H2020—ICT—732410

RobMoSys

Composable Models and Software for Robotics Systems

Deliverable 5.1: Open call I preparation documents

Marie-Luise Neitz, Adam Schmidt (TUM), Sara Tucci (CEA), Herman Bruyninckx (KUL), Christian Schlegel (HSU)

Project acronym: RobMoSys

Project full title: Composable Models and Software for Robotics Systems

Work Package: WP 5, Management of Open Calls

Document number: 5.1

Document title: Open call I preparation documents

Version: 4.0

Due date: June 30th, 2017

Delivery date: 03.07.2017

Nature: Report (R)

Dissemination level: Public (PU)

Editor: Marie-Luise Neitz, Technische Universität München (TUM)

Author(s): Marie-Luise Neitz, Adam Schmidt (TUM), Sara Tucci (CEA), Herman Bruyninckx (KUL), Christian Schlegel (HSU)

Reviewer: Chokri MRAIDHA (CEA)

1 Executive Summary

Project RobMoSys, co-funded from the European Union's Horizon 2020 research and innovation programme under agreement No 732410, foresees as an eligible activity the provision of financial support to third parties, as means to achieve its own objectives. An overview of the workflow, the user-driven approach as well as the call documents developed for this first open call (call identifier: RobMoSys-1FORC) are outlined in this deliverable D5.1.

The vision of RobMoSys is to create **better models, as the basis for better tools and better software, which then allow to build better robotic systems**. Striving for a **step change in system-level composition of robotics**, RobMoSys launches this first open call to address single institutions or small consortia with a strong track record in model-driven software development, offering complementary, multi-disciplinary competences that go beyond the mainstream robotics community; for example, robotics experts teaming up with software engineering people, or tool builders, or experts from automotive, aerospace, embedded, cyber physical systems. Therefore, proposals submitted by tandems with complementary expertise are especially encouraged, e.g.:

- software engineering + robotics,
- industry (SME, large industry, small-craft industry) + academia,
- robotics expert + domain expert.

In the framework of this first call, 6-7 teams will be selected, with competences in **tooling**, the **development of models** and the **generation of associated software** (implementations that realise the models, and that are created/configured by the tooling) demonstrated on **system**-level prototypical scenarios in, e.g., navigation and manipulation. The tools, models and software developed by the successful third parties of this first open call will then be made publically available and serve to the industrial experiments and be integrated in the second Open Call.

RobMoSys thus asks for contributions that **realise a step change** in system-level composition for robotics, and that demonstrate this in **real-world scenarios**. The step change must not only be visible in the modelling foundation of the contributions, but also in the **industry-grade quality** of their realisation. Indeed, in the medium-term future, companies should be able to rely on the RobMoSys outcomes to build robotic applications by composing high quality composable models and associated software functions.

The high focus on user input despite the tight schedule of this call (to be opened just 6 months after the start of the project with a lot of preparatory work required) was possible because the consortium did not need to develop the approach from scratch, but could heavily rely on the instruments generated and expertise gained in the EU-funded FP7 projects ECHORD, ECHORD++ as well as the Horizon 2020 project HORSE and the FET-Flagship Human Brain Project. All documents and instruments had to be customized, though, to comply with the requests outlined in the best practice guidelines of the European Commission[1]. In addition to this, the project will strictly adhere to the Conflict of Interest rules developed by the European Commission and outlined in[2]

The deliverable is prepared by TUM and reviewed by CEA, HSU and KUL. The report contains a short introduction of the call, preparatory documents for the open call, dissemination plan, and how the recommendations from Tier-1 members are integrated in shaping the call. Deliverable 5.1 is contributed by task 5.1: Organisation of experts Workshops and Task 5.2: Preparatory activities.

2 Introduction

The deliverable 5.1. gives an overview of the timeline, the content (in brief) and the intention standing behind the first open call of the project. It also provides a short overview of the electronic tools and the guidelines, templates and supporting documents used to manage this call as well as the channels to

[1] See annex 7 Good practices and templates for organizing open calls under the H2020 financial support to third parties scheme

[2] http://ec.europa.eu/research/participants/data/ref/h2020/grants_manual/pse/h2020-guide-pse_en.pdf

promote the call. The RobMoSys project is heavily characterized by a **user-driven approach** of the software development. This claim is also reflected in the preparation of this first open call. The consortium ran an intensive loop of consultancy with a group of Tier 1 experts. Their recommendations have been integrated into the scope of the call. Furthermore, the preparatory activities also involved an intensive dialogue with the community. Again, the input was analysed, consolidated and condensed into the call documents. The deliverable D5.1. Can be illustrated as follows:

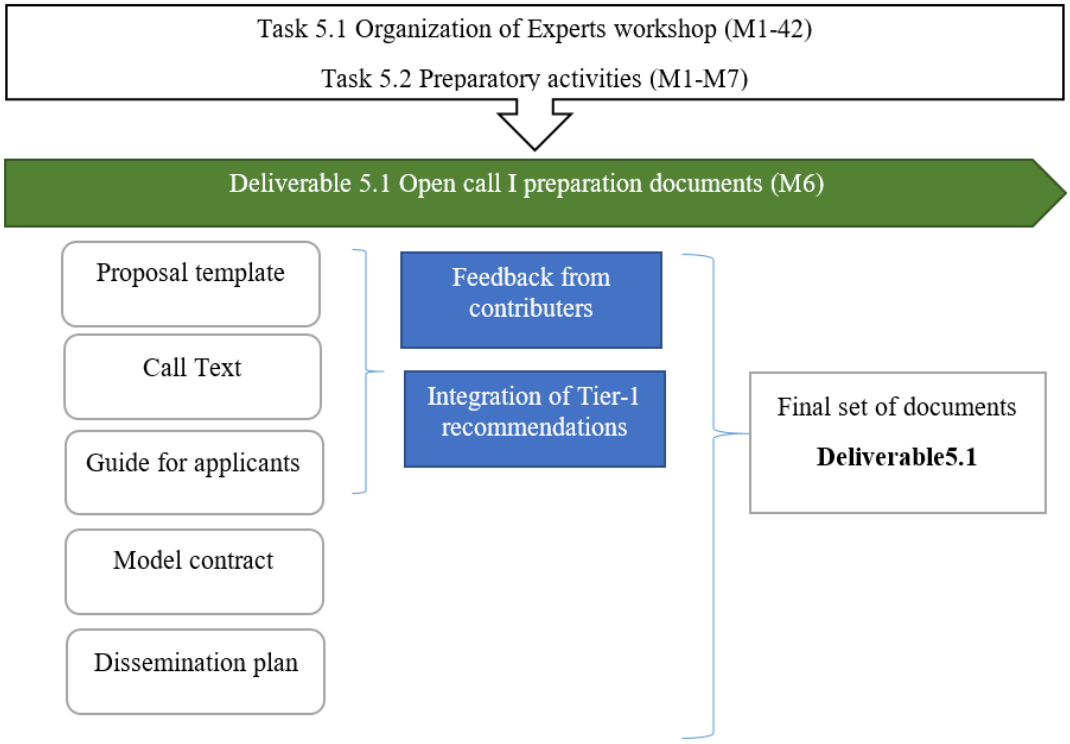


Figure 1: Content of deliverable D5.1. At a glance

With the first open Call, RobMoSys wants to extend the range of tools, models and software strictly adherent to the RobMoSys modelling principles (composability and conformity to meta-models) by adding 6-7 teams (small consortia or even single institutions) as third parties to the project. To achieve this goal, types of activities that qualify for financial support are *software developments* under the form of:

- **Models**
  - Composable models of components (ports, blocks, connectors enriched with composition constraints, resource requirements, etc.).
  - Models of system-level composition (system composed out of models of components) within a relevant use-case (composition for design-time or run-time composability).
  - Models to realise an architectural pattern, a design principle or best practice.
- **Tools and Meta-Models**
  - Extensions and/or improvements of, the provided RobMoSys meta-models (for instance for additional non-functional concerns such as Quality of Service, timing, performance, etc.).
  - Extensions and/or improvements of, the provided RobMoSys tools baseline (e.g. for design-time predictability, sanity checks, composability analysis, formal conformance verification, etc.). The current RobMoSys tools baseline is available here: http://robmosys.eu/wiki/baseline:start

Full open source contributions are preferred but not mandatory. However, the RobMoSys consortium expects at least the models and their transformations to proprietary tools to be under an open source license. In this first open call, preference will be given to projects that illustrate their contribution in the **domain of**

robot-centric motion, **navigation** and **manipulation.** The RobMoSys technical user-stories[1] provide a variety of possible topics that RobMoSys encourages to consider. More information and the full call documents, including the guides for applicants and an electronic submission system, can be found on the web site www.robmosys.eu. This First Open Call in a nutshell can be described as follows:

Call identifier: RobMoSys-1FORC

Call title: First Open Call for RobMoSys Contribution

Publication date: 10.07.2017

Deadline: 09.10.2017, 17:00 Brussels time

Expected duration of participation: 12 months

Indicative budget for the call: €2,000,000

Maximum funding request per proposal: 300,000 €
Funding rate: 100% (non-profit), 70% (for-profit)

Prefinancing: 25%

Submission language: English

Web address for full open call information: http://robmosys.eu/open-calls/

E-mail: opencalls@robmosys.eu

3 User-driven approach in shaping the call

Expert Workshops aim at gathering all the possible insights and knowledge (mainly from near communities and industrial representatives) to (i) evaluate best-practices established in near and mature domains and (ii) identify current showstoppers that could arise in the robotics domain. This understanding is necessary to make sure that Open Calls will be prepared to provide concrete answers to the community, to finally overcome identified showstoppers and secure broad adoption.

The first Expert Workshop was indeed prepared and held in February 2017, please refer to Annex 1 to find its detailed summary.

The method employed to gather insights from the experts included three phases, as follows:

- The RobMosys consortium explained initial ideas and focus points to the experts, notably on the scope of modelling, the emphasis on composability and the system level, the differences between Call 1 and Call 2.
- The experts critically reflected on our suggestions, and provided their views which went beyond "mainstream robotics"; a detailed list of recommendations can be found in Annex 1, Section 6.2.
- An interactive discussion between the consortium and the experts concluding on the following points
    - the conformance of our robotics approach with, especially, the OPC-UA developments in Industry 4.0 were discussed, and we found a very good fit; the RobMoSys ambition and concrete starting points are even leading those of OPC-UA, for example in the aspects of communication patterns, robot motions, and hierarchical system composition.
    - The concept of concurrent/collaborative system design, corroborating RobMoSyS ideas on modelling views and separation of concerns
    - Importance of methods suited for performance analyses at early design stages in robotics software to push assessment of design as early as possible in the development process

We can state that final result of the workshop is that our suggestions in the robotic domain were

[1] (see http://robmosys.eu/wiki/general_principles:user_stories)

corroborated and rephrased more sharply, aligning RobMoSys methodology and platform principles to the most advanced approaches in near domains, with a potential interesting synergy with the Industry 4.0 endeavour.

The call is in fact oriented to provide the community with general structures and fundamental modelling principles to (i) specify user/domain-specific knowledge in a coherent manner for further re-use and capitalisation, (ii) identify and use design patterns for communication, synchronisation, redundancy, timely behaviour, etc.; (iii) follow rigorous and effective design principles through composability, separation of roles and concerns.

4 Timeline, electronic tools, and documents

The timeline and workflow of RobMoSys to manage the first Open Call have been set up in compliance with the requirements summarized in the best practice guidelines of the European Commission.

