68th International Astronautical Congress (IAC), Adelaide, Australia, 25-29 September 2017.


IAC-17,D3,4,7,x40657

# THE HARDWARE DEVELOPMENT TOOL STACK FOR FUTURE SPACE EXPLORATION

**Simon Vanden Bussche*[a], Marco Witzmann*[b], Louise Lindblad*[c]**

** Valispace UG (haftungsbeschränkt), Hermann-Köhl-Str. 7, 28199 Bremen, Germany*
*[a] simon@valispace.com, [b] marco@valispace.com, [c] louise@valispace.com*

## Abstract

In modern software development, companies select a number of programming languages, tools and frameworks for every project; the "stack". The benefits and usefulness of each individual framework are analysed for power and suitability, but also the integration of these tools is evaluated, to ensure a powerful low-friction development environment. This involves the complete chain of team communication, bug tracking, requirements management, development, configuration management, deployment, customer support and maintenance.
Analogously, in "New Space" hardware development, lean and collaborative engineering approaches are necessary to cope with the rising demands towards disruptive reduction of costs and increase in speed of space projects. A radically new approach is necessary, breaking with classical processes and tools, which have grown as a patchwork in most companies over decades. Instead, new tools and most importantly their interactions and interfaces need to be analysed to fundamentally improve the efficiency of hardware development. This is all the more relevant for an international collaborative space exploration effort.
In a first step, this paper identifies the landscape of functionalities, needed by space companies and agencies for their technical operations: requirements management, Computer Aided Design (CAD), engineering data management, simulation, reporting, software development, testing, verification management, internal and external communication, task management, etc. Then the identified functions are put into relation, identifying the different interfaces and relationships among them.
As a second step an analysis shows, why Model Based Systems Engineering (MBSE) has not been able to fulfil the needs of engineers across disciplines. An alternative solution to the problem is presented: A browser-based hardware development tool stack as a bottom-up solution, which combines functionalities of different existing software solutions. As an example stack, 11 tools are shortly presented which include collaborative browser-based 3D software, Kanban tracking tools, concurrent engineering data management systems, chat-based communication channels, etc.
An outlook is given to benefits and risks of the use of such a tool stack, as well as the potential integrations of such tools. It becomes apparent in which areas and at which interfaces modern tools have the highest impact, to paint a roadmap also for major "Old Space" players of how to transform their engineering processes in a way that will enable cost effective development of ever more complex spacecraft in the future, allowing for space exploration without breaking the bank.

**Keywords:** Model Based Systems Engineering (MBSE), Hardware Development Tools, Concurrent Engineering, New Space, Space Exploration

**Acronyms/Abbreviations**

| | |
|---|---|
| AOCS | Attitude and Orbit Control System |
| API | Application Programming Interface |
| CAD | Computer Aided Design |
| CFD | Computational Fluid Dynamics |
| DHS | Data Handling Subsystem |
| ECSS | European Cooperation for Space Standardization |
| FEA | Finite Element Analysis |
| IEEE | Institute of Electrical and Electronics Engineers |
| INCOSE | International Council on Systems Engineering |
| MBSE | Model Based Systems Engineering |
| PDM | Product Data Management |
| PLM | Product Lifecycle Management |
| PM | Project Management |
| REST | Representational state transfer |

## 1. Introduction

As became clear in the late 1950s, space exploration requires dedicated methods and tools due to its complexity and the combination of multiple disciplines. To build the Saturn V rocket, more than 400,000 engineers [1] used a then newly invented document based engineering approach to coordinate their efforts. In the past decades, engineering has moved into the digital age, but although these documents are now prepared on computers instead of typewriters, they still form the central part of any modern engineering process [2]. Documents by their nature are static and their content is rapidly outdated in a dynamic engineering effort. The sheer amount of documents makes it impossible to track inconsistencies among them which can lead to catastrophic results, such as the loss of the NASA Mars Orbiter due to an inconsistency between metric and imperial units [3].

