Trade-Off between Concurrent Engineering Software Tools for utilisation in Space Education and beyond

Christian Bach[1], Christian Drobny[2], Martin Tajmar[3]

Abstract

Concurrent engineering is an approach to the development of complex systems that is characterised by direct communication between the disciplines involved. Instead of processing the individual disciplines one after the other, as in sequential design, or processing via a single contact person, as in centralised design, all systems work simultaneously. Learning this interaction and understanding what information needs to be communicated between disciplines are among the central learning objectives of the course "Spacecraft Design" at Technische Universität Dresden, Institute of Aerospace Engineering. In this course, the students represent different disciplines and work out a mission study that is commissioned by the lecturers. The lecturers thus participate in the development process in the role of customers.

Key to the concurrent engineering approach is that each discipline has access to the most current design data at all times. This can be done via a dedicated software solution. Both commercial and open source software tools are available. Within the frame of the above-mentioned course, several tools have been tested. The covered software solutions comprise ESA Open Concurrent Design Tool (OCDT), RHEA Concurrent Design Platform (CDP), Valispace and IBM Rhapsody.

This contribution presents the experience that we gathered with these concurrent engineering software tools. First, the tools are described and their commonalities and distinctions are highlighted. Subsequently, a detailed trade-off between the tools is being presented. This trade-off will particularly focus on the utilisation of these tools within the scope of course work at universities, as this entails special requirements and boundary conditions, such as very limited time for introducing the software, highly heterogeneous user group, limited utilisation of the software in terms of depth and functionality, to only name a few. Within this contribution, we will also explore alternative approaches, such as using no software at all.

The aim of this contribution is to offer other teachers and students some guideline for selecting a concurrent engineering software solution and implementing it in course work, in a way that using the tool itself does not become the central learning challenge of the course. The results might be of interest beyond university courses, as some requirements, like short times to get familiar with the software or certain interface requirements, also apply to other environments in research and development.

Keywords

Concurrent engineering, concurrent design, software tools, education

[1] Corresponding author: Technische Universität Dresden, Germany, Christian.Bach1@tu-dresden.de
[2] Technische Universität Dresden, Germany, Christian.Drobny@tu-dresden.de
[3] Technische Universität Dresden, Germany, Martin.Tajmar@tu-dresden.de

1. Introduction

Concurrent engineering (CE) is an approach to the development of space systems and mission. It is characterised by direct communication between subsystems and parallel working of the involved disciplines. Learning this interaction and understanding how the different subsystems are connected to each other (i.e. which interfaces there are and which in- and ouputs have to be transmitted) might be just as important for students as learning about the individual disciplines (e.g. propulsion, thermal, communication). At Technische Universität Dresden (TUD), there is a dedicated course to introduce them to the CE philosophy [1].

At the beginning of the course, the characteristics as well as advantages and disadvantages of design processes are taught. Special focus is put on concurrent engineering. In addition, an introduction to the utilised CE software is given. The remaining time is used to carry out a concurrent engineering process for the conceptual design of a space system (e.g. a Mars probe or a Moon rover). For this purpose, a mission objective is issued by the teachers and the role of the customer/client is assumed. The mission is first discussed by the students and initial solution concepts are postulated, which are then evaluated. We / the students divide themselves into different roles/disciplines. Each discipline develops the corresponding subsystem (e.g. for energy supply or communication) or carries out the tasks belonging to the corresponding role (e.g. cost or risk analysis).

The CE process implementation is usually done with a dedicated infrastructure, which involves hard- and software. Latter is nowadays represented by a multitude of tools, including commercial and open source solutions. This contribution presents our experience with a selection of the available software tools. The aim of this contribution is to offer other teachers and students some guideline for selecting a concurrent engineering software solution and implementing it in course work, in a way that using the tool itself does not become the central learning challenge of the course.

Therefore, the tools will be described in section 3. The actual trade-off will be executed in section 4, before concluding the paper in section 5.

2. Concurrent Engineering Tools

Numerous tools to aid the concurrent design process are available. The tools tested here were chosen due to previous experience with them from workshops, projects or similar usage. This list is not meant to be a complete overview of all software tools that could be utilised, but represents the tools that we actually investigated both theoretically (*Rhapsody* and *OCDT*) and practically (*Rhea CDP* and *Valispace*). Note that further tools are being used in concurrent design facilities (CDF), such as the *Virtual Satellite* [2] tool used at the German space agency (DLR) or the tool *Poseidon* developed by NASA [3].