4.1 Timeline

The first call of RobMoSys will be opened on 10th July and will be closed on 9th October 2017. The entire workflow from the opening of the call in July till the final selection in December will take 6, 5 – 7 months. The process can be illustrated as follows:

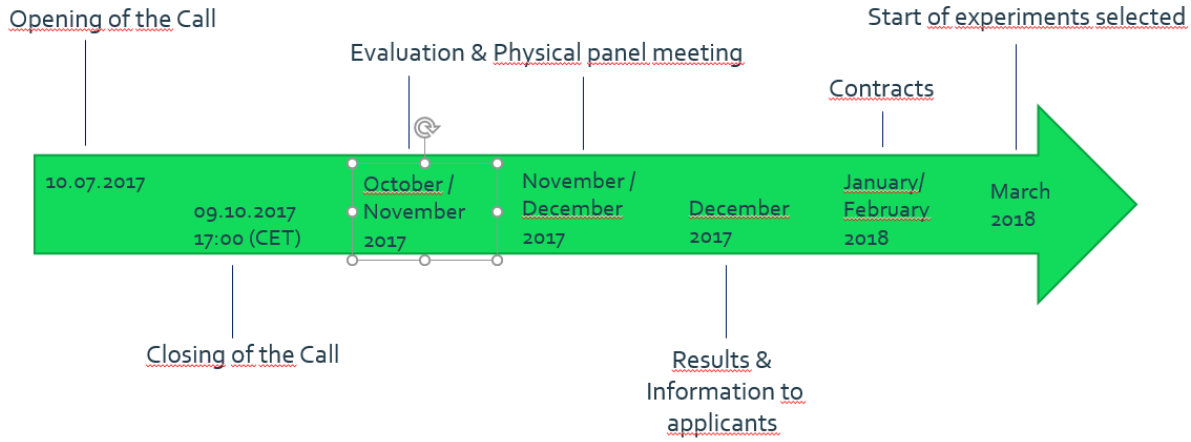


Figure 2: Timeline of the first Open Call

4.2 Electronic tools

The RobMoSys consortium has either customized or developed from scratch several electronic tools to comply with the requirement of the European Commission for a fair, transparent and impartial approach in the management of Open Calls for Third Party funding.

4.2.1 Project Management Tool – Redmine

In order to allow different members of the consortium to jointly work on the preparation of the first Open Call and to track the progress towards completion to meet the deadline, a Management platform- Redmine was set up. This platform allows to structure the entire process, assign roles and responsibilities. It gives an overview of the structure of each task, deliverable, and Milestone and provides the possibility to generate a Gantt chart for easy reference. This system helps to keep track of the progress in the preparation of this first Open Call in a centralized way.

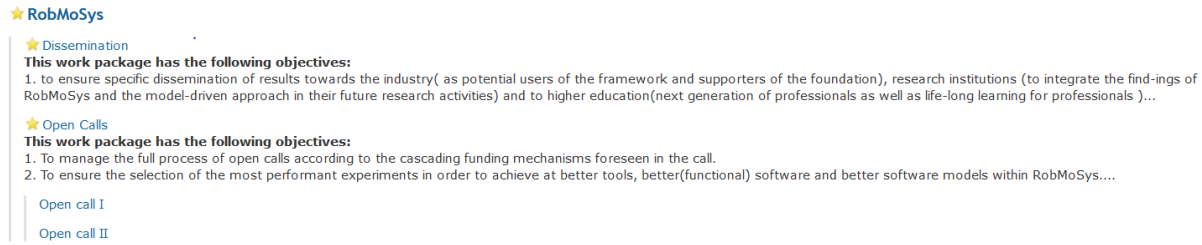


Figure 3: Overview of the Project management platform for RobMoSys

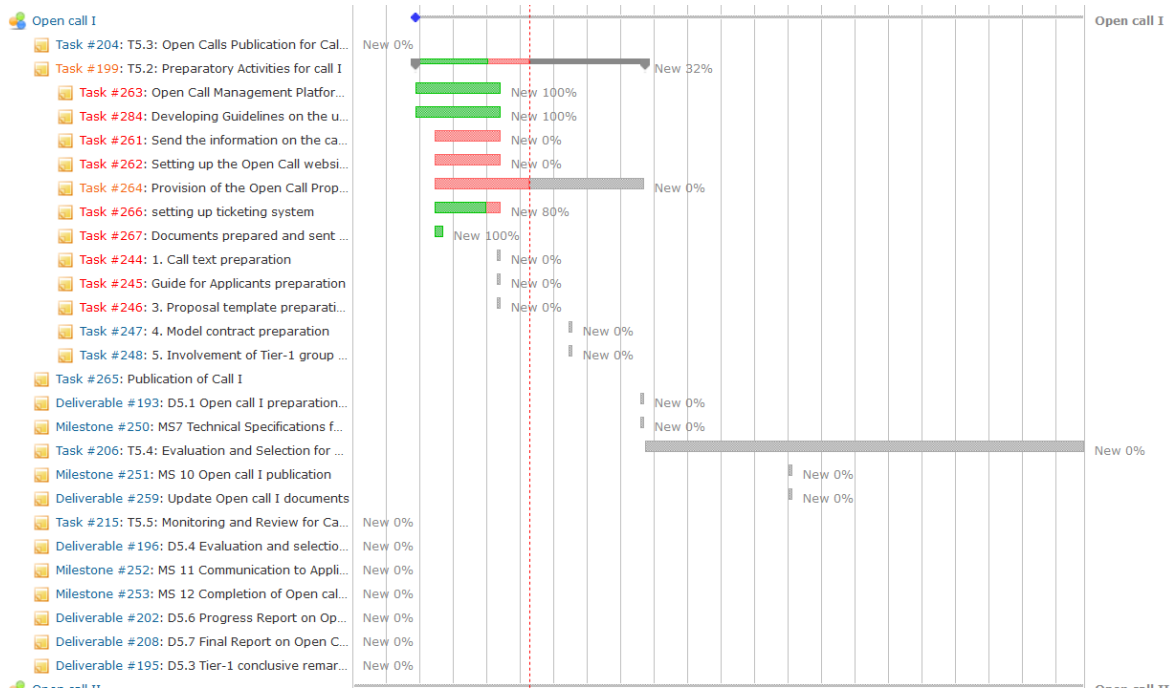


Figure 4: Gantt chart extracted from Redmine

Guidelines have been provided to all members of the core consortium with access to this management platform.

4.2.2 Open Call Management Platform

The platform will be used to manage applications received for the first Open Call of the RobMoSys project (to be opened: July 10th, 2017, to be closed at 17:00 Brussels time, October 9th 2017).

Applicants to the Open Call of RobMoSys will need to fill out the sections below.

Administrative data: This section on the platform will be used to collect general information on the consortium that is applying (also relevant to generate statistical data after the call)

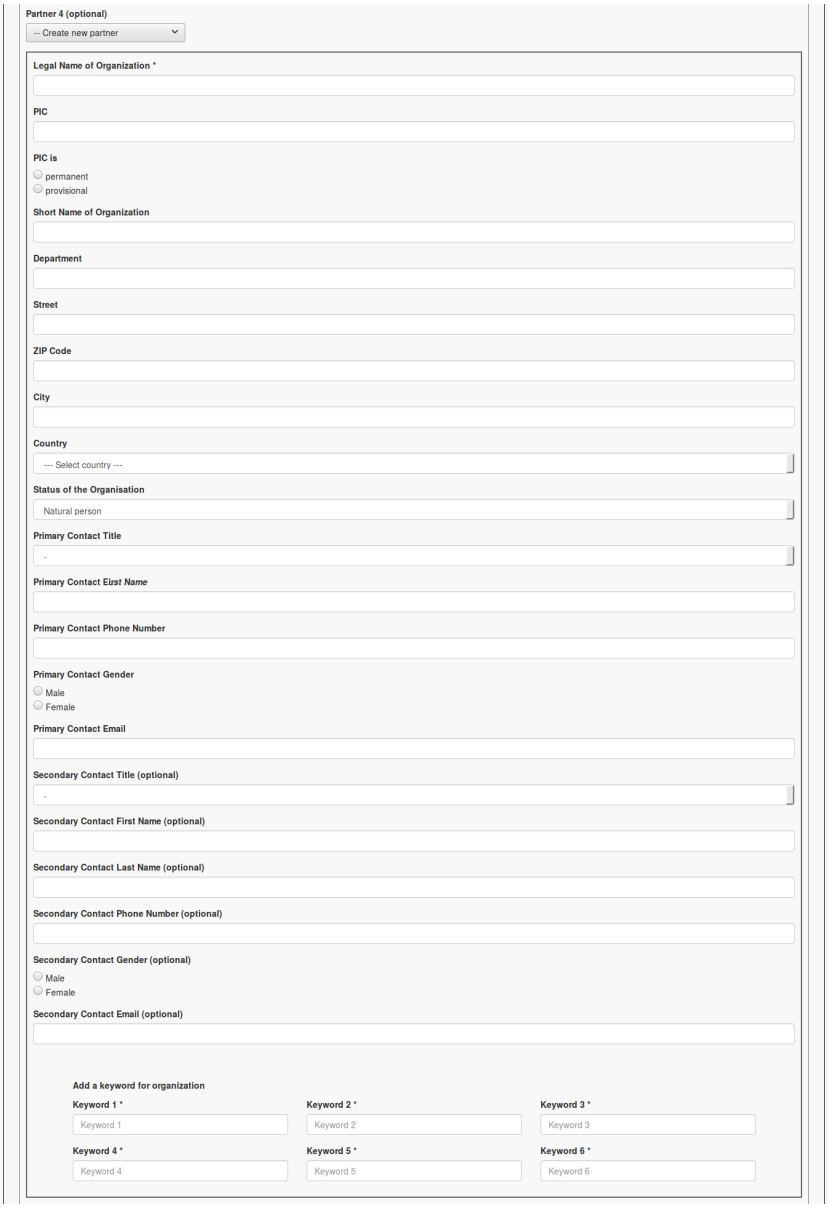


Figure 5: How to enter administrative data into the Open Call Platform

Proposal template: the proposal template will be provided to the applicants to complete it. Then it needs to be uploaded to the platform for submission.

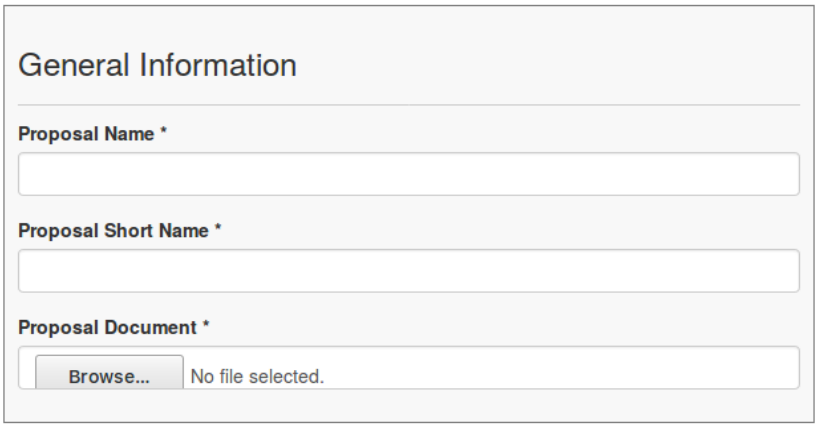


Figure 6: How to upload the proposal onto the Open Call Platform

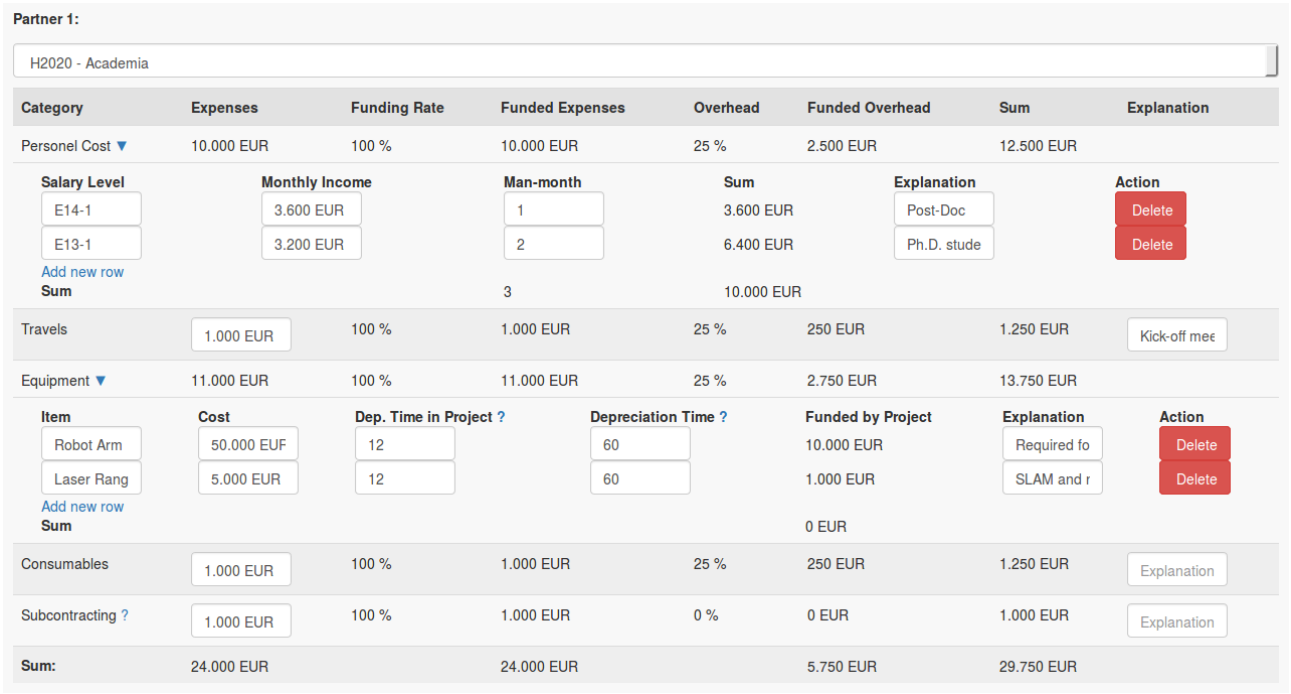


Figure 7: How to enter financal data into the Open Call Platform

Budget information – to make the provision of financial data – compliant with H2020 funding rules also applicable for the Open Call in RobMoSys – easier (mainly also for applicants without former experience with EU-funded projects), the Open Call Platform also provides tables which need to be completed and guides the applicants through this exercise.