This paper shows that today documents are oftentimes used as a communication device between island solutions of digital engineering tools. Copy-pasting of data between documents does not only lead to an error prone process, but also results in high costs of spacecraft development since data along the engineering lifecycle, as well as between disciplines, are not coupled and have to be maintained manually.

## 2. Methodology

The lifecycle of spacecraft design according to the ECSS-E-ST-10C standard [4] is analysed and the necessary engineering steps are broken down into a landscape of tasks. Resulting from this, tool functionalities needed by space companies and agencies for their technical operations are deduced. Also, the data interactions within and across the different disciplines are presented. While this approach does not guarantee completeness and cannot take into account the specifics of each mission, it gives a broad structure to the most significant tasks in the disciplines involved in spacecraft engineering and the relation between them.

In the analysis, typical approaches and tools which currently help engineers fulfil these tasks are identified.

In a second step of the analysis, a set of modern collaboration tools is presented. This web-based tool set is evaluated in terms of the potential to replace typical tools used today and streamline the engineering process through increased tool interoperability.

## 3. Theory

### *3.1. Engineering Tasks identification*

By studying the ECSS-E-ST-10C standard, the following tasks along the engineering lifecycle of a spacecraft have been identified:

- Requirements engineering
- Computer Aided Design (CAD)
- Calculation
- Simulation
- Software engineering
- Testing
- Verification and quality management
- Data management
- Documentation and reporting
- Internal and external communication
- Project management

Some tasks, such as project management, are not necessarily at the core of engineering activities per se, but have very strong interactions with the engineering work on a daily basis.

While each task in itself is well known and studied, the interaction between these tasks and their domain specific tools presents a major difficulty even for experienced companies.

### *3.2. Data Interaction Analysis*

There are two types of interactions between tools and disciplines during the development of a spacecraft, both which require data interaction and present a possible risk of inconsistencies in the design. Firstly, looking at the design tasks within a specific discipline, the design evolves significantly throughout the design phases of a project. Many iterations have to be made within the same discipline over time to arrive at a feasible result. Secondly, there are interactions and dependencies between each of the design disciplines which have to be taken into consideration at any given phase of the design, so that the right assumptions are made.

#### *3.2.1. Iterations Within a Discipline*

A spacecraft design constantly evolves and changes during its development lifecycle. Whenever technical problems arise at different points in time of the process, feedback loops will lead to adaptations and corrections in the design (Fig. 1)

The major difficulty arising from these feedback loops and iterations throughout the design phases is to maintain consistency between data, knowledge management over time, as well as the impact analysis of any feedback to the spacecraft design within that discipline. Each identified task and its corresponding tool must ensure consistency within the own design domain.

