embedded Feedback Control Systems

Jan Wikander, Martin Törngren, De-Jiu Chen
Mechatronics Lab, Machine Design, Royal Institute of Technology,
S-100 44 Stockholm, Sweden
jan@md.kth.se; martin@md.kth.se; chen@md.kth.se;

1. General

The goal of the MARCH project (Mechatronic ARCHitecture – MARCH, is the short form project name.) is to progress state of the art of software architecture for complex feedback control systems. A specific goal of the project is to provide a system architecture suitable for scalable, maintainable and reconfigurable mechatronic control systems. The research findings are in the end planned to be evaluated and partly verified on a real robotic system, namely the 4-legged robot Warp1, which is now made operational within CAS (Centre for Autonomous Systems), KTH.

2. Comparison between project plan and results

The research plan according to the original project proposal includes the following research tasks that are here complemented with corresponding comments about status and achieved results.

- Establishment of a requirement specification for the architecture. This task is performed strongly linked to analysis of the control requirements (as formulated at the corresponding point in time) as well as non-functional platform requirements.

Deliverable:	Specification
Status:	Finished
Results in brief:	Large mechatronic software should meet both the requirements of logical and temporal determinism imposed by motion and behavior control, and the requirements of non-determinism arising from the needs for flexibility. To meet these goals, the system complexity resulting from the multidisciplinary nature of mechatronics design as well as the unconstrained flexibility of software has to be properly managed. Utilizing the results from work on generic software architecture to improve mechatronics software should be carried out with great caution due to the semantic gap between these two software application domains. For example, the stringent requirements of temporal determinism and dependability, arising from the dynamics under control, are often omitted in generic software architectures. One important concern in our research is to provide a good comprehension of the relationships/interdependencies among mechatronics software constituent units.

- A state of the art study on software architectures for real-time feedback control systems. This study includes definition of concepts and notation (e.g. what is an architecture, which are the problems?, etc).

Deliverable:	Report
Status:	Finished
Results in brief:	An extensive state of art study has been performed. Software architecture constitutes one promising approach to achieving systematical development of software systems, where the satisfactions of multiple quality goals are carefully assessed and balanced at early

	development stages. As the science and art of structuring, software architecture should focus on the design of overall software structures using domain specific technologies as well as generic structuring principles (e.g., separation-of-concerns). By providing abstractions of the system under development, software architecture also supports complexity management, which is essential to facilitate e.g., cross-domain communication, verification, and maintenance. Currently, the technology to support this approach in the development of mechatronics software, characterized by embedded computer control specific issues, is still underdeveloped. There are many reasons for this. While recent research in software engineering has made progress in architecting generic software systems (e.g., through object-oriented design), the technologies are often developed independent of embedded computer control. The lack of computer control specific considerations significantly limits the usability of those results.

- Evaluation (without any implementation or at least only minor implementation work) of existing relevant software architecture concepts with respect to the specification. This task includes the establishment of an evaluation procedure.

Deliverable:	Evaluation report.
Status:	Finished
Results in brief:	The uses of architectural concepts in engineering practices have been studied by means of three solution architectures, which are subject to different purposes and different applications. These architectures include an open architecture for machine tool controllers (i.e., *OSACA* - Open System Architecture for Controls within Automation Systems), a fault tolerant system that supports online upgrade of control software (i.e., *Simplex* Architecture), and time-triggered infrastructure for real-time control (i.e., *TTA* - Time-Triggered Architecture). The issues that are of particular concern in the evaluation include objectives, application characteristics, scopes, techniques, and the corresponding development concept. From an architecture point of view, these architectures share one common theme in structuring. That is, one key to most of the nonfunctional quality attributes such as modifiability, integrability, and reusability is the interface or boundary between system building blocks. In a sense, architectural design can be thought of as boundary or interface design.

3. Main achievements and contribution

In current approaches to architecture for general purpose software systems, the interfaces and the relationships of architecture "building blocks" under consideration are mainly concerned with direct data and control interactions. The relationships with respect to application semantics and implementation such as triggering and allocation are most often not explicitly considered. In embedded computer control, the logical, temporal, and spatial characteristics of software must be considered as a whole because of the physical process under control.

The initial results from our efforts to adopt the concept of software architecture in the development of mechatronic software systems are presented using a design framework for architecting such systems. The design framework includes two models: (1) an *architecture model* that concretizes fundamental architecture "building blocks" and their relationships (i.e., *interactions* and *interrelations*) in the context of embedded computer control, and (2) a *preliminary decision model* for flexibility that extends the notion of coupling-and-cohesion in the context of embedded control software to support the reasoning about flexibility. The architecture model aims to provide an effective basis for managing complexity and performing architecture based system development. The basic consideration is that the qualities of a large system are largely determined by the external properties of system constituent units and their relationships. Complexity could be managed through controlling the number and the multiplicity of system constituent units and their relationships both in modeling and in system solution itself. The preliminary decision model aims to provide a basis for reasoning about how to meet flexibility-related qualities. The basic consideration is that flexibility as well as most of the other nonfunctional quality attributes mainly depends on the interdependence and the boundary of system constituent units defined in structuring. In the proposed framework,

flexibility represents the ability to make changes quickly, easily, and cost effectively. It denotes a collection of non-functional quality attributes referring to the abilities of replacing, adding or deleting, enhancing or reducing, or restructuring application software.

Through explicitly defined fundamental “building blocks” and “load bearing beams” of mechatronic software systems, our framework has laid a foundation for architecture based development of embedded computer control software systems. Especially, this provides a basis for correctly modeling the architectures and for correctly defining the software components. The basic consideration is that the architecture model of the framework principally defines what should be modeled in a system description and what should be included in the interface of a software component. The architecture model can also be used as a conceptual basis when different domain specific structuring techniques need to be coordinated/integrated.

4. Publications

De-Jiu Chen, Martin Törngren, *System Architecture in a Mechatronics Perspective*, Compendium of Papers, SNART 99 Real-time System Conference, Linköping, 1999.

De-Jiu Chen, Martin Sanfridsson, *Introduction to Distributed Real-Time Control*, Technical Report, TRITA-MMK 1998:22, ISSN 1400-1179, ISRN KTH/MMK/R-98/22-SE., 2000.

De-Jiu Chen, *System Architecture for Mechatronics Systems - A survey of the concepts, theories and methods with the focus on software*, Technical Report, TRITA-MMK 1999:30, ISSN 1400-1179, ISRN KTH/MMK/R-99/30-SE., 2000.

De-Jiu Chen, Martin Törngren, *Towards A Framework for Architecting Mechatronics Software Systems*, Proceedings Seventh IEEE International Conference on Engineering of Complex Computer Systems, (ICECCS 2001), Skövde, Sweden, 2001, ISBN 0-7695-1159-7.

De-Jiu Chen, Architecture for Development of Mechatronics Systems, Licentiate Thesis, Dept of Design, KTH, TRITA-MMK 2001:06, 1400-1179, ISRN KTH/MMK--01/06--SE, 2001.

Other Reports:

De-Jiu Chen, *Towards a software component framework for mechatronics applications*, CBSE Course Report, 2000.

De-Jiu Chen, *Towards A Model-Based Evolutionary Design for Complex Mechatronics Systems*, Konstruktions-metodik, Graduated Course Report, KTH – LiTH, 2000.

5. Statement from associated industries