University of Minho

Department of Computing

Masters in Computing Engineering

http://mei.di.uminho.pt/?q=en/mfes-en

Formal Methods in Software Engineering (MFES)

2016/17

Cohesive Project (CP) Workshop : Presentation of Project proposals

Room DI-0.03

9h Feb, 2017

With Partnership by

On-board Software Component Interaction (OSCI)
Collaborative Project ESA - MEI (University of Minho, Portugal)
2017

Figure 1 Front gates of ESA's technical heart (ESTEC) in the Netherlands (Copyright ESA–G. Porter)

The **European Space Research and Technology Centre (ESTEC, Netherlands)** is the largest ESA site and it's where most space projects are born and guided through the various phases of development.

The **on-board/flight software (OBSW)** is the "brain" of any spacecraft and is designed for a multitude of mission profiles, such as **science and robotic exploration, earth observation, navigation, telecommunications and human space-flight.**

This software provides indispensable functionality like attitude and orbit control, science data handling, thermal control and life support. Nevertheless any OBSW faces hard challenges like operation under extreme environmental conditions for several years with limited computing resources. Also the need for spacecraft autonomy is increasing as ground communication may take several minutes in long distance missions and may even not be possible, for instance when a celestial object is between the spacecraft and the Earth. Any SW failure under these harsh conditions can thus represent the loss of the **mission (mission critical SW) or even life (life critical SW)**.

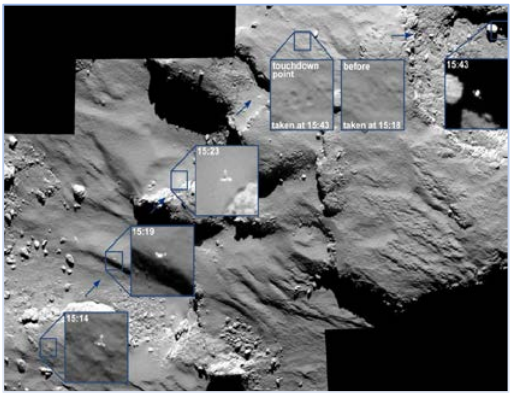


Figure 2 Rosetta's Philae lander touchdown on Comet 67P/Churyumov–Gerasimenko (Credit: ESA/Rosetta/MPS for OSIRIS Team MPS/UPD/LAM/IAA/SSO/INTA/UPM/DASP/IDA)

Motivated by these challenges, the **Software Systems Division (TEC-SW)** develops R&D projects that aim at high dependable software, decreased development times/cost, and increased reusability and performance. Some adopted approaches include Model based software and systems engineering (MBSSE), correct-by-construction design, leverage of Domain Specific Languages (DSLs), formal techniques application, code generation, thorough simulation and fostering of reference architectures.

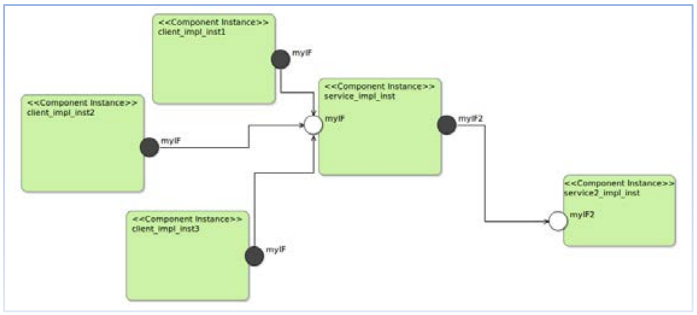


Figure 3 OSRA concurrent clients and call chain

The proposed project **On-board Software Component Interaction (OSCI)** focus on the agency's On-board Software Reference Architecture (**OSRA**) communication aspects. This architecture promotes a strong component based approach to the development of flight software and is intended to become the "lingua franca" among all the European space industry members (national space agencies like CNES and aerospace companies such as Airbus and Thales).