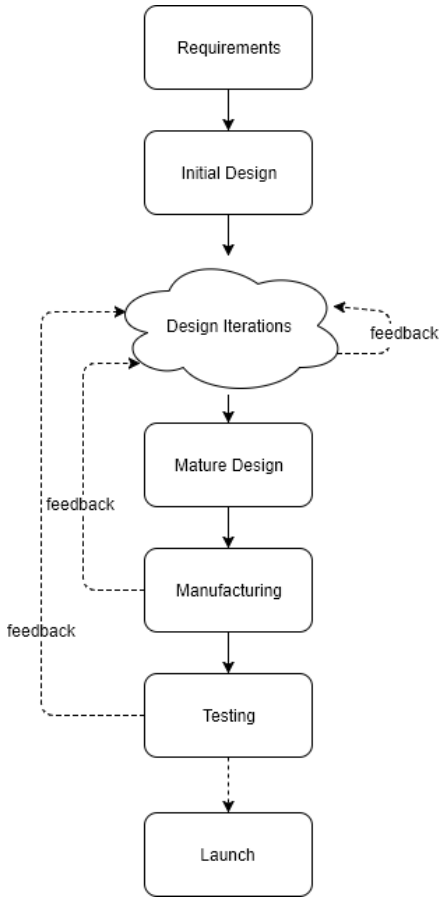


Fig. 1. Simplistic design steps for spacecraft design, including feedback loops

*3.2.2. Interdisciplinary Dependencies*

The complexity of spacecraft leads to the need for a big array of disciplines to interact with each other. Dependencies exist amongst almost all of the disciplines, while some are stronger (e.g. AOCS ⟷ DHS) and some less strong (e.g. DHS ⟷ Mechanical). These interactions exist throughout the engineering lifecycle.

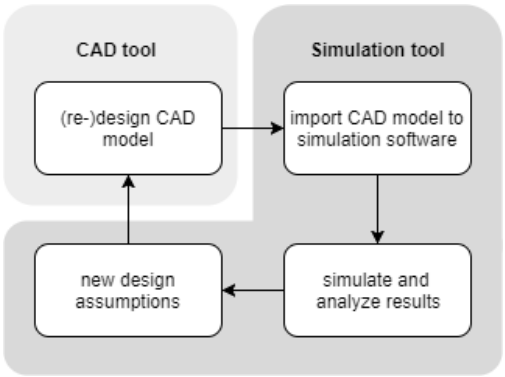


Fig. 2 Interdisciplinary interaction between the CAD and simulation tasks during spacecraft design.

Fig. 2 shows an example of the design cycle and data exchange between the CAD and simulation tasks that is needed to be certain that the right assumptions are made in the individual design activities.

During the design phases of a spacecraft project, the exchange of data between the different identified tasks and their corresponding tools is key to ensure consistency in the final design. Fig. 3 shows the interactions between the identified tasks which is needed to ensure a consistent design.

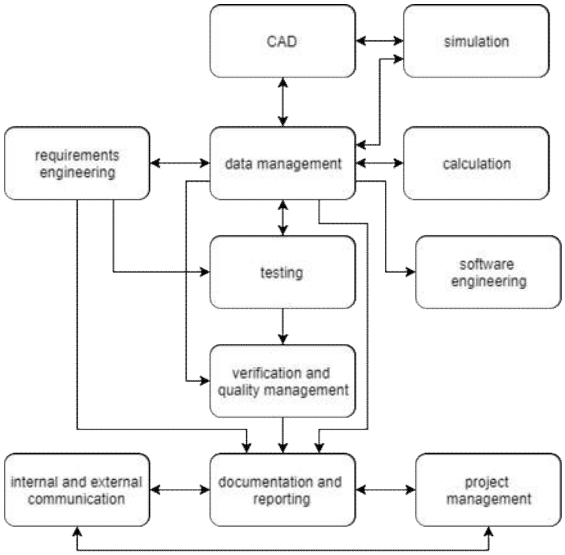


Fig. 3 Typical interaction between the identified engineering tasks

## 4. Analysis

*4.1. Status Quo*

This chapter explores the main existing approaches which aim to support the design of spacecraft throughout the engineering lifecycle.

*4.1.1. Classical MBSE – Theory and Industrial Application*

Model Based Systems Engineering (MBSE), defined as "An approach to engineering that uses models as an integral part of the technical baseline that includes the requirements, analysis, design, implementation, and verification of a capability, system, and/or product throughout the acquisition life cycle" [5] has been discussed in depth in academic circles since the 1990s and has since then given hope to the systems engineering community to solve the problems of collaborative engineering along the lifecycle and between disciplines [6]. In this approach, models describe the complete spacecraft with all of its important technical properties, allowing all engineers to