Once the open call is closed, the open calls management team will be able to provide statics regarding the overview of applications and the applicants. Statistics on data such as the number of applicants, domains of expertise of the applicants, countries that participated in the application and type of institution that have applied will be provided to the EC.

4.2.3 Ticketing System

In addition to the Project management platform, a ticketing system (OTRS 5) to systematically archive and address enquires of applicants was set up. The ticketing system allows all incoming inquiries to be channelled to those members of the core consortium who are most competent to answer them (Administrative, General, and Scientific/ Technical). In addition, it provides an overview of the status of the enquiries (pending or closed) with a time stamp to make sure that all potential applicants will receive the requested information in a timely manner. The categories are administered by designated persons from four members of the core consortium: TUM, CEA, KUL, and HSU. Another benefit of the system is that the entire correspondence is stored in a closed system which allows tracking the flow of information between the RobMoSys consortium and the applicants in case of a redress. Statistical data on the number of enquires, the response time etc. can be generated, as well.

To guide the applicants throughout the entire application process, a description of the call will be provided along with Guide for applicants and Call text. An email address to the ticketing system will be also provided on the platform, if applicants have questions regarding administrative, scientific or general questions related to this call.

4.3 Documents and guidelines

The following documents have been generated to prepare the call:

- Annex I – submitted to the European Commission for publication on 10th July 2017
- Full call text (still under preparation – will be included in the updated version D5.1., due on 31st July 2017)
- Guide for applicants (still under preparation – will be included in the updated version D5.1., due on

31st July 2017)
- Proposal template (still under preparation – will be included in the updated version D5.1., due on 31st July 2017)

In addition to this, guidelines were provided to manage the electronic tools supporting the process:

- Guide on "How to handle the Open Call Management Platform" (for applicants)
- Guide on "How to manage the ticketing system" (for consortium members)
- Guide on "How to handle the project management tool Redmine" (for consortium members)

5 Promotion of the call

Different channels will be used to announce the Open Calls. The website plays a very important role as it will provide the link to the Open Call Management Platform which supports the proposal submission of the applicants and the access to the open call documents (call text, guide for applicants and proposal template). Documents are provided to download as word and TeX file.

Two Brokerage days are planned for informing the participants about the project and giving them the opportunity for networking among each other. The first Brokerage Day will be right in the beginning of the call, on July 5th in Leuven, Belgium, the second one will be on August 24th in Frankfurt, Germany. Registration for the brokerage events will be possible over the website.

The Open Call and the brokerage days will be announced over the social media channels, referring to the website for further information and registration. The RobMoSys newsletter will announce the Open Call and the brokerage events to its subscribers, additionally the consortium partners have offered to spread the information via their newsletters: EUnited for reaching out to the industry and the Eclipse Foundation for reaching out to software developers. The announcement will also be sent out to the euRobotics and the robotics-worldwide mailing list to reach all roboticists in general. The announcement will be accompanied by a press release which will be distributed to the special interest press for software developers, research, academia and industry.

These channels will be used to announce the Open Call before the opening, reminded in the middle of the call period and participants will be reminded again just before the closing of the call. Once the evaluation has been completed, the applicants will be informed about the results with the evaluation reports (mails). Furthermore, the results will be published on the RobMoSys website. The results will be announced over other channels, as well: social media, newsletters, mailing lists, press release.

6 Outlook

The Call will be closed on 9th October 2017, 17:00 CET Brussels local time. October 2017 and November 2017 will be dedicated to the remote evaluation. Each proposal will be evaluated by two independent experts. The remote evaluation will be finished with a consensus report (provided by an independent expert again, the rapporteur). In December 2017 a physical panel meeting will take place to accomplish the evaluation, ranking and selection of third parties. The first two months – January and February - 2018 will be invested to inform the applicants about the results and contract the successfully applicants and third parties. The evaluation and selection process will be described in deliverable D5.4.

Annex 1 - Expert Workshop Report

This report summarizes the content of the first Expert Workshop held in Frankfurt the 7th-8th of February 2017.

This is the first workshop of a series of Expert Workshops we set along the project lifetime to gather all the possible insights and knowledge (mainly from near communities and industrial representatives) to (i) evaluate best-practices established in near and mature domains and (ii) identify current showstoppers that could arise in the robotics domain. This understanding is necessary to make sure that Open Calls will be prepared to provide concrete answers to the community, to finally overcome identified showstoppers and secure broad adoption.

For this first workshop, the Consortium invited experts, from relevant related domains, with a strong scientific background in the design of complex and critical systems, namely:

- Arne Haman (Bosch) from automotive and embedded systems
- Jan Broenink (University of Twente) mechatronic/cyber physical systems – hybrid simulation
- Saddek Bensalem (Verimag) cyber physical systems – formal methods
- Jurgen Bock (Kuka) Industrie 4.0 – ontologies, semantic technologies

To prepare the workshop, each Expert discussed in an individual teleconference with the Consortium the general objectives of RobMoSys and the particular mission we were about to give to them. In order to set the context, the Consortium presented the following questionnaire:

"What is the aim of RobMoSys? RobMoSys envisions an integrated approach built on top of the current code-centric robotic platforms, by applying model-driven methods and tools. RobMoSys will enable the management of the interfaces between different robotics-related domains in an efficient and systematic way according to each system's needs. RobMoSys aims to establish Quality-of-Service properties, enabling a composition-oriented approach while preserving modularity. RobMoSys will drive the non-competitive part of building a professional quality ecosystem by encouraging the community involvement. RobMoSys will elaborate many of the common robot functionalities based on broad involvement of the community via two Open Calls. These are the topics we would like to hear your opinion on:

- How do you deal with composition and which are your priorities?
- How do you make sure that different vocabularies in connected components semantically match?
- How do you manage the link between composition and certification?
- How do you deal with quality-of-service properties (extra-functional properties) and how do you make sure quality-of-service is right?
- How do you assess good practices and what kind of metrics do you apply?
- What is still missing?
- In the different steps of the process, which one do you think is the hardest part & how would you solve it or absolutely not solve it?"

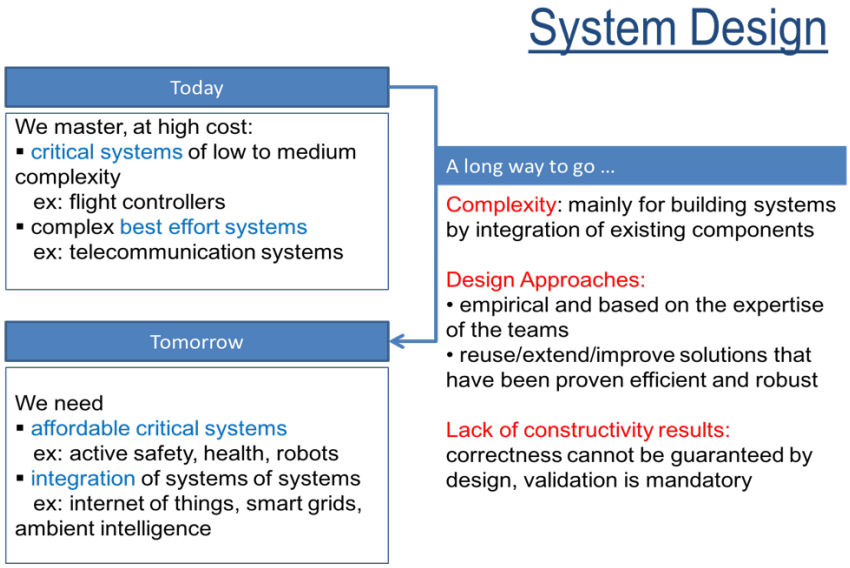
In the first part of this report (Experts Contributions) we summarize the contribution of each expert: the content of the presentation the expert made the first day. The second part (Current design methodologies assessment) presents the output of the second day: each expert was asked to fill a table pointing out several aspects and pain-points of presented design approaches. The third part of the report (Synthesis on recommendations for RobMoSys) summarizes the general recommendations of the experts.

6.1 Experts Contributions

6.1.1 Saddek Bensalem: a formal framework for system design

Saddek Bensalem presented principles and concepts related to rigorous system design. Saddek pointed out that reactive systems are increasingly important in modern computing systems: embedded systems, cyber-physical systems, mobile systems, web-services. They are hard to design due to unpredictable and subtle interactions with the environment, emergent behaviours, etc. Robots are a class of Cyber-Physical Systems.

System design is facing several difficulties, mainly due to our inability to predict the behaviour of an application software running on a given platform. Other difficulties stem from current design approaches, often empirical and based on expertise and experience of design teams. Naturally, designers attempt to solve new problems by reusing, extending and improving existing solutions proven to be efficient and robust. This favours component reuse and avoids re-inventing and re-discovering designs. Nevertheless, on a longer-term perspective, this may also be counter-productive: designers are not always able to adapt in a satisfactory manner to new requirements. Moreover, they a priori exclude better solutions simply because they do not fit their know-how.



Limitations of V-like models of traditional Systems Engineering processes can also be observed in this context. Indeed, V-like models:

1. Assume that all the system requirements are initially known, can be clearly formulated and understood.

2. Assume that system development is top-down from a set of requirements. Nonetheless, systems are never designed from scratch; they are built by incrementally modifying existing systems and by component reuse.

3. Consider that global system requirements can be broken down into requirements satisfied by system components. Furthermore, it implicitly assumes a compositionality principle: if components are proven correct with respect to their individual requirements, then correctness of the whole system can be inferred from correctness of its components.

4. Rely mainly on correctness-by-checking (verification or testing)

To overcome limitations of current approaches, a novel rigorous approach for system design is then needed, promoting the following principles:

- **Separation of Concerns**
- **Component-based approach**
- **Semantic Coherence**
- **Correct-by-construction**;

While rising three Grand Challenges:

- **Marrying Physicality and Computation**. We need theory and models encompassing continuous and discrete dynamics to predict the global behavior of a system interacting with its physical environment. The development of application software and its implementation must take into account constraints from: the physical resources and the physical environment of the system.

- **Component-based Design**. We need theory, models and tools for the cost-effective building of complex systems by assembling heterogeneous components

- **Adaptivity**. Systems must provide a service meeting given requirements in interaction with uncertain environments. Uncertainty can be characterized as the difference between average and extreme system behavior. Non-determinism of physical environments increases uncertainty.

and meeting the following objectives:

- **Productivity**. This can be achieved by system design flows providing ***high level domain-specific languages*** for ease of expression, allowing reuse of components and the development of component-based solutions, integrating tools for programming, validation and code generation.

- **Performance**. The design flow must allow the satisfaction of **extra-functional properties** regarding optimal resource management. This means that resources such as memory, time and energy are first class concepts encompassed by formal models. Moreover, it should be possible to analyze and evaluate efficiency in using resources as early as possible along the design flow. Design space exploration should be promoted to resolve choices such as reducing parallelism (through mapping on the same processor), reducing non-determinism (through scheduling), and fixing parameters (quality, frequency, voltage).

- **Correctness**. This means that the designed system meets its specifications. Ensuring correctness requires that the design flow relies on models with ***well-defined semantics***. The models should consistently encompass system description at different levels of abstraction from application software to its implementation. Correctness can be achieved by application of verification techniques.

To meet these objectives **model-based and component-based design should be merged** in a uniform **formal framework** with the following characteristics:

- Adopt a ***model-based design***. Model-based design means that software and system descriptions used along the design flow are based on a single semantic model. This is essential for maintaining the overall coherency of the flow by guaranteeing that a description at step n meets essential properties of a description at step n - 1. This means in particular that the semantic model is expressive enough to directly encompass various types of component heterogeneity arising along the design flow.