2.1. Valispace

Valispace [4] uses a browser-based web-interface to access a central database (so-called single source of truth) in which the actual design is been stored and advanced. Depending on the chosen license, this can be either a cloud-based database or a distribution on a local server. The database can be accessed by any user at any time from any browser system, which guarantees wide compatibility and low software requirements. However, this can also be a challenge due to the wide range of available browser types and active browser versions.

The design itself is based in a so-called product tree, which is a hierarchic representation of components and subcomponents with its representing parameters (so called Valis) that define the component. Valis can be dependent of each other, allowing automated calculations as well as budgets over different layers of the component structure. This allows, for instance, quick and easy parametric studies when varying single Valis.

Many quality-of-life-features are included, like alternative containers ad system modes. A complete unit calculation is implemented, including non SI-units. Furthermore, a history graph allows to follow the evolution of any Vali value over time. Datasets can be implemented as lookup table or for. Lastly, a network of interactions between Valis can be plotted, to name a few.

Valispace has implemented many more features that revolve around the product tree and allow for a more convenient design procedure. Although featuring all these capabilities, Valispace strife’s to be slick in its interface and intuitive to understand and use. Short introductions to the tool proved to be sufficient for students to get a grip of its functionality and start designing. The tutorial, that is available at the website [5], allows to get started in a rather short time. This allows for easy and convenient access for any user, which may be in particular beneficial for not as experienced user like beginners (i.e. students) or customers.

2.2. Rhea CDP

The Concurrent Design Platform (CDP) by Rhea [6] is a detailed design tool with high focus on implementation of space standards like the ECSS-E-TM-10-25A [7]. Here, we want to share our experience with mainly the CDP3.12 as well as the CDP4 versions. However, we need to consider the fact that the tool has since been developed further and is now available under the product name “Comet”.

One unique aspect of CDP is the design procedure, which avoids real time changes in favor of a discrete approach of forwarding changes. If any user adds or changes existing parameters, these changes are stored in a dedicated routine. Although every user may see indications that changes have been done, these are not activated right away. A user with a higher level of authority, for instance the team leader of the study, has to manually publish these changes so that it may be live in the actual design. Although this may seem like a highly inconvenient feature at first, it significantly reduces the continuous noise of changes occurring in the earliest design phases. This lowers the risk of potential performance issues of the tool, since it does not require permanent updating. Also, a very high number of additions and changes may only be expected during the initial phase, fast publishing can avoid any problems. In later stages of a design, changes mainly update initial values, in which the exact value may not be critical for other components, as long as they are connected correctly. In any case, this design procedure requires additional tasks and communication, which can negatively affect the development process particularly in a setting with students that are first-time users of the software.

The design itself is stored in a product tree that consists of components and subcomponents with dedicated parameters. Latter are defined in large detail. Furthermore, a strict ownership is established that defines who will be able to adjust a certain parameter, depending on who created it, respectively how it was defined initially. These aspects can make it very difficult for a new user to quickly get into creating objects and design content. However, once getting used to this technique and understanding the important aspects, it is easy to very clearly define all the aspect of any parameter.

2.3. IBM Rhapsody

From the tools discussed in this paper, IBM Rhapsody [8], may be the one with the fewest correlations to space mission design, as it is developed as a general model-based system engineering (MBSE) tool for any application. Still, it provides crucial features to enable the concurrent design approach. In our evaluation, SysML is used as modelling language.

The general idea of Rhapsody is to have different types of views onto one central model, where each view is optimized for different aspects of specification of the model. The central model itself can again be represented in a product tree, allowing an easy hierarchic structure of the major components. The different views, also called diagrams, focus, for example, on the structure of the subsystems, the definition and connection of requirements, the interaction with users, the definition of states of the system, the definition of actions and data exchanged in the system and so on. Consequently, an initially simple hierarchic structure of a model gets multiple layers of complexity, but the different diagrams keep it comprehensible.

Since the focus of Rhapsody is not on the guidance of calculations and therefore the implementation of parametric studies, but rather on the best possible modelised representation of the design, the user has the possibility/task to define any data up to the highest level of detail. For anyone new to the program and its implementation, this may very well be overwhelming, which can be, to the authors experience, a significant hurdle for anyone starting to model in order to exchange data. On the other hand, since much of the set up of data may be multiple layers bellow the initial level of the diagrams, this can make it much easier for any spectator to get the general grasp of the structure and functionality of the model in a top layer view.