work on a common understanding of the product under development. However, after more than 15 years of availability of the method and major participations in working groups and conferences (such as e.g. INCOSE's MBSE initiative, the IEEE Technical Committee MBSE or the SECESA conference), studies suggest that MBSE has not yet reached industrial applicability. In 2009 Eisenmann concludes "A fully operational MBSE process with a corresponding tool set has not yet been realized in space projects today." [7]. And in 2015 he looks back on many standardization efforts of different institutions and their progress, but also acknowledges that "MBSE needs significant evolution for interoperability significantly for a 'plug and play' of different tools.". This is in line with findings from studies of industrial usage of model based methods and tools, which find as the main reasons for low adoption "lack of perceived value of MBSE" [8], the "inability to sufficiently merge and integrate multiple engineering applications involved in the design, production, and inspection of products across the production network" [9], as well as the conclusion that "MBSE tools are not flexible enough" [9]. The fact that ESA has been requesting the industry explicitly to apply MBSE to projects in their tenders [10], supports the claim, that MBSE in its academic form has not (yet) become an industrial reality.

However in some areas, MBSE methods have been successfully applied in practical engineering efforts. Applying the concurrent design philosophy in early stages of space engineering projects, institutions such as ESA, DLR or NASA JPL, as well as some companies have implemented Concurrent Design Facilities (CDFs) for their early design studies. In this environment, in which all disciplines are gathered in person during short periods of time for early phase studies, the interdisciplinary challenges explained in section 3.2.2 have been overcome. Integrated models do allow for cross-disciplinary engineering work. However, since the CDF and its tools have been designed for early project stages, they do not succeed in the time domain (see 3.2.1) of the engineering process. After the initial designs have been concluded, the models are mostly abandoned by systems engineers during the following years of detailed design work and manufacturing. For these phases, the MBSE models and tools do not provide sufficient value and/or capabilities, to support the daily work of engineers, working in very big teams across locations and companies.

#### *4.1.2. Industrial Reality – Solving Interactions, Using the Greatest Common Divisor*

As summarized by Haskins, there is a "need for more 'success stories' and published verifiable evidence that MBSE works to overcome the uncertainty that surrounds the capabilities and achievements of MBSE and the concerns that MBSE is just the latest 4-letter acronym fad" [11]. In view of this, the industry has mostly defaulted to proven, known processes and tools.

In "classical" engineering companies, a variety of tools exist for each special purpose: requirements management, simulations, CAD, etc. which might include models and data of different kinds. However, this data is usually not connected among each other, but instead the results of analysis are extracted (i.e. "copy-pasted") and made available to other parties in the form of documents or email exchanges.

In addition, the main deliverables from industry towards space agencies are still documents in Word, Excel or PDF format (see e.g. the chapters on deliverables according to the ECSS standard [12]). This does not only result in communication between industry and agencies to be document based at official reviews,it also determines the default communication approach for these companies, in many cases also for projects that do not have the space agencies as customers.

### *4.2. Analysis of Alternative, Emerging Solutions*

This chapter focuses on emerging software tools.

#### *4.2.1. Web-Based Tools with Integrated, Hidden MBSE*

Web-based software tools have emerged in the past to support all kinds of business processes in companies. Most of the existing tools focus in a generic way on improving Project Management (PM) tasks. While they often times make use of established PM methods (such as Kanban boards, Gantt charts, etc.), they leverage two major advantages of being web-based:

1. ***Concurrency***
   Contrary to file or document based systems all information is available in real-time to all of its users. Baselining and versioning are only relevant to reconstruct the history of data, but no longer to maintain consistency: There is a "single source of truth", which is accessible to everyone in the team and can be viewed and edited in real-time.
2. ***Data-driven***
   Although these tools use proprietary, non-standardized and often times not publicly documented models to store and connect data, information is stored as connected data. This implies a major difference to information which is stored in documents or local Excel files: the possibility to access the data inside, search, filter, sort, update or re-use it.

In the past few years more and more specialized web-based tools have entered the market, providing the same advantages for different use cases. For example, *OnShape* provides concurrent online CAD capabilities. Many users can simultaneously work on the same CAD model, observing all changes in real time, without the

need to exchange any files. Additionally, web-based software provides the advantage that computing power is provided by powerful servers in the cloud, instead of the local device. In the case of online CAD software this allows the users to view and manipulate the models also on low-performance devices, such as tablets or phones.