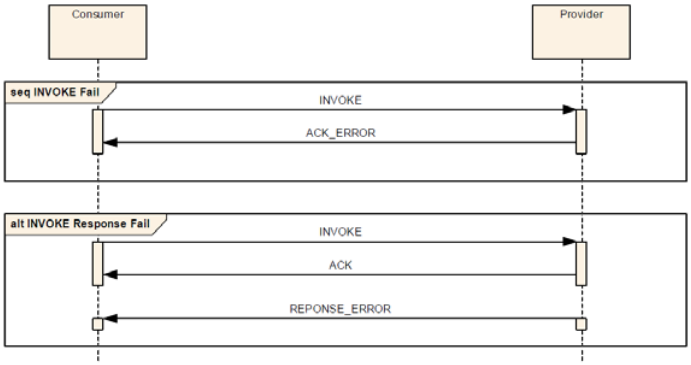


Figure 4 INVOKE Interaction Pattern (CCSDS 521.0-B-1)

Components are seen as service providers with clearly defined interfaces representing the interaction "contracts" between them.
OSRA fosters a clear **separation of concerns, in particular functional and non-functional aspects**.

In this project the students will model the communication between the components as a set of **interaction patterns inspired in real space industry standards (ECSS, CCSDS)** and reason about the correctness of the resulting transactions. Complex scenarios like multiple concurrent client components and long call chains shall be considered. Desirable properties related to the composability of components can be devised and verified. As a result, a better understanding on how these interactions can be generically implemented in a dedicated OSRA middleware shall be achieved.

Students will have the opportunity to get to know an industrywide reference architecture, experiment with software tools from the TEC-SW laboratory, get familiar with space industry standards and their philosophy, look into real project documentation, understand ESA's needs and interact with experienced engineers. In order to meet the students' skills and, more importantly, interest, no particular formalisms or tools will be imposed.

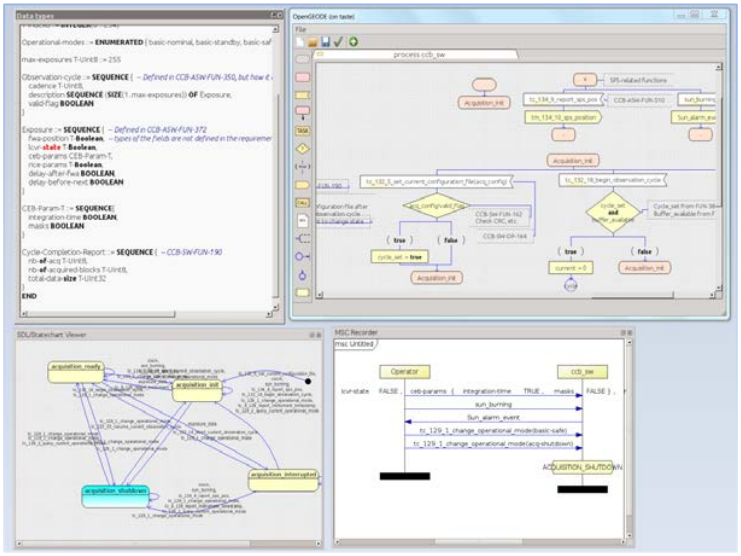


Figure 5 TASTE toolset UI (TEC-SW)

Legal reasoning

Design of Argument Assistance systems

Summary

A lawyer uses arguments to convince a judge of his client's position. In the literature about argument assistance systems, there is consensus about two points.

1. Mathematical logic may be useful for building software, but it doesn't go very far in helping legal professionals in their daily work.
2. Argument assistance systems (AAS) must try to aid a legal professional to create his or her own line of argumentation. It should not try to make any decisions for him or her.

This project is about developing software that can help lawyers to build arguments.

The goal of the information system that will be developed, is to check if the argumentation that has been built by the lawyer, is valid or not. In the literature we have not found any formal model to support legal reasoning. Systems that have been built so far were designed without a formal analysis of the structure of legal argumentation. In this study, such a formal model will be built. In doing so, we will use Ampersand, because an information system can be generated directly from the formal model. This will help get a better understanding of structures of legal reasoning.