- Adopt a ***component-based approach***. Component-based design promotes composability and compositionality principles. The key issue is how to build systems from a set of given atomic components (behavior) that meet a given property. This is a hard problem. Nevertheless, it is important to have a framework for tackling this problem and decomposing it into simpler problems. That is to have a construction methodology.

Build a component C satisfying a given property P, from

1. Co a set of atomic components modeling behavior
2. GL ={gl1, …, gli, …} a set of glue operators on component

Glue operators:

- Model mechanisms used for communication and control such as protocols, controllers, buses.
- restrict the behaviour of their arguments.

The formal framework should offer a minimal set of constructs and principles for guaranteeing **correctness by construction**, *such as decomposition and flattening* **as depicted** in Figure 1.

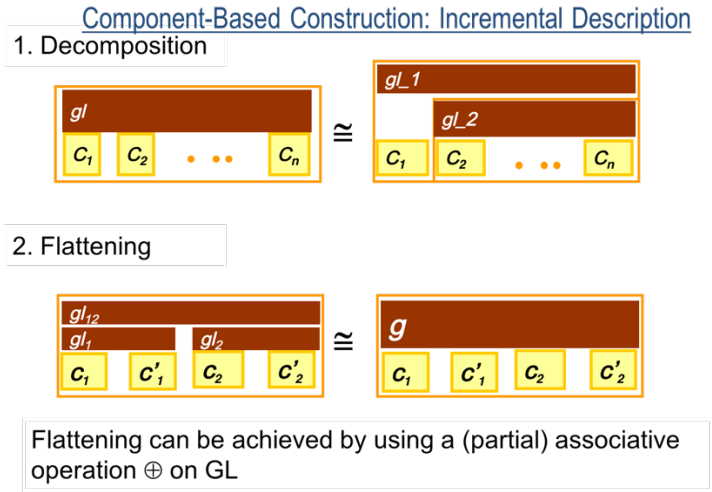


Figure 1: Decomposition and Flattening

The framework enjoys a generalization of associativity. Any n-ary glue operator is the composition of binary glue operators - This is very important for systems with dynamically changing structure. Dually, hierarchically structured components can be flattened – single glue operator applied to the atomic components. To achieve flattening some composition operation on glue is needed.

Component-based construction is based on two important properties, namely **composability** and **compositionality**. Composability is about composing components without breaking their properties after composition. Composability guarantees preservation of a component property across integration.

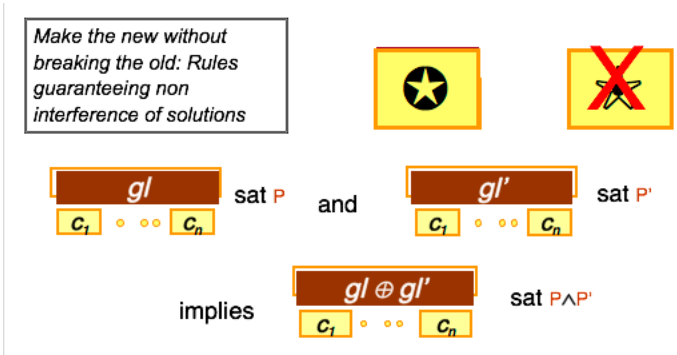


Figure 2: Composability

Compositionality allows deduction of the composed global properties from its component properties; this property enables correctness- by-construction.

Figure 3: Compositionality

To make an example Figure 4 shows two kind of compositions. The first one is a composition in which the composed components satisfy a certain property (no deadlock), and the property is preserved at the level of the composite component thanks to a proper operator. In the second composition, components do not satisfy any deadlock-free property, but the composite satisfies a Mutex property thanks to a Mutual Exclusion operator.

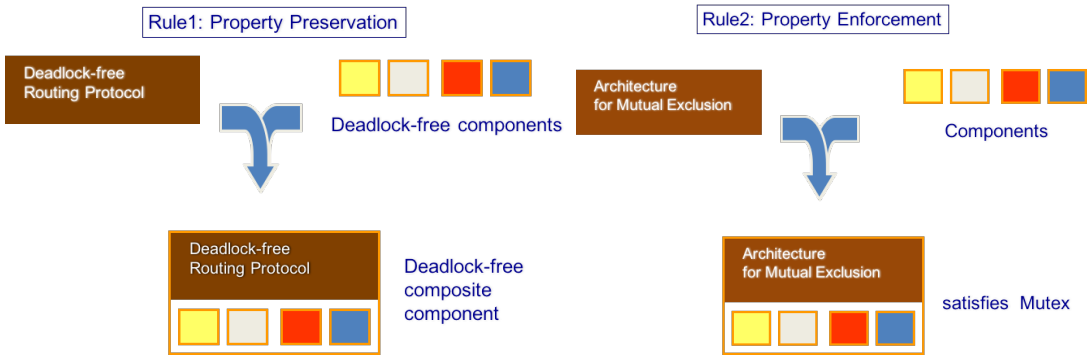


Figure 4: Examples of Compositions and Related Properties

- The formal framework should also be expressive enough to encompass ***heterogeneity of execution*** (synchronous and asynchronous components); interaction (function call, broadcast, rendez-vous); abstraction levels (hardware, middleware, application software).

- The formal framework should as well provide ***automated support*** for efficient implementation on given platforms and automated support for validation and performance analysis following the design flow suggested in Figure 5.

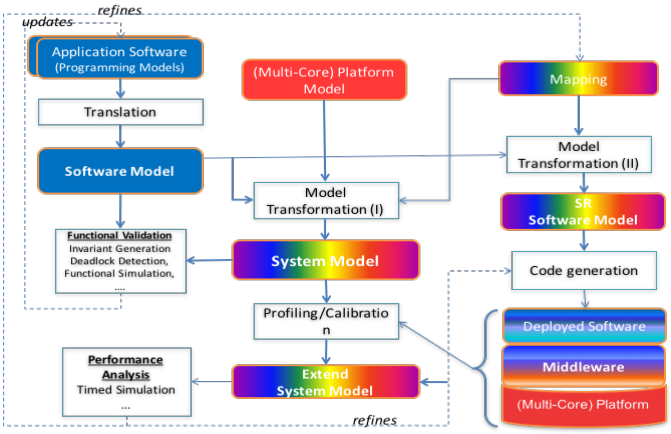


Figure 5: Design Flow

6.1.2 Arne Hamann: Essential Analysis and QoS management

Arne Hamann pointed out that ***model driven methods are key to boost design efficiency and confidence***. What makes model-driven methods fundamental is their suitability to compose functionalities on ***system level and then derive/predict system-level properties***. However, finding the right models & abstractions is extremely hard and requires in-depth domain knowledge. Many bad examples are out there, which lead to

the bad reputation of model based methods, like for instance using UML for modelling software. UML models often either lack a meaningful level of abstraction or lack clear semantics. UML models are, therefore, perceived to be only of little help by many software engineers.

6.1.2.1 Essential Analysis for Functional Domains

In order to deal with ***composition*** and issues on an underlying ontology supporting meaningful semantic connection among components, the suggestion here is to refer to morphological and essential system analysis[1]. These tools give help identifying right abstractions and concepts with respect to a given functional domain.

Essential Analysis (Figure 7) systematically ***decomposes the overall problem space according to discrete "situations" in the system context***. The objective is to identify sub-problems called **system *modes*** that can be solved independently. The obtained system modes can be used as blueprint to structure the implementation in later development stages. In the final implementation this automatically leads to the separation of control flow and the data flow. The decomposition of the problem space and the identification of system modes is based on a light-weight formalisation of system knowledge, i.e. a compact and unambiguous specification. This approach allows to check two fundamental properties during system design:

- **Completeness**: all possible states in problem space have been analysed
- **Consistency**: each part of the problem space belongs exactly to one system mode

Obviously, the decomposition of the functional domain can be carried out only through a dialogue between the System Expert that has a specific domain knowledge and the Methodologist that has a specific knowledge about the method. Essential analysis (including consistency and completeness checks) can be performed thanks to a tool proposed by ETAS[2].

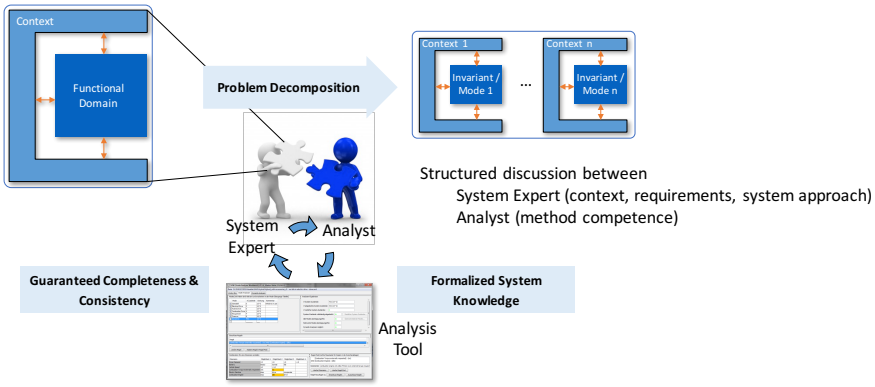


Figure 6: Essential Analysis Method

6.1.2.2 QoS management and link between composition and certification

The key factor to correctly manage QoS and certification issues for the expert is the ***separation of concerns*** between function and implementation. This aspect is also corroborated by the method suggested for the functional domain (see previous section), which of course can be used only if function and implementation are distinguished in the overall design process. Once again separation of concerns is of primary importance since SW components are often polluted by implicit assumptions that only hold for a specific target platform / middleware. More concretely the separation of concerns principle can be successfully pursued developing implementation-agnostic specifications, i.e. developing applications against **Abstract Interfaces** that are guaranteed on the target platform by **Platform Specific Implementations**.

---

[1] https://en.wikipedia.org/wiki/Morphological_analysis_%28problem-solving%29
Essential System Analysis - Stephen M. McMenamin, John F. Palmer, 1984

[2] Available solution in the Embedded/Control Systems domain is the Scode tool by ETAS:
https://www.etas.com/download-center/files/products_RTA_Software_Products/Whitepaper_SCODE_2016_12_19.pdf

In the context of QoS management those Abstract Interfaces usually rely on tailored platform mechanism matching specific Models of Computation, i.e. they explicitly refer to an execution model. Another important aspect is that the Abstract Interfaces must expose stable unambiguous semantics including non-functional properties. Obviously, to be useful in practice, the Abstract Interfaces must be verifiable and efficiently implementable.

Example 1. Logical Execution Time

The Logical Execution Time (LET) paradigm (Figure 7) decouples logical timing structure from physical execution resulting in portability, composability and deterministic communication between concurrent functional units. The LET paradigm solves the problem of software distribution on multi-core platforms and represent a strong-base argument for certification[1].

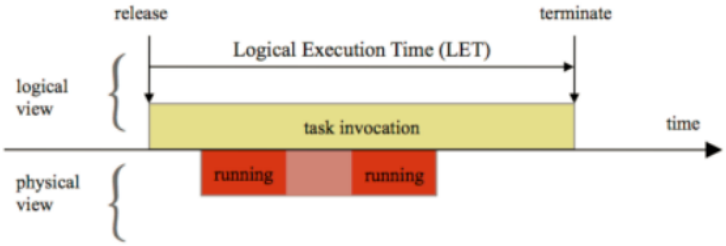


Figure 7: Logical Execution Time

Example 2. Reservation-based Scheduling

In robotics, most approaches are agnostic to OS mechanisms laying below the middleware layer, so that temporal behaviour is accidental and difficult (or impossible) to predict. As a matter of fact, standard scheduling approaches used in the embedded systems domain (such as Rate Monotonic Scheduling) are not adequate for robotic applications with dynamic resource requirements, leading to **strongly varying or unknown response times**. In the current state of practice, system integration in robotics is often achieved by drastic overprovisioning of computational resources. While such an approach is adequate for research prototypes, product engineering must rely on more systematic approaches leading to provably guaranteed temporal properties (for cost and certification reasons).

Reservation-Based Scheduling (RBS) naturally extends the LET paradigm to the execution management domain. RBS represents a comprehensible abstraction for handling computing resources enabling composability. With RBS, processor capacity is viewed as a quantifiable resource that can be reserved like physical memory. A task receiving a fraction U (<1) of the processor capacity behaves as if it were executing alone on a U times slower processor. Composability is achieved by temporal isolation: an application has access to reservation regardless of the other application executed in the system. Interestingly RBS can be dimensioned for average case (avoiding worst-case design) while improving overall utilization (no idling of cores like with time-triggered approaches).