*4.2.2. Integrations Between Web Tools*

The classical MBSE approach follows a top-down principle; it aims at defining one single model, which allows to cover all possible engineering tasks and the expectation exists that once it has been fully defined, software vendors will implement tools which will cover all use-cases.

Contrary to that, many web-based tools follow a bottom-up approach; here tools are designed first, to fill the need of specific use cases. For interoperability, software vendors usually provide several tool integrations themselves, as well as a REST API (a web-standard for programmable data access). The internal models of the different software tools do not need to be compatible or even publicly known. They can be treated instead as a black box, which provides clearly documented in- and outputs.

This approach has up- and downsides, since on the one hand software vendors are able to come up with the models best suited for their specific application, but on the other hand, tool interoperability remains a manual task and the tools themselves are not easily interchangeable.

Three main solutions for web-tool interoperability have been pursued by software vendors:

Tool integrations provided by the software vendors themselves. These can be uni- or bidirectional and the selection and amount of integrations usually depends on the popularity of the software tool.

Tool integrations provided by the community or other software vendors. The chat application Slack is a great example of this approach. By providing a programmable API and an open marketplace for other companies, more than 600 integrations have been provided to its software.

3rd party tool integrations. Services such as IFTTT ("if-this-than-that") allow for smart integrations of services which are not yet connected. By defining triggers and actions, users can setup their own rules of tool interaction (e.g. whenever a file is placed in a specific Dropbox folder, notify the team on Slack).

The existence of web-based tools which provide these integration methods and are constantly improving them already enables engineers to collaboratively design complex products, which makes these tools a viable option for most engineering tasks.

Slowly, typical desktop applications are also migrating to web-based versions. Faced with the new competition of collaborative online tools, major vendors such as MathWorks MATLAB, Autodesk, IBM DOORS and many others have created web-based versions of their tools to try to compete with their modern counterparts. Today though, natively designed web applications seem to offer better collaboration capabilities, more integrations and standard web APIs. It therefore remains open how much market share will eventually be captured by the uprising software vendors.

*4.3. A Proposed, Integrated Hardware Development Tool Stack for New Space Engineering*

Given the benefits outlined in the previous chapter, a completely web-based tool stack for complex system development is proposed.

*4.3.1. Selected Tools*

The selected tools form an example tool stack to demonstrate that most major engineering areas can be covered by web-based tools already today. However, a full and detailed analysis and comparison of alternative tool stacks will be left to further study. The presented tool stack is not necessarily the "optimal" one, but serves as a proof of concept.

- **Requirements engineering**
  *Jama* [T1] allows browser-based traceability of requirements as well as live collaboration, including comments. When user-defined fields of requirements are changed, impacted requirements are marked for review, which ensures a consistent requirements baseline.
- **Computer Aided Design (CAD)**
  *OnShape* [T2] enables users to collaborate in real-time on mechanical CAD models. Changes are immediately visible in all browsers and the same part can be edited simultaneously by several engineers. For alternative design solutions, it supports branching, to be able to explore multiple options. Its exported files can immediately be sent to 3D printing services for prototyping.
  *AUTODESK CIRCUITS* [T3] allows for online design of (simple) electronic circuits. Their behaviour can be simulated in the tool, including interactions with code which is executed on microcontrollers, such as Arduino. In a next step PCB layouts can be designed and then ordered.
- **Calculation**
  *Valispace* [T4] connects non-CAD engineering data with formulas which are updated immediately. It replaces hundreds of inconsistent Excel files by a single model in which changes ripple through the complete design. Engineering budgets (e.g. power, mass, etc.) are auto-generated, even for complex, mode-dependent cases. It also includes interfaces to classical tools as Excel and Matlab, allowing to move gradually from a traditional engineering environment to one based on web tools.