Our literature research shows that the argumentation theory of Stephen Toulmin and the ArguMed system of Bart Verheij form a solid basis for the development of the prototype. The characteristics of both (underlying) theories have been compared to each other.

This assignment is part of the "MirrorMe" project, an open source project within Ampersand. Participation of Ordina and TNO are part of this project.

Keywords
Legal reasoning, formal logic, information system, ampersand, Toulmin, ArguMed

Acknowledgement
This summary and the literature list have been taken/adapted from the preparatory graduation report of Elleke Baecke, "Legal reasoning – Formality awaits", Open University of the Netherlands, 30 september 2016.

Literature

Bex, F. & Reed, C. (2011). Schemes of Inference, Conflict, and Preference in a Computational Model of Argument. *Studies in Logic, Grammar and Rhetoric*, *23 (36),* pp39-58.

van den Braak, S.W. (2010): Sense-making software for crime analysis. Enschede: Susan van den Braak.

Joosten, S., Wedemeijer, L., & Michels, G. *Rule Based Design* (December 2013 ed.). Open Universiteit Netherlands.

Michels, G., Joosten, S.J.C., Woude, van der, J.C.S.P. & Joosten, S.M.M. (2011). Ampersand: Applying relation algebra in practice. In H. Swart, de (Ed.), *Relational and Algebraic Methods in Computer Science (12th International Conference, RAMICS 2011, Rotterdam, The Netherlands, May 30–June 3, 2011. Proceedings)* (pp. 280-293). (Lecture Notes in Computer Science, No. 6663). Berlin: Springer.

Prakken, H. (2005). Argumentatiemanagement voor juristen. Groningen: Henry Prakken.

Reed, C., Walton, D., & Macagno, F. (March 2007). Argument diagramming in logic, law and artificial intelligence*. The Knowledge Engineering Review, 22 (1)*, 1-22.

Toulmin, S. (1958). The uses of argument. Cambridge, UK: Cambridge University Press.

Verheij, B. (1998). ArguMed: A template-based argument mediation system for lawyers. *Legal Knowledge Based Systems. JURIX: The Eleventh Conference* (eds. Hage, J.C., Bench-Capon, T.J.M. Koers, A.W., de Vey Mestdagh, C.N.J., & Grütters, C.A.F.M.), 113-130. Nijmegen: Gerard Noodt Instituut.

Verheij, B. (1999). Automated argument assistance for lawyers (abstract). *BNAIC '99. Proceedings of the Eleventh Netherlands/Belgium Artificial Intelligence Conference* (eds. Postma, E., & Gyssens, M.), 269-270. Maastricht.

Verheij, B. (2001). Anchored Narratives and Dialectical Argumentation. *ICAIL-2001 Workshop on AI and Legal Evidence*.

Verheij, B. (2003). Artificial argument assistants for defeasible argumentation. *Artificial Intelligence* 150 (1-2), 291-324.

Verheij, B. (2007). Argumentation Support Software: Boxes-and-Arrows and Beyond. *Law, Probability & Risk, 6,* 187-208.

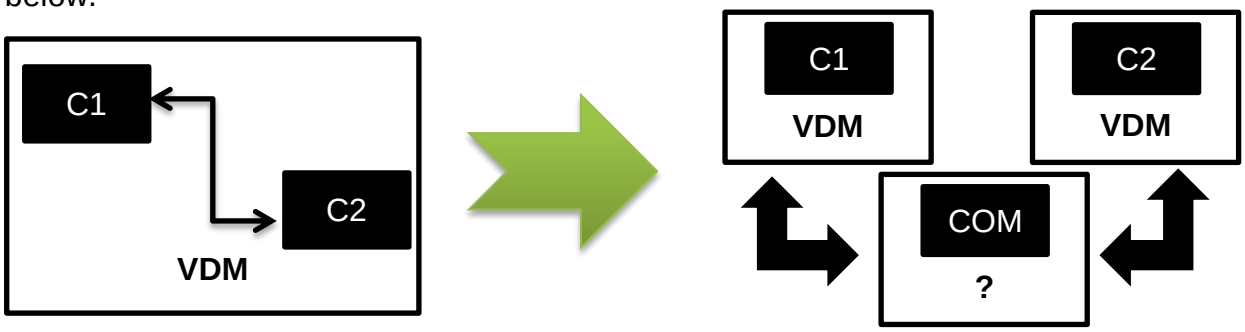
Verheij, B., & Hage, J. C. (2002). Rechtsinformatica als tak van wetenschap. In A. Oskamp, & A. R. Lodder (Eds.), *Informatietechnologie voor Juristen.* (2 ed., pp. 69-91). (Reeks informatica en recht; Vol. 20). Deventer: Kluwer.