[1] Derler et al.: Simulation of LET Models in Simulink and Ptolemy Monterey. Workshop 2008: Foundations of Computer Software. Future Trends and Techniques for Development pp 83-92.

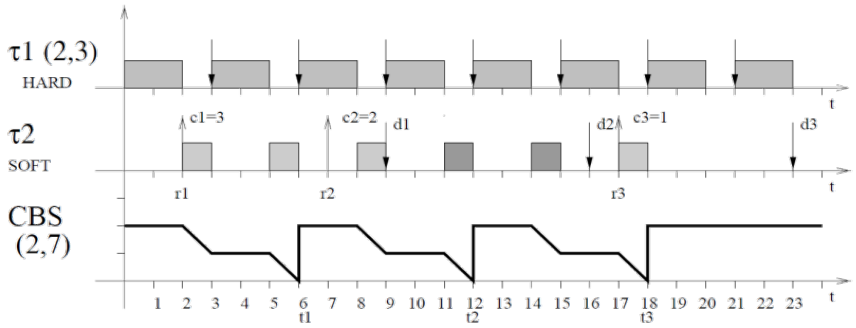


Figure 8: Reservation Based-Scheduling using the example of the Constant Bandwidth Server (CBS)[1]

RBS allows to develop applications independently and integrate them at the end, since reasoning about functional and non-functional properties is possible before integration. This leads to a huge gain in productivity and a string base for certification.

6.1.3 Jan Broenink : Multi-Paradigm Modelling for Cyber-Physical Systems

Robots can be viewed as a specific class of Cyber-Physical Systems, where the total system (cyber and physical) must be integrally treated and where safety aspects are relevant. The combination among the physical design (mechanical and electrical) and the cyber part (control software) must be handled properly, taking into consideration that after the transformation of signals to data a communication network is in general involved (Networked CPS). To handle such complexity a multi-paradigm modelling is proposed. Different kind of models must be then used to treat the total system. Used models differentiate each other in terms of modelling principles and Models of Computation. In the realm of Discrete Event models for instance different types of models can be found such as State Machine-like models, Process diagrams (block diagrams-like) and hardware description formalisms. Discrete Time models can either consider fixed Time Events or Variable Time Events. Other aspects define the model of computation such as the Communication model i.e. asynchronous (buffer) vs synchronous (rendezvous) and the level of synchronicity between calculations (asynchronous vs synchronous).

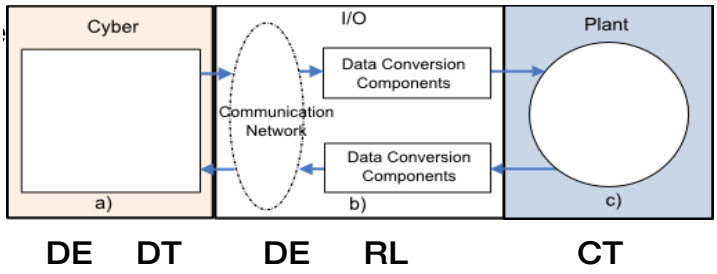


Figure 9: Combined Modelling

Figure 9 shows the different modelling paradigms chosen for the different parts of a cyber-physical system, ranging from Discrete Event (DE) to Continuous Time (CT).

Figure 10 specifies the combination of models used for the different activities involved in the design process. The software architecture (I-a), includes logic for decisions (sequence control) and strategy algorithms (supervisory). To model a software architecture Discrete Event/Time modelling is used, e.g. finite state machines for decisions and software modules (data-flow) for strategy algorithms. Control algorithms (I-b) endow a tighter notion of real-time, so that Discrete Time is used for loop-control algorithms and Continuous Time is used for plant modelling. These models are used in combination for verification and simulation (II) before software implementation synthesis (III).

[1] Luca Abeni, Giorgio Buttazzo : Integrating multimedia applications in hard real-time systems, Real-Time Systems Symposium (RTSS), 1998.

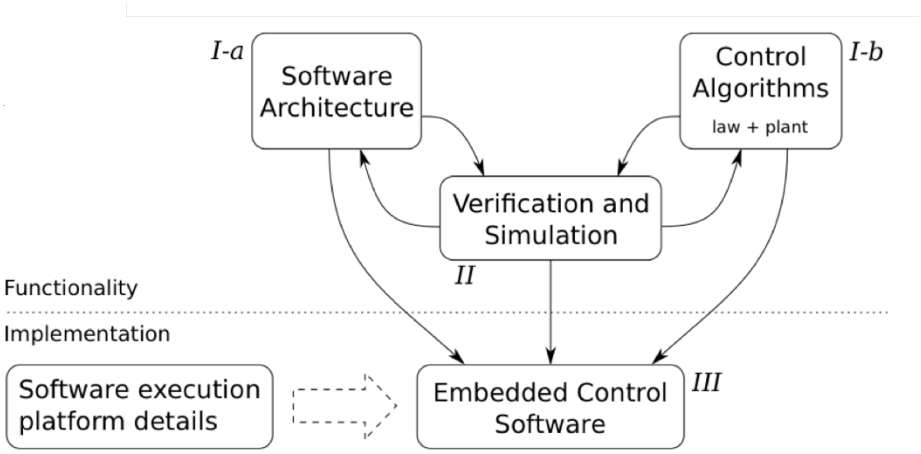


Figure 10: Combined Modelling and Design Process

Jan Broenink presents his specific choice for models, targeting graphical languages:

- Cyber part relies on a graphical representation of Communicating Sequential Processes (gCSP). One of the fundamental features of CSP is that it can serve as a notation for describing concurrent and communicating processes at different levels of abstraction, using different communication models as for instance Rendezvous communication.

- Physical part relies on Bond graphs. A bond graph is a graphical representation of a physical dynamic system. It allows the conversion of the system into a state-space representation.

Both graphical representations are similar to a block diagram or signal-flow graph; with the major difference that the arcs in bond graphs represent bi-directional exchange of physical energy, while those in block diagrams and signal-flow graphs represent directional flow of information.

Models are then support for a concurrent design flow where software, electronics, control and mechanics design run concurrently.

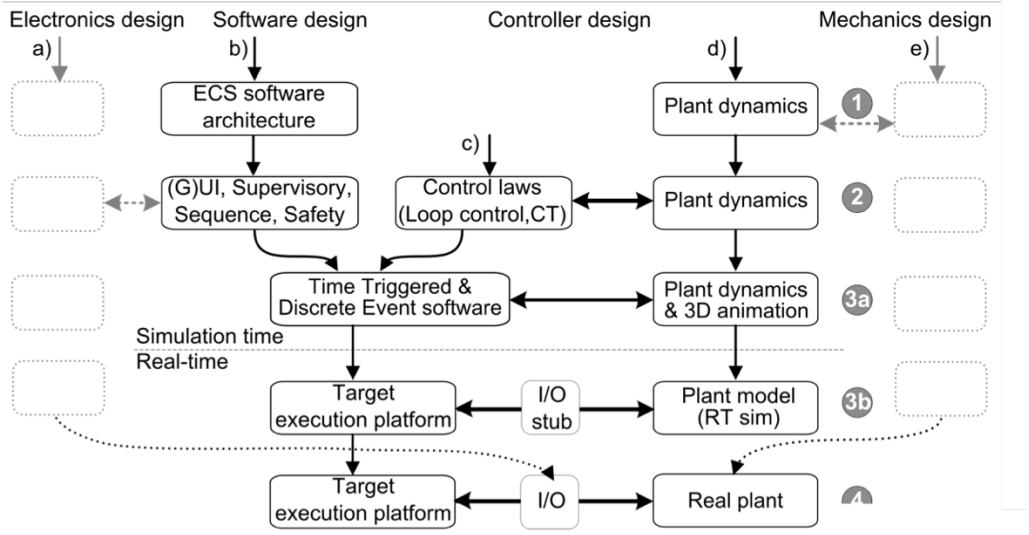


Figure 11: Concurrent Design Flow

The models are then refined step-wise

1. Architecture and Dynamic Behavior
2. Model-based control-law design
3. Software, functional co-simulation and real-time specification/simulation of the implementation

4. First-time right realization

While the importance of models is undeniable, it is important to make a distinction between languages vs techniques/methods vs tools to correctly use them during the design flow. Formalisms/Languages are instruments to exploit models: they provide expressiveness to write down models (syntax and semantics). Techniques, pertaining to the realm of model-driven engineering techniques, fall into two main categories: transformation techniques to transform models (expressed in a given formalism) to other models (possibly expressed in a different formalism) and techniques to give insights or retrieve information captured by the models. Methods have a larger scope than techniques, they pertain to reasoning frameworks or design approaches (e.g. the BIP framework presented by Saddek Bensalem). Tools, finally, support methods and implements techniques. Decoupling techniques/methods from tools allows focusing on methods instead of getting lost in tools implementation.

The link between languages and so-called meta-models is of paramount importance. A meta-model is a model of models, i.e. a meta-model defines the language used to write a model. A meta-model indeed specifies rules for checking the correctness of models and represents a basis for tools, specifically for editors and compilers. Common ground between different meta-models can be found in meta-meta-models (a meta-model conforms to a meta-meta-model), while transformations between models are ruled by a transformation among corresponding meta-models.

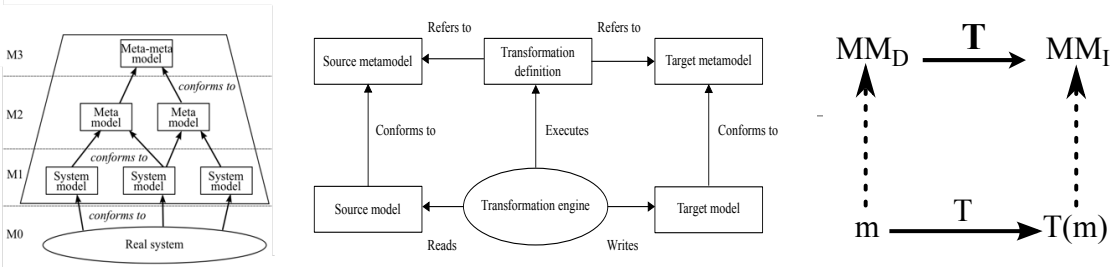


Figure 12: Models Pyramid and Model transformations

6.1.4 Jurgen Bock: Industry 4.0 as paradigm of digitally connected components.

Industry 4.0 (I4.0) is a term coined in Germany to refer to the fourth industrial revolution. This is understood as the application of concepts such as Internet of Things (IoS), Cyber-physical Systems (CPS), the Internet of Services (IoS) and data-driven architectures in the real industry.

6.1.4.1 Background: The I4.0 component and the reference model

The Reference Architecture Model for Industry 4.0 (RAMI 4.0 describes fundamental aspects of the Industry 4.0: it illustrates the connection between IT, manufacturers/plants and product life cycle through a three-dimensional space. Each dimension shows a particular part of these worlds divided into different layers as depicted in Figure 13. Left vertical axis represents IT perspective which is comprised of various layers such as business, functional, information, etc. These layers correspond to the IT way of thinking where complex projects are decomposed into smaller manageable parts. In the left hand, horizontal axis the product life cycle is displayed where Type and Instance are distinguished as two main concepts. The model allows the representation of the data gathered during the entire life cycle. Along with the right hand horizontal axis the location of the functionalities and responsibilities are given in the hierarchical organization. The model broadens the hierarchical levels of IEC 62264 1 by adding the Product or a workpiece level at the bottom, and the Connected World goes beyond the boundaries of the individual factory at the top.

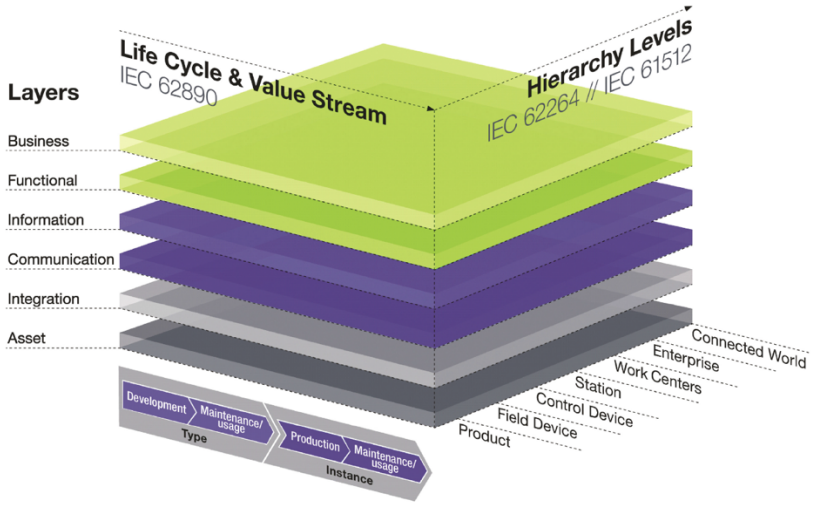


Figure 13: The reference model of Industrie 4.0

A component is a basic concept in Industry 4.0. It is used as a model for representing the properties of real objects in a production environment connected with virtual objects and processes (a CPS system). It is comprised of two foundational elements: one or more assets and Administrative Shell surrounding the assets (Figure 14).