- **Simulation**
  *SimScale* [T5] provides CFD, FEA and thermal analysis capabilities in the browser. Simulations are run in the cloud and its capacity is paid by computing core hours. This allows for complex simulations in short amounts of time when they are needed, without the need of purchasing permanent local computing power.
- **Software engineering**
  *codeanywhere* [T6] is a browser-based code-editor which allows e.g. for pair-programming in the browser similar to Google Docs, but for software code.
  *GitHub* [T7] is a software code repository in which the entire codebase of a big project can be tracked and code collaboration works with the pull and push mechanism, as well as branching. Also bugs and feature requests can be managed here.
- **Testing**
  *Jama* allows the development of test-cases which can be linked to requirements. Test steps can be marked as executed and as-run values can be stored.
- **Verification and quality management**
  *Jama* allows to trace all requirements to test cases and therefore to ensure the completeness of requirements coverage.
- **Data management**
  *Valispace* provides a central repository of all engineering data of a product. It contains all engineering margins, compares the current design to existing requirements, keeps a history of every value and handles conversion between engineering units automatically. It provides the "single source of truth" for data which has been calculated, measured or provided by suppliers across all disciplines.
- **Documentation and reporting**
  *Valispace* lets the user create analysis "documents" which contain dynamic data from its data repository. These "documents" are updated automatically whenever a value changes. This way reports, user manuals, design descriptions, etc. are always consistent and up-to-date. Baselines let the user compare older versions with the current one and when necessary, these analyses can be exported as PDF documents.
- **Internal and external communication**
  *Slack* [T8] is a company chat tool, which allows conversations in multiple "channels". It reduces the amounts of emails sent, documents discussions for future reference, and provides a transparent platform for asynchronous communication.
  *appear.in* [T9] is a browser-based video conference tool which connects entire teams via webcam and microphones. Since it is browser-based, it allows for simple to setup discussions and screen sharings with suppliers and customers to any meeting through a web link.
- **Project management**
  *Trello* [T10] is an implementation of a Kanban board which allows teams to organize tasks. Tasks are assigned to people and moved across the board while they are progressing. It is in practice a shared to-do list, where team members pick tasks which need to be done and the whole team can observe the progress and who is working on what.
  *TeamGantt* [T11] allows for online scheduling by being able to collaborate on one single Gantt chart for the project. Task dependencies and human resource planning are managed inside the tool, so that e.g. critical path analysis or team capacity can be performed for even for complex projects.

*4.3.2. Existing Interoperability Within the Tool Stack*

The advantage of using the purely web-based tools, as in the proposed tool stack, is that tool interoperability becomes straightforward with standard APIs. As an example, the simulation software SimScale allows the user to directly import CAD models from OnShape with the click of a button. After a simulation the results can be analyzed and further design decisions made. Another example is the project management app Trello's integration with the company chat Slack, where new Trello tasks can be added directly from Slack without needing to switch apps. While the tool integrations are not complete, these examples show the potential of connecting the whole tool stack to an integrated chain where inconsistencies are minimized throughout the design phases of a spacecraft mission.

*4.3.3. Benefits of this Tool Stack*

In addition to the points already identified in 4.2.1, the proposed, web-based tool stack provides the following benefits:

- increased collaborative efficiency
- short learning curve
- extendable tools which can be modified according to needs
- give selective visibility to customers when desired directly in the tool, without the need to prepare extra documents for them

*4.3.4. Risks of a Web-Based Engineering Tool Stack*

There are certain risks of switching to a web-based tool chain compared continuing with a more traditional approach. Security issues can be raised by storing sensitive data in the cloud. Most of the mentioned tools employ high security measures to protect and backup data, and additionally some companies offer local installations on client servers (e.g. Valispace).