Wahlgren, P. (1992). Automation of legal reasoning: A Study on Artificial Intelligence and Law.

Communication Verification within FMI Co-Simulations

United Technologies Research Center Ireland

Cyber-Physical Systems (CPSs) are composed of complex, networked computational and physical elements. This makes them challenging to design and analyze, as tools and capabilities from distinct disciplines are required.
One way to tackle CPSs is through Co-Simulations, where models from different notations (co-models) are simulated together (co-simulation). The INTO-CPS project (http://into-cps.au.dk/) is building a tool chain for model-based design of CPPs, using the FMI Co-Simulation standard. As part of INTO-CPS, UTRC-I are building a case study in the HVAC domain.
The FMI standard is primarily geared towards the exchange of continuous time signals, whereas computational models are often based on discrete event formalisms. In this project, students will experiment with an approach that tries to bridge this gap while also bringing verification capabilities to the FMI models.
Broadly, the project consists of taking an existing VDM model that encapsulates all communication within and transforming it into multiple models that communicate during co-simulation. To cope with the limitations of the FMI standard, a formal model of the communication should be developed, so as to verify desired properties, as illustrated below.



The high level tasks are:

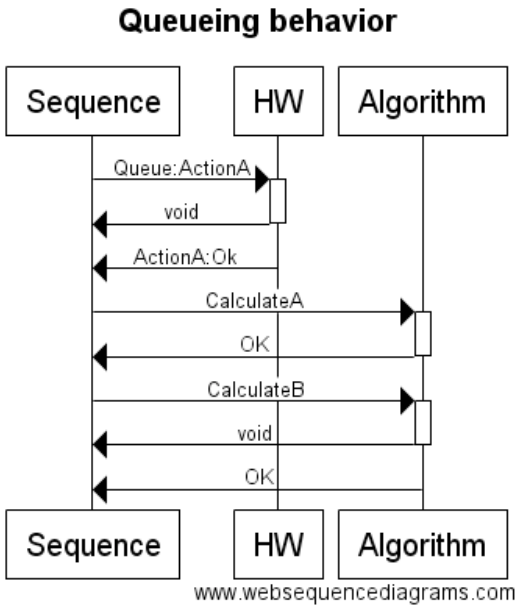
1. Modeling and verification of communication. One possibility is to model the BACnet protocol or its Controllers. For notation, we suggest RT-SPIN or UPPAAL
2. Construction of a Functional Mockup Unit to act as communication medium. This FMU must make use of communication timing and constraints uncovered through the verification. Possibilities include experimental support for FMUs in UPPAAL, using the JFMI tool or simply modeling the medium in VDM
3. Expansion of the existing HVAC example with the communication FMU
4. As a stretch goal, students are encouraged to experiment with code generation possibilities of the involved tools.

Alloy for sequence generation

ASML complex machines execute a sequence of actions in order to measure a wafer. The objective of this project is generating all possible scenarios given a set of actions.
A sequence consists of a set of actions. There are two types of actions: physical actions (queueing some action in the HW) and algorithms. Actions have dependencies between each other. There is always an order dependency between physical actions. This is HW actions are always queued in a pre-defined order.
Besides queueing requirements, there are also data dependencies between actions. An action might need the output of some algorithm in order to be triggered.

Defining all possible sequences using the constraint solver Alloy is the objective of this project.