2.4. ESA OCDT

Used in the CDF of ESA, the Open Concurrent Design Tool (OCDT) is a client/server software package that was developed for ESA. It shares most commonalities with Rhea's CDP. As CDP, it implements a standard semantic data model based on ECSS-E-TM-10-25. The database, which is stored on a server, is accessed via an OCDT client, which is based on the Microsoft software Excel. Therefore, analyses and calculations can be done directly in Excel, that utilises various spreadsheets that can be added to the workbook as needed. Thus, the work is done locally and the data is then shared via the OCDT interface. [9]

This exchange of information is not done automatically and therefore not instantaneously. Parameters need to be "pushed" to the database by their creator, who is responsible to keep it up to date. Users who wish to use this parameter need to subscribe to it, which defines the interrelations inside the model. Afterwards, they still need to pull the parameter to their local Excel interface. Moreover, like in Rhea's CDP, the team leader or system engineer needs to publish data sets after checking the values for consistency. [9]

Apart from that, users are free to create elements/components and attach parameters to them. Those parameters can have advanced characteristics, such as state or option dependencies. Former are used to model system modes or mission phases. Latter are used to model different system options, e.g. to compare an electrical with a chemical propulsion solution and the system effects thereof. [9]

2.5. Analogue tools

All tools presented here have great advantages for particular areas supporting the concurrent design process. However, the tool needs to be intuitive and easy to learn in order to be used by the students in the academic scenario presented. If the software is to complex, students will fall back to familiar alternatives. We observed that students will avoid the software interface and rather just note and share disconnected information on a common board in the room or facility they are in.

For a course in presence, this may be an option since everyone is working at the same time and means of exchange and communication can be very short. And indeed, we normally started of our courses with a discussion about the general concept idea together on a whiteboard. And even at later stages of the study, this became a pivotal point for the evolution of the design. For general and basic design, this may even be the best option, since students don't have to learn how to write information on a board, and can focus solely on the design of the respective subsystem responsible.

However, since the design will get complex by itself in no time, the design would quickly get unorganized. In addition to that, for any non-centralized design study over a longer period of time, as it was required in the resent years, this cannot be an option, and the dedicated opportunities for exchanging data needed to be required to be used.

While the analogue option surely has rather narrow limitations, it remains a viable option to learn the CE Methodology. And for any system with low complexity it may still be the way to go.

3. Trade-off

The following trade-off will particularly focus on the utilisation of described tools within the scope of course work at universities, as this entails special requirements and boundary conditions, which might not apply to other environments, such as the industrial utilisation of CE. Within this trade-off, we summarise our experience with and assessment of the tools. We didn't conduct this trade-off a priori and then implemented the most promising solution into our course, but we actually tested different options to see what works for us and what not.

3.1. Evaluation Criteria

This section contains the selection of the evaluation criteria for the trade-off with a short description of each criterion to clarify what it represents and how it is assessed. The following criteria will be used:

Usability: A key factor for using a CE software in a course is the time the students need to make use of it, as there is only limited time available. Therefore, the software should be easy to understand in its basics, but not necessarily in its full potential. This including the availabiltiy of freely accessible manuals and tutorials.

Complexity: While enabling very complex models is surely a key aspect for most CE users, it is of secondary concern for the use in an educational framework. However, it is still important to consider. A less complex software could prove beneficial for the course work. However, it would be even better if the software provides complexity, allowing interested students to dig deeper, but not unleashing the full complexity all at once at the new user.

Interface: Aside from the usability and complexity, the design of the user interface also plays an important role, as it defines how the user interacts with the software. While some tools rely on the use of Excel as an interface, other software use browser-based interfaces. While it is clear that the borders to usability and complexity are fluent, this criterion shall put focus on how easily, or better naturally, the user can engage with the software.

Performance: Another criterion is the software's performance. Not only too much complexity or a bad user interface can turn the student away from the screen, also performance issues can. We experienced that as soon there are problems with the stability of the software or serious latency in the data synchronisation, the acceptance drops. Thus, the software and it's implementation in the hardware must ensure not the highest, but flawless performance for a representative user group.

Manageability: This criterion represents the administrational effort for the lecturers, which themselves have limited time and want to put as much focus as possible on the students and their learning processes. Still, they have to set up the software and take care of any troubleshooting along the way. Therefore, this criterion highlights the knowledge that is needed and how much effort it takes to get and keep the software running.

These five criteria (usability, complexity, interface, performance and manageability) will be used for our trade-off. However, the analysis could be extended by further criteria. This could involve the supported interfaces for the implementation of further software solutions (such as design and simulation software). Another aspect might be the requirements of the software towards the hardware infrastructure. Lastly, some might consider available licenses and corresponding prices important.

The five criteria presented are all significant in their very own aspect, which concludes that the failure to fulfill any one of these may have severe influence on the usage by the students participating at the course. Therefore, it was decided to not add any additional weighting factors between these evaluation criteria.