Another drawback of working in the cloud is that engineers have to be connected to the internet to work. However, some software tools (e.g. Autodesk Fusion 360) allow for offline work and synchronizes the data as soon as the user connects to the internet again.

## 5. Results and Future Work

As a result of the analysis of this paper, it is concluded that MBSE has not been widely adopted (yet) by the industry as a solution to the system engineering needs of complex hardware projects such as spacecraft. The top down MBSE approach, where it is assumed that interdependencies are solved once a complete spacecraft model is defined, is not only impractical in many cases, but also incomplete. Emerging web-based tools on the other hand provide a bottom-up approach, where the tools fulfill the needs of specific use cases. The tools can then be connected through web interfaces (e.g. REST) and the whole spacecraft design can evolve throughout the design phases without the need to start with a complete model. Increased interoperability between the web-based tools creates a powerful tool stack to track the design of a spacecraft throughout the design lifecycle.

The proposed tool stack presents a proof-of-concept which can replace the typical engineering tools with a more streamlined and interconnected design process. As tool integrations become more advanced, engineers can take advantage of the connected data to perform deeper analyses and optimizations on the design variables. It is shown that the new, integrated approach to systems engineering can provide cost effective development of increasingly advanced space exploration missions.

Future work includes setting up example cases of individual parts of a web-based tool stack as well as interconnected systems. An important topic is to improve the integrations between the different tools. It is critical to identify the use case gaps where no good browser-based solution exists at the moment. Also interesting is a cost / benefit analysis of the different tools and a comparison of their efficiency gains.

## References

[1] C. Riley, The 400,000 strong backup team, 02. July 2009, https://www.theguardian.com/science/2009/jul/02/apollo-11-back-up-team (accessed 01.09.2017)
[2] P. Logan, D. Harvey, D. Spencer, “Documents are an Essential Part of Model Based Systems Engineering”, INCOSE International Symposium, Italy, July 2012.
[3] LLOYD, Robin. Metric mishap caused loss of NASA orbiter. CNN Interactive, 1999.
[4] ECSS-E-ST-10C Rev. 1, 15.02.2017
[5] BERGENTHAL, Jeff. Final Report Model Based Engineering (MBE) Subcommittee. February 2011 https://ndiastorage.blob.core.usgovcloudapi.net/ndia/2011/system/12953_BergenthalWednesday.pdf, 2011, (accessed 03.09.2017)
[6] EISENMANN, Harald. MBSE Has a Good Start; Requires More Work for Sufficient Support of Systems Engineering Activities through Models. INSIGHT, 2015, 18. Ed., Nr. 2, pp. 14-18.
[7] EISENMANN, Harald; MIRO, Juan; KONING, Hans Peter. MBSE for European Space-Systems Development. INSIGHT, 2009, 12. Ed., Nr. 4, p. 47-53.
[8] MOTAMEDIAN, Bita. MBSE applicability analysis. International Journal of Scientific and Engineering Research, 2013, 4. Ed., Nr. 2, p. 7.
[9] LINDBLAD, Louise, et. al., Systems Engineering from a web browser: turning MBSE into industrial reality, SECESA, Madrid, 2016, 05.-07. October
[10] “JESSICA”, 28.08.2017 http://blogs.esa.int/cleanspace/2017/08/28/applying-mbse-to-a-space-mission/, (accessed 03.09.2017)
[11] HASKINS, Cecilia. 4.6. 1 A historical perspective of MBSE with a view to the future. In: Incose international symposium. 2011. pp. 493-509.
[12] ECSS-M-ST-40C Rev. 1, 06.03.2009

## Software Tools

[T1] https://www.jamasoftware.com/
[T2] https://www.onshape.com/
[T3] https://circuits.io/
[T4] http://www.valispace.com/
[T5] https://www.simscale.com/
[T6] https://codeanywhere.com/
[T7] https://github.com/
[T8] https://slack.com/
[T9] https://appear.in/
[T10] https://trello.com/
[T11] https://www.teamgantt.com/