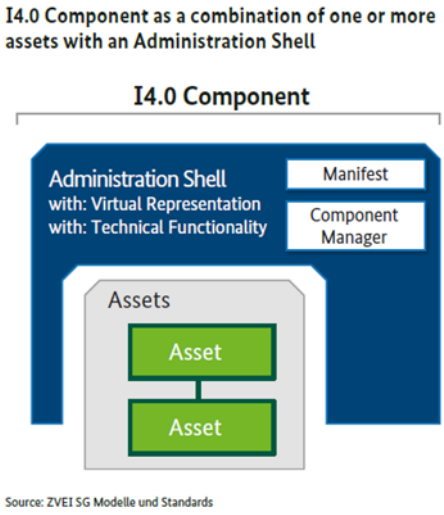


Figure 14: I4.0 component

An I4.0 component can be a production system, an individual machine, an assembly inside a machine or a software platform. Indeed, I4.0 component can be on different hierarchy levels (product, field device, ..., enterprise, connected world). A basic prerequisite is that I4.0 components are able to communicate and understand each other for cooperation scenarios. To this end I4.0 components need to have self-X properties (starting off with self-description) and need to interact. Implementation of interaction is often based on OPC-UA concepts.

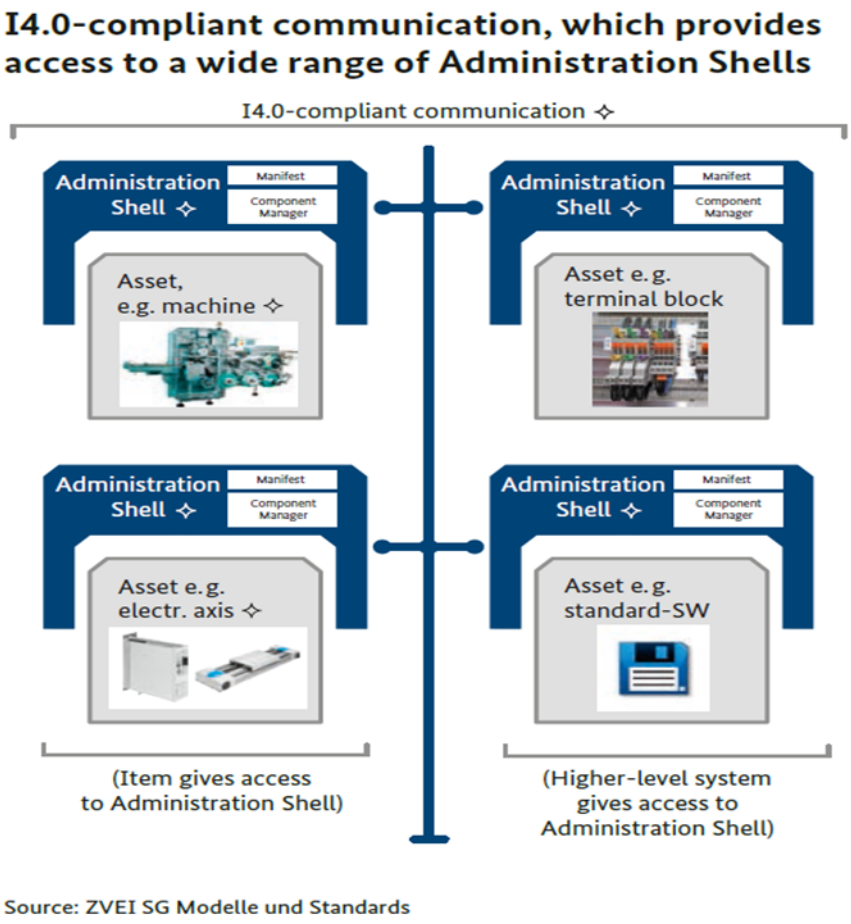


Figure 15: I4.0 components interaction

6.1.4.2 Interaction model and message-based communication

I4.0 components shall be able to interact with each other following an interaction model Figure 16. Within this model, we can find a set of typical interaction patterns, namely:

- Identification
- Negotiation of security measures
- Request for a task (is a particular task executable?)
- Negotiation of a task
- Order and execution of a task
- Report of errors

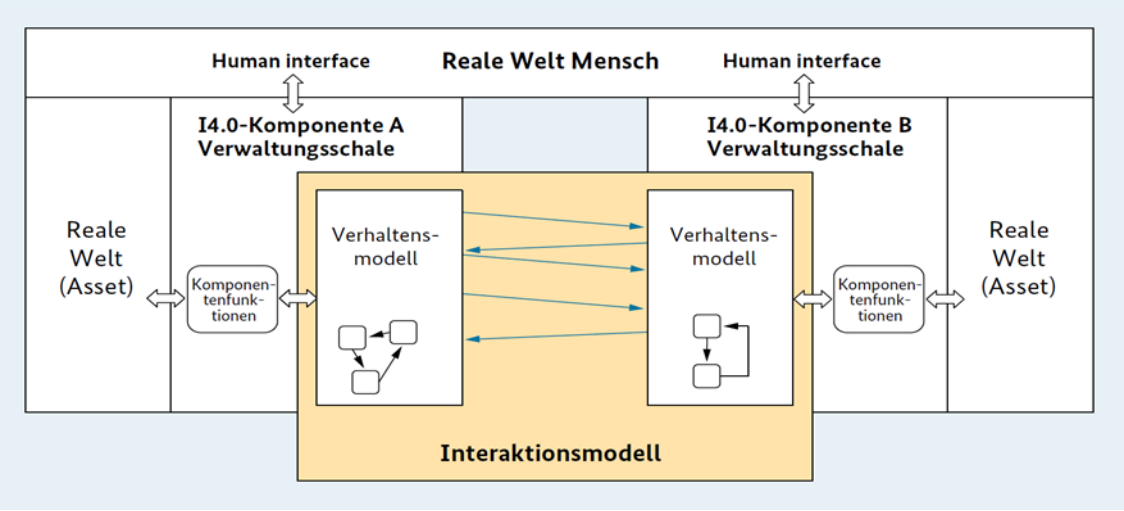


Figure 16: Interaction model[1]

The point-to-point communication is message-based. Both base ontology and domain specific ontologies are considered. Base ontology provides a vocabulary for message format description while the domain ontology provides a vocabulary for message content description.

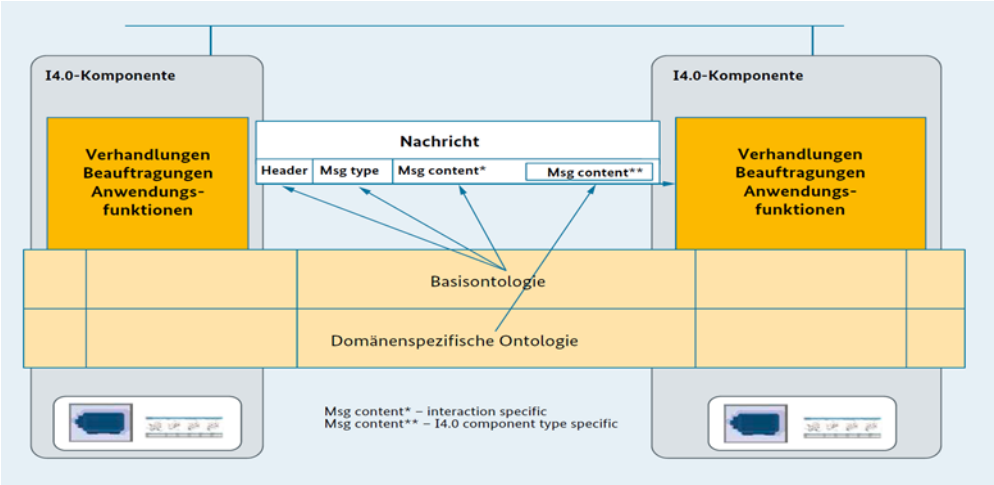


Figure 17: Message-based Communication[1]

6.1.4.3 OPC-UA

OPC-UA is often the preferred choice to implement communication between I4.0 components. OPC-UA has a client-server architecture, but a publish-subscribe architecture is currently under specification. One of the fundamental concepts in OPC UA is information modelling that can be used to describe the component (provide a structured access to data and methods). The semantics of information models is not formal, but there are companion standards that describe common structures for information models in specific domains or for specific purposes (e.g. Companion Standard "Device Integration" for field devices). OPC UA offers various services, such as security and discovery services.

6.1.4.4 Composition of Industry 4.0 components

Assets can be arranged freely (Figure 18); different composition patterns might be possible. Composition must obey to the following rules:

- Components to be composed should be Industry 4.0 components
- The Asset Administration Shell should be a standardized interface

[1] Bock, J.; Diedrich, C.; Hänisch, R.; Kraft, A.; Neidig, J.; Niggemann, O.; Pethig, F.; Reich, J.; Schulz, T.; Vollmar, F. & Vialkowitsch, J. Weiterentwicklung des Interaktionsmodells für Industrie-4.0-Komponenten Plattform Industrie 4.0, 2016

- There should be an interaction model for Industry 4.0 components
- Asset Administration Shells can contain sub-models and are thus flexible
- Sub-models can be used for various composition-related aspects, e.g. self-description, negotiation (QoS, tasks, etc.)
- The implementation should be based on OPC UA

The priority to start off with Asset Administration Shells based on OPC UA and subsequently implement sub-models.

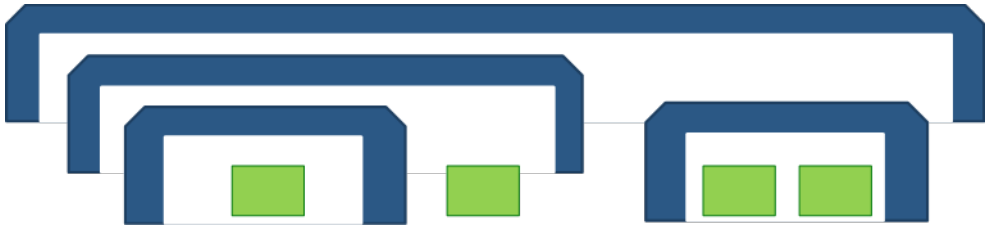


Figure 18: Composition of assets and components

6.1.4.5 Semantics and Vocabularies

The asset administration shell (AAS) is a generic interface to anything that has a value (asset) for an owner. Every AAS provides a basic set of information about the asset. An added value is only generated if the basic set of data is extended with domain specific data. Therefore, in the header of the AAS a statement is made that the AAS supports a specific model.

In a first step, semantics will be based on shared vocabularies. Current parameters and states are provided by "properties", more precisely, by property value statements. Every property is defined in a dictionary, e.g., ecl@ss or Common Data Dictionary (CDD). In these dictionary, it is actually defined what for example a "pipe diameter" is.

Expressive formal semantics is currently missing but AAS offer a way to incorporate expressive formal semantics: Messages in an interaction model can refer to externally declared properties, e.g. in eCl@ss, or more expressive ontologies. At implementation level this is immediately reflected through OPC UA information models linking to expressive ontologies, OWL, etc.

Currently, different groups are looking for domain specific properties (e.g., drive engineering) in order to insert them into dictionaries like ecl@ss.

6.1.4.6 QoS management

Let a formal description for QoS be part of the Asset Administration Shell through the definition of a sub-model, then use the interaction model in combination with the sub-model to negotiate QoS properties.

6.1.4.7 Tooling

While the AAS specification is pretty advanced, tools are not yet available.

6.2 Current design methodologies assessment

In this section, we report the assessment that each expert provided on current development methodologies. The assessment pertains to the methodologies presented by each expert. For each methodology, the consortium asked for potential risks in applying the methodology, current pain-points, the focus and priorities covered, the suggested good practices, tooling and what in the opinion of the expert are bad practices. Recommendations for the RobMoSys projects are then presented in the last column.