The semantic of an action can be blocking or non-blocking. If an action is blocking the Sequence needs to wait and cannot do anything in the meanwhile. Queueing an action is always a blocking call but the result of that action is returned using a notification. This can be seen in the next picture. An algorithm can return immediately or return the result using a notification. You can see examples in the following picture. Notifications can be triggered at any order.



Here is partial model (only with queueing actions and no failures) provided by Professor José Nuno Oliveira illustrate the problem:

```
open util/ordering[Time]

abstract sig Action {pre: set Action}
```

```
one sig A, B, C, D extends Action {}

fact{

        pre = A -> B + D -> B + D -> C

}

sig Time { queue: Action -> lone State}

abstract sig State {}

one sig Acted, Ok extends State{}

pred Queue [t, t': Time] {

   some a: Action-(t.queue).State |

      a.pre in (t.queue).Ok and t'.queue=t.queue+(a->Acted)

}

pred Ok [t, t': Time] {

   some a: (t.queue).Acted | t'.queue=t.queue++(a->Ok)

}

fact {

  no first.queue

  last.queue=Action->Ok

  all t:Time-last | Queue[t,t.next] or Ok[t,t.next]

}

run {} for 9
```

Especificação e Implementação de Processos Baseados em Folhas de Cálculo

Supervisores: Doutor Francisco Duarte (Bosch)
Doutor João Saraiva (UMinho) [1]
Jorge Mendes (UMinho) [2]

As folhas de cálculo são dos sistemas de software mais usados. Embora tenham sido inventadas com a intenção de proporcionar um ambiente amigável e eficiente para efectuar operações matemáticas simples, a realidade actual é que folhas de cálculo são muito usadas na indústria para fazer tarefas bem mais elaboradas.

Neste projeto pretende-se estudar o caso do departamento de qualidade da Bosch (Braga), onde mensalmente um relatório de qualidade das componentes Bosch é enviado para a administração. Este relatório é produzido por um sistema de folhas de cálculo, onde a informação sobre problemas com componentes fabricadas é importado de diferentes sistemas. Estes dados são tratados na folha de cálculo: alguma informação é eliminada e outra adicionada, dando origem a um relatório (que inclui gráficos e que sintetiza toda a informação importada) a ser enviado à administração.

A manipulação de toda esta informação é feita mensalmente por um engenheiro de software, que importa e trata os dados manualmente: linhas/colunas na folha de cálculo são inseridas/removidas, fórmulas/gráficos são adicionadas. Todo este processo consome tempo e é feito manualmente sem qualquer especificação, nem documentação.

Em colaboração com a Bosch, e no contexto de um projeto de cooperação com um grupo de investigação Alemão, foi desenvolvido na Universidade do Minho uma metodologia para especificar todo o processo de importação, tratamento e evolução dos dados na folha de cálculo. Esta metodologia permite a automatização de todo o processo de produção de relatórios de qualidade tal como é feito na Bosch. Isto é possível descrevendo os elementos da folha de cálculo envolvidos e posicionando-os no local correcto do fluxo de trabalho.

Neste projeto pretende-se implementar esta metodologia como um add-on para o Microsoft Excel e posteriormente a sua validação usando a Bosch como caso de estudo.

Conhecimentos envolvido: alguma linguagem .NET (e.g., C#), desenvolvimento de linguagens (*parsers*).

[1] saraiva@di.uminho.pt
[2] jorgemendes@di.uminho.pt

Schedule:

- 10h00 - Welcome
- 10h15 - Presentation of project proposed by ASML (NL) by Andr Passos
- 10h35 - Presentation of project proposed by ESA (NL) by Tiago Jorge
- 10h55 - Presentation of project proposed by BOSCH (PT) by J.A. Saraiva
- 11H15 - Break
- 12h00 - Presentation of project proposed by INTO-CPS (IR) by L. Couto
- 12h20 - Presentation of project proposed by OU (NL) by Stef Joosten
- 12h40 - Closing

IMPORTANT:

The information contained in the current document is confidential and for exclusive use by FMSE sponsor companies, students and project tutors inside the classroom. NDAs will be signed wherever requested by industrial sponsors.