Risks	Pain Points	Priorities	Bad practices/wasted time	Good Practices	Tools	Recommendations
SADDEK						
Verification complexity	State space explosion	- Correct-by-construction Design - Composability rules	Existing development methodologies are of limited interest for systems. They prescribe only general principles and fail to provide rigorous support and guidance. Correctness-by-checking contributes to trustworthiness but it is limited to requirements that can be formalized and checked efficiently (mainly verification of functional properties for application software). For the same reasons, its application to optimization requirements is limited to the validation of scheduling and resource management policies on abstract system models.	- Component/model-based design - Compositional Verification - Assume-Guarantee approach - Incremental Verification - Model checking is applied only to medium size systems when it is possible to make automated proofs or when the cost of faults is high.	- BIP framework - Metropolis Framework	Putting Correctness-by-Construction into Practice: - Horizontal correctness: the construction process of component C is bottom-up. Increasingly complex composite components are built from atomic components by using glue operators. Two principles can be used in this process to obtain a component meeting P: property enforcement and property composability. - Vertical correctness, we need to develop component-refinement tools to allow downstream movement in the abstraction hierarchy from application software modeled in the chosen component framework. - The requirement to deal with a small number of types of components is essential for the formalization of the composition rules between components. Frameworks with a large number of types of components are badly amenable to formalization.

Figure 19: Saddek Bensalem's Assessment

Risks	Pain Points	Priorities	Bad practices/wasted time	Good Practices	Tools	Recommendations
JAN						
Unbalanced solutions due to unawareness of capabilities and restrictions of computing resources	Not appreciating / understanding the value of composition / architecture from a reusability point of view. Application experts and component providers still have difficulty understanding each other.	Vocabulary / ontology; Explain stacks / concerns such that both application experts and component providers can understand each other and benefit from each other; Provide instruments to let roboticists get benefit from the above	Ignore the effect of implementations (i.e. non-idealness of computers / networks). Ignore the viewpoint of the roboticists whose focus is primary on computation	Use declarative models: Bond Graphs, Communicating Sequential Processes in my case Verification and Co-simulation, especially during design. Step-wise refinement as design "guide"	gCSP model checker / co-simulation 20-sim for bond-graph models and control law design. Automatic code Generation from block diagrams Execution lib providing concurrent execution ROS etc, DDS	Separate the language, and methods from the Tools Push way of working as: Model declarative; Refine stepwise; Automatically generate procedural code; Verify / Co-simulate during design…

Figure 20: Jan Broenink's Assessment

Risks	Pain Points	Priorities	Bad practices/wasted time	Good Practices	Tools	Recommendations
ARNE						
Artificial Complexity Demonstrator driven implementation, lack of systematic engineering practices following predefined structures Reuse/portability of SW components not widely recognized as design goals in robotics	SW components implemented with implicit assumptions which are only true on specific platforms (e.g. ROS) → SW components not portable among target platforms No separation of concerns: SW component implementations are overly complex because they cannot assume any guarantees (e.g. QoS) from the underlying platform Missing link (or mutual ignorance) between function and performance	Consistent vocabulary of the different domains, the structuring of the motion stack serves as guiding example Right Abstractions e.g. Abstract Machine Interfaces to handle QoS attributes on application level. Thereby most importantly, abstractions for handling computational / communication resources Definition of the computational model on system level (i.e. how and when do components exchange data, when are computations executed, etc.) Definition of end-to-end timing constraints for cause-effect-chains spanning multiple SW components Configuration and variability of SW components (context: using a generic component in the context of different robots) Safety handling	Models with non-adequate abstraction level: e.g. Code Generation from UML models (too concrete), Boxes and Lines (too unconcrete) Non-constructive approaches (too many iterations) Approaches by-passing architectural patterns (e.g. complex drivers in AUTOSAR)	Morphological method to structure a functional domain Logical Execution Time for composability and portability on communication level (good 4 certification) Reservation Based Scheduling Schemes for composability on computational level (good 4 certification) Methodology driven structured engineering approaches such as AUTOSAR in automotive (freedom from choice) Declarative models including methods/tools to derive procedural realizations GALS structure on system level. Ignore "minor" coupling effects on functional level to derive flexible implementation with sufficient degrees of freedom: "approximate refinement".	Essential Analysis Tool LITMUS^RT as backbone for execution container Tool for specifying cause-effect-chains: as link between the functional domain and the execution domain Real-time analysis tools like SymTA/S, Inchron	Ask to extension to existing implementations e.g. LINUX extension for RBS Developed concept should be technology agnostic but show cased using concrete technology like ROS, ROS2.0 Different aspects in the call should be illustrated with concrete problems (cf. end-2-end latency with CFS vs RBS) Prefer declarative approaches over procedural approaches, since the former better supports composability Search for executable models (i.e. defining clear computational models) on component level. Ask explicitly for quantitative evaluation of the proposed method/tool/approach compared to the current state of practice (i.e. show the benefit)

Risks	Pain Points	Priorities	Bad practices/wasted time	Good Practices	Tools	Recommendations
ARNE						
Artificial Complexity Demonstrator driven implementation, lack of systematic engineering practices following predefined structures Reuse/portability of SW components not widely recognized as design goals in robotics	SW components implemented with implicit assumptions which are only true on specific platforms (e.g. ROS) → SW components not portable among target platforms No separation of concerns: SW component implementations are overly complex because they cannot assume any guarantees (e.g. QoS) from the underlying platform Missing link (or mutual ignorance) between function and performance	Consistent vocabulary of the different domains, the structuring of the motion stack serves as guiding example Right Abstractions e.g. Abstract Machine Interfaces to handle QoS attributes on application level. Thereby most importantly, abstractions for handling computational / communication resources Definition of the computational model on system level (i.e. how and when do components exchange data, when are computations executed, etc.) Definition of end-to-end timing constraints for cause-effect-chains spanning multiple SW components Configuration and variability of SW components (context: using a generic component in the context of different robots) Safety handling	Models with non-adequate abstraction level: e.g. Code Generation from UML models (too concrete), Boxes and Lines (too unconcrete) Non-constructive approaches (too many iterations) Approaches by-passing architectural patterns (e.g. complex drivers in AUTOSAR)	Morphological method to structure a functional domain Logical Execution Time for composability and portability on communication level (good 4 certification) Reservation Based Scheduling Schemes for composability on computational level (good 4 certification) Methodology driven structured engineering approaches such as AUTOSAR in automotive (freedom from choice) Declarative models including methods/tools to derive procedural realizations GALS structure on system level. Ignore "minor" coupling effects on functional level to derive flexible implementation with sufficient degrees of freedom: "approximate refinement".	Essential Analysis Tool LITMUS^RT as backbone for execution container Tool for specifying cause-effect-chains: as link between the functional domain and the execution domain Real-time analysis tools like SymTA/S, Inchron	Ask to extension to existing implementations e.g. LINUX extension for RBS Developed concept should be technology agnostic but show cased using concrete technology like ROS, ROS2.0 Different aspects in the call should be illustrated with concrete problems (cf. end-2-end latency with CFS vs RBS) Prefer declarative approaches over procedural approaches, since the former better supports composability Search for executable models (i.e. defining clear computational models) on component level. Ask explicitly for quantitative evaluation of the proposed method/tool/approach compared to the current state of practice (i.e. show the benefit)

Figure 21: Arne Hamann's Assessment

Risks	Pain Points	Priorities	Not Relevant	Bad practices/wasted time	Good Practices	Tools	Recommendations
JURGEN							
Being too abstract for system integrators, end users, component providers, ..., to transfer into real systems	Currently it's a difficult ongoing discussion, on where and how the asset administration shell is being implemented. There should be a distinction between the role, type, instance, and real physical asset of any I4.0 component. Strictly speaking, the role "robot with properties 123, type "KUKA KR6", instance "KUKA KR6 with S/N 987", and the real physical robot, are three assets, but do they all need administration shells? (These are ongoing discussions in I4.0 consortia and projects.)	Detailed specification and implementation of the Asset Administration Shell Base Ontology for interactions (Messages) Abstract communication primitives Semantic interoperability Coordination (interaction models specified as state machines) Safety and security	Real-time (in many higher level coordination tasks) Lower levels (Hardware, OS, Exec. Cont.) might not have to be seen as I4.0 components, as they are managed by the asset providers	Trying to be too disruptive on the shopfloor (legacy systems) will not be accepted	Quickly get to the point of having an implementation (with quantifiable analysis) Agreeing on a particular standard toolchain is essential (e.g. OPC UA in Industry 40)	OPC UA (SDKs and software stacks), tools for authoring OPC UA information models (e.g. UaModeller) Tools for creating / configuring I4.0 concepts, e.g. AAS, Interaction Manager, etc.	Focus on small and concrete use cases (concrete use case requested!)

Figure 22: Jurgen Bock's Assessment

6.3 Synthesis on recommendations for RobMoSys

In this section, we provide a synthesis of the Experts recommendations relevant for the RobMoSys Project.

A first aspect pertains to the nature of the models employed. From experts' recommendations, we can derive the following conclusions:

- Prefer declarative models (including methods/tools to derive procedural realizations) over procedural models since declarative models better support composability;
- Definition of a model of computation (MoC) on system level, i.e. how and when do components exchange data, when are computations executed, etc. The use of a MoC enables verification and co-simulation during design.
- Enable refinements to allow downstream movement in the abstraction hierarchy. GALS (Globally Asynchronous and Locally Synchronous) is a paradigm on system-level allowing refinements with a certain degree of freedom towards implementations.
- Small number of types of components in order to manage the formalization of composition/composability rules and properties.

A second aspect is related to good practices to employ during the development process:

- Put correctness by construction into practice supporting both bottom-up construction (component built out of atomic components) and top-down approaches (refinements) via composition rules and step-wise correct refinement of components
- Push verification and co-simulation activities during design; use model-checking only on small models, composable/incremental verification
- Push the concept of concurrent design (e.g. mechanical, electronical, software)
- Handle safety and security aspects as soon as possible and not as an afterthought

A third aspect pertains to tooling

- Separate the language from the methods and from the tool
- Agree on a standard tool-chain and a common vocabulary for interoperability: at least syntax, better to have semantic interoperability
- Make use of tooling to formalize and assess the structuring of domain knowledge: low dependency/overlapping between concepts

Final remarks have been also provided for use-cases

- Focus on small and concrete use cases
- Developed concepts should be technology agnostic but show-case using concrete technology (e.g. real OS, middleware, etc.)
- Provide assessments to show the benefit of the concept/method/tool with respect to current state of the art

Annex 2 – Annex I sent to the EC

Project acronym: RobMoSys

Grant agreement number: 732410

Project full name: Composable Models and Software for Robotics Systems

Project RobMoSys, co-funded from the European Union's Horizon 2020 research and innovation programme under agreement No 732410, foresees as an eligible activity the provision of financial support to third parties, as means to achieve its own objectives.

The vision of RobMoSys is to create **better models, as the basis for better tools and better software, which then allow to build better robotic systems**.

The project asks for contributions that **realise a step change** in system-level composition for robotics, and that demonstrate this in **real-world scenarios**. The step change must not only be visible in the modelling foundation of the contributions, but also in the **industry-grade quality** of their realisation. Indeed, in the medium-term future, companies should be able to rely on the RobMoSys outcomes to build robotic applications by composing high quality composable models and associated software functions.

Proposals need to illustrate their contribution in a relevant **use-case** with coverage of *all of the following*: **tooling**, **models** and associated **software** (implementations that realise the models, and that are created/configured by the tooling) demonstrated on **system**-level prototypical scenarios in, e.g., navigation and manipulation.

To achieve this goal, types of activities that qualify for financial support are software developments under the form of:

- **Models**
  - Composable models of components (ports, blocks, connectors enriched with composition constraints, resource requirements, etc.).
  - Models of system-level composition (system composed out of models of components) within a relevant use-case (composition for design-time or run-time composability).
  - Models to realise an architectural pattern, a design principle or best practice.
- **Tools and Meta-Models**
  - Extensions and/or improvements of, the provided RobMoSys meta-models (for instance for additional non-functional concerns such as Quality of Service, timing, performance, etc.).
  - Extensions and/or improvements of, the provided RobMoSys tools baseline (e.g. for design-time predictability, sanity checks, composability analysis, formal conformance verification, etc.). The current RobMoSys tools baseline is available here: http://robmosys.eu/wiki/baseline:start

It is crucial that the contributions to the RobMoSys ecosystem strictly adhere to the RobMoSys modelling principles (composability, and conformity to meta-models). Full open source contributions are preferred but not mandatory. However, we expect at least the models and their transformations to proprietary tools to be under an open source license. Let us also note that in the first open call, we prefer projects that illustrate their contribution in the domain of robot-centric motion, navigation and manipulation. The RobMoSys technical user-stories (see http://robmosys.eu/wiki/general_principles:user_stories) provide a variety of possible topics that RobMoSys encourages to consider.

Because of the expected step change contributions, the Call welcomes, in particular, consortia offering complementary, multi-disciplinary competences that go beyond the mainstream robotics community; for example, robotics experts teaming up with software engineering people, or tool builders, or experts from automotive, aerospace, embedded, cyber physical systems.

More information and the full call documents, including the guides for applicants and an electronic submission system, can be found on the web site www.robmosys.eu.

Call identifier: RobMoSys-1FORC

Call title: First Open Call for RobMoSys Contribution

Publication date: 10.07.2017

Deadline: 09.10.2017, 17:00 Brussels time

Expected duration of participation: 12 months

Indicative budget for the call: €2,000,000

Maximum funding request per proposal: 300,000 €

Submission language: English

Web address for full open call information: http://robmosys.eu/open-calls/

E-mail: opencalls@robmosys.eu

Annex 3 – Call text

[to be added after 10th July 2017, in the updated version of deliverable D5.1.]

Annex 4 – Guide for applicants

[to be added after 10th July 2017, in the updated version of deliverable D5.1.]

Annex 5 – Proposal template

[to be added after 10th July 2017, in the updated version of deliverable D5.1.]

Annex 6 – Good practices and templates for organizing open calls under the H2020 Financial Support to Third Parties scheme

1. Introduction

Your call should be carried out in the light of the same basic principles which govern Commission calls:

i. **Excellence.** The proposal(s) selected for funding must demonstrate a high quality in the context of the topics and criteria set out in the call;
ii. **Transparency**. Funding decisions must be based on clearly described rules and procedures, and all applicants should receive adequate feedback on the outcome of the evaluation of their proposals;
iii. **Fairness and impartiality.** All proposals submitted to a call are treated equally. They are evaluated impartially on their merits, irrespective of their origin or the identity of the applicants[1];
iv. **Confidentiality.** All proposals and related data, knowledge and documents are treated in confidence;
v. **Efficiency and speed.** Evaluation of proposals and award of the financial support should be as rapid as possible, commensurate with maintaining the quality of the evaluation, and respecting the legal framework.

1. Preparation activities

The Call Announcement

You should prepare a brief announcement about the call (you may use the model included in Annex 1 of this document) which will be published on the Horizon 2020 Participants Portal, and on the project website. It contains a link to the section on the project website where the full call details are published. In order to ensure timely publication on the Participant Portal, please provide the call announcement at least 30 days prior to its foreseen date of publication to your Project Officer.

The Full Call Details

You should prepare a dedicated section of your project's website, which will give proposers the Full Call Details. This must be in line with the specific requirements of the work programme and contain:

- A clear and exhaustive list of the types of activities that qualify for receiving financial support.
- Any restrictions on participation in any part of the call (e.g. only certain types of organisation are required, only organisations based in certain countries etc.). Please note that the calls must have a clear European dimension which can be achieved either through cross border experiments or through expanding local experiments to European scale.
- The criteria determining the award of the financial support.
- The criteria for determining the exact amount of financial support and the form that the financial support may take (e.g. a lump sum – either pre-defined or based on estimations of the grant recipient - or the reimbursement of actual costs incurred by the recipients when implementing the supported activities).
- The specific arrangements that the beneficiaries may impose on the third parties (e.g. specific reporting and feedback obligations from the third party towards the beneficiary in respect to the implementation of the supported activities; specific arrangements for providing the financial support; specific rights for the beneficiaries to access and use the results of the supported activities).

[1] In the frame of any restrictions provided for in the call

- **The information needed to submit a proposal**
    - **The template to be used for the proposals**
    - **The coordinates (email address and telephone number) of a help facility which you must maintain for proposers during the call**
    - The email address to which proposals should be submitted and the call identifier which will be used on these emails
    - The deadline for proposal submission, clearly specifying the local time involved (normally this is local time at the website where the proposals are received).

2. Publication of the call

Following the requirement of the General Annex K of the Work Programme, you will publish the Full Call Details, at least, on the project's own website.

Your Project Officer will arrange to publish the Call Announcement and (a reference to) the Full Call Details on the dedicated web page of the Horizon 2020 Participants Portal.

The call must remain open for the submission of proposals for a period of at least three months. If call deadlines are changed, this must immediately be communicated to the Project Officer for updating the Call Announcement on the Horizon 2020 Participant's Portal. The Full Call Details must be updated on the project's own website and all registered applicants must be informed of the change.

Please make sure that all proposers receive fair and equal treatment. Information or facilities which you supply to any proposer must be equally available to all.

3. Proposal reception

Proposals should be submitted through an electronic exchange system which allows the identification of the time of submission. On receipt of each proposal you should send an Acknowledgment of receipt to the proposer (see example in Annex 2).

You may not accept late submissions; late submitters should receive by return email a "call closed" message from you.

You should evaluate the proposals as submitted: after the call closure no additions or changes to received proposals should be taken into account.

4. Proposal evaluation and selection

Evaluation criteria and procedure

You will evaluate proposals received in the light of the criteria laid down in the Full Call Details. You may use the attached form (see Annex 3).

You remain responsible for the evaluation towards the proposers, even though you may count on the assistance of experts[1].

If you engage experts for evaluating the proposals, please ensure that they are independent from the organisations involved in the consortium and from any proposer.

The selected experts should sign a declaration of confidentiality concerning the contents of the proposals

[1] The selection of these experts should follow the conditions foreseen in Article 10 of the Model Grant Agreement.

they read and they should also confirm the absence of any conflict of interest (see an example of such declaration in Annex 4).

The outcome of the evaluation will be a ranked list of all proposals, based on the scores obtained by each proposal.

Proposal selection

Whilst normally the highest ranked proposals will be selected for funding, there might be objective reasons for objecting to a specific third party, for example commercial competition. In this case the choice may pass to the next-ranked proposal.

You may conclude that even the highest scoring proposal is of inadequate quality, in which case you will make no selection. This conclusion is obligatory if all the proposals fall below the threshold scores applied at the evaluation.

In the event of no selection being made, you may re-open the call at a later date. Alternatively, you may conclude that no successful outcome can be expected and abandon the plan to hold an open call. This decision would have to be justified and be the subject of a grant agreement amendment.

5. Reporting, documentation and feedback

Reporting

Shortly after the evaluation you should publish a **public summary report** of the evaluation results on your project website within 30 days of the end of evaluation taking into account your feedback process to the proposers (i.e. the proposers should have received your individual feedback before the public summary report is published). This report should comprise an account of the call, its evaluation and its results, including dates of call, how it was published, dates of evaluation, number of proposals received, number of proposals funded, as well as a list of all selected proposers and their funding amounts (you may use the model included in Annex 5).

Documentation

Additionally to the summary report you have to keep your internal records on the evaluation as audit trail in case of e.g. contestations by proposers, audits, or checks by the commission. These records comprise as a minimum:

- A listing of proposals received, identifying the proposing organisations involved (name and address).
- All received proposals
- All communications with applicants before call closure and during evaluation
- The names and affiliations of the experts involved in the evaluation;
- For each proposal a copy of the filled forms used in the evaluation;
- A record of all incidents which occurred during the evaluation (e.g. how conflict of interest were handled if they were detected during the evaluation process) and any deviation from standard procedure (e.g if a proposer selected was not the highest scoring one, you must document the objective reasons why the highest scoring one was passed over)

Feedback to proposers

After the evaluation of the proposals, you will get into contact with the successful proposer(s).

You should communicate to the other proposers that their proposal was not successful in the call, and should enclose to each a summary of the evaluation result of their proposal addressing the respective

award criteria.

Annex 1 – Call announcement format

Announcement of an open call for recipients of financial support

Project **acronym:** XXX
Project **grant agreement number:** XXX
Project **full name:** YYY

Project XXX, co-funded from the European Union's Horizon 2020 research and innovation programme under grant agreement No XXX, foresees as an eligible activity the provision of financial support to third parties, as a means to achieve its own objectives.

The types of activities to perform that qualify for receiving financial support are XXX.

Deadline: XXX
Expected duration of participation: XXX
Maximum amount of financial support for each third party: XXX
Call identifier: XXX call
Language in which proposal should be submitted: XXX
Web link for further information (full call text/proposal guidelines/call results) on your official project web site: www.xxx-project.eu/xxx
Email address for further information: XXX@XXX.com

Annex 2 - Acknowledgment of receipt

Acknowledgement of receipt

Dear XXX,

Thank you for submitting your proposal for consideration as recipient of financial support in the frame of project XXX.

The evaluation of all proposals received will take place in the next few weeks. You will be notified as soon as possible after this of whether your proposal has been successful or not.

On behalf of my colleagues in the project I would like to thank you for your interest in our activities.

Yours sincerely,

Annex 3 – Evaluation form

Individual evaluation/Consensus (delete as appropriate)

Proposal No. :	**Acronym :**

1. Award criterion 1	**Score:** *(Threshold 3/5; Weight)*[1]
2. Award criterion 2	**Score:** *(Threshold 3/5; Weight 1)*

[1] Thresholds and weights are standard values which can be adapted to the needs of the specific evaluation, if necessary

0 The proposal fails to address the criterion under examination or cannot be judged due to missing or incomplete information; 1 Poor The criterion is addressed in an inadequate manner, or there are serious inherent weaknesses; 2 Fair While the proposal broadly addresses the criterion, there are significant weaknesses; 3 Good The proposal addresses the criterion well, although improvements would be necessary; 4 Very good The proposal addresses the criterion very well, although certain improvements are still possible; 5 Excellent The proposal successfully addresses all relevant aspects of the criterion in question. Any shortcomings are minor.

3. Award criterion 3	**Score:** *(Threshold 3/5; Weight 1)*
Remarks	**Overall score:** *(Threshold 10/15)*

I declare that, to the best of my knowledge, I have no direct or indirect conflict of interest in the evaluation of this proposal

Name	
Signature	
Date	

Name	
Signature	
Date	

0 The proposal fails to address the criterion under examination or cannot be judged due to missing or incomplete information; 1 Poor The criterion is addressed in an inadequate manner, or there are serious inherent weaknesses; 2 Fair While the proposal broadly addresses the criterion, there are significant weaknesses; 3 Good The proposal addresses the criterion well, although improvements would be necessary; 4 Very good The proposal addresses the criterion very well, although certain improvements are still possible; 5 Excellent The proposal successfully addresses all relevant aspects of the criterion in question. Any shortcomings are minor.

RobMoSys

Annex 4 – Confidentiality and conflict of interest declaration

I the undersigned declare that, in participating as an independent expert in the evaluation of proposals received in the open call of project XXX

I undertake to treat as confidential all information contained in the proposals which I am asked to evaluate, both during the evaluation and afterwards.

I will not reveal to any third party the identity or any details of the views of my fellow evaluator(s), neither during the evaluation nor afterwards

I do not, to the best of my knowledge, have any interest in any of the proposals submitted in this call, I have not been involved in their preparation and I do not benefit either directly or indirectly from the eventual selection. Should I discover a conflict of interest during the evaluation, I undertake to declare this and to withdraw from the evaluation.

Name	
Signature	
Date	

THIS PROJECT HAS RECEIVED FUNDING FROM THE *EUROPEAN UNION'S HORIZON 2020 RESEARCH AND INNOVATION PROGRAMME* UNDER GRANT AGREEMENT NO. 732410

Annex 5 - Public evaluation report

Results of open call (call ID ref XXX) for recipients of financial support

Project **acronym:** XXX
Project **grant agreement number:** XXX
Project **full name:** YYY

Project XXX, co-funded from the European Union's Horizon 2020 research and innovation programme under grant agreement No XXX, launched an open call (call ID ref XXX) for recipients of financial support.

The call closed on XXX.

A total of XXX proposals were received for this call. XXX proposals will receive funding for a total amount of XXX EUR.

The evaluation and selection has been completed. All proposers have been informed about the evaluation results for their proposal for financial support.

Call information

The call was published on project XXX's website (URL XXX) and on the Horizon 2020 Participants Portal (URL XXX) on XXX. Full call details were published at: (URL XXX)

Please add any other location where the call was published (if any) or any other relevant information.

Response to the call in detail[12]

	Number of proposals	Funding requested
Proposals received		
Eligible proposals		
Proposals above threshold		
Selected proposals		

List of selected proposals

Organisation	Country	Funding awarded

[12] If different activities where called for, repeat this table for each activity.