3.2. Evaluation

Due to the limited time available during the course, easy accessibility of the functionality of the tool is of significant importance. Since most students are fairly firm with basic Excel operations, it does not take long to get acquainted with the OCDT tool. It is easy to start and available on most PCs. The availability of a browser for Valispace is even more so given to any user, making it highly accessible. However, some time to understand the setup of the tool is required to get the principal idea. Still, the tool is kept rather simple and intuitive, and catching the tutorials available will only take a few hours and has proven to be well suited to get started. For CDP and Rhapsody, additional software has to be installed. Some knowledge about server setup may be required, but access itself is easy. Once this one is covered, it can be challenging for beginners to get used to the tools, due to its very detailed options available. With both tools, significant time has to be invested to understand how information is created, connected, and to be stored in the model. From our experience, the level of expertise and therefore the level of usage will differ much stronger for CDP and Rhapsody than for Valispace and OCDT, simply due to the different background and interests of the students. This higher difference makes it more challenging for the tool to be actually used by the students during the course.

The OCDT, Valispace as well as CDP are particularly designed to aid the design of space related missions. Although other studies may also be conducted, numerous features support this general field of study, including the handling of units. For new users, this can be quite an important feature to guide the addition of information. Furthermore, a well-known or intuitive interface will also be beneficial for starters. Guiding the user step by step to add more information is best implemented in Valispace, where only basic information needs to be defined initially, but more detailed parameters can be added at a later intuitively. Although updating of parameters is also feasible with CDP and Rhapsody, the user will be confronted with these parameters already at the initial definition of an object, which results into a much slower process of adding information and more hesitance by the unexperienced users. Particularly with Rhapsody, a lot of information has to be added up front, but an experienced user can present this information later visually very appealing and sorted by use of different types of diagrams.

The functionality of updating the model differs for the tools. Naturally, Excel comes to its limits once a system gets more complex and will consequently take more time to update. Similar challenges have been observed using Valispace, since the update of a multitude of parameters can be resourceful and take a moment. For the CDP, the model will only be updated by a top-level user, making the system more discrete, but also requiring less data being exchanged continuously, improving the performance significantly. For Rhapsody, the aspect for downloading a recent part of the model und uploading it again to the cloud can be a nuisance, in particular when starting from a blank slate and many changes by many different users are to be expected.

From the educator's point of view, the setup of the tools is similar for all options, since respective accounts/access rules have to be added with all of them. However, making use of widely available access points like Excel for the OCDT and a browser for Valispace makes for more flexibility in planning the courses and allowing decentralized work. In the end, installing additional software and setting up the respective server for data exchange has always to be respected as a certain time factor, if no dedicated design felicity can be used.

4. Conclusion

Multiple tools have been used by the authors to conduct concurrent design studies in a university level course with students. Still, this is definitely not a full list of tools, as there are more available. In addition to that, the user experience by the authors is obviously also limited, and experienced users may be able to cover many more tasks with the dedicated tools. After all, the authors want to encourage any reader to at least give these tools a try, since they all are very capable and powerful in their very own way. Also, the tools are under constant development, which means that certain aspects may have changed since the writing of this paper.

For the course at hand, the software implementation by Valispace is our preferred solution so far. The tool grants easy access and requires only a minimum of initial training, which also can be self-taught, to enable students to work with the tool and start designing. Since the results of our design is not the main priority and the design itself will not get as complex, we can respect possible limitations quite well. Additional tools like time management and the implemented requirement management and report tool are additional benefits for our course. From our experience, the tool provided the best introduction to the general CE approach for the students and resulted in the greatest amount of data shared within such a tool.

Acknowledgements

We would like to thank all participating students for their adaptability, commitment, understanding and feedback.

References

[1] C. Bach, C. Drobny, T. Schmiel, and M. Tajmar, "Remote Concurrent Engineering from the customer's perspective," *Lessons Learn. 1*, 2021.

[2] "Virtual Satellite Download Page." [Online]. Available: https://dasclab.eu/virsat/.

[3] B. Wickizer, T. Snyder, J. DiCorcia, R. Evans, R. Burton, and D. Mauro, "A New Concurrent Engineering Tool for the Mission Design Center at NASA Ames Research Center," in *2021 IEEE Aerospace Conference (50100)*, 2021, pp. 1–12.

[4] "Valispace Website." [Online]. Available: https://www.valispace.com/.

[5] "Valispace Tutorial." [Online]. Available: https://docs.valispace.com/vhd/Fan-Tutorials.1512243215.html.

[6] "RHEA CDP Website." [Online]. Available: https://www.rheagroup.com/services-solutions/system-engineering/concurrent-design/.

[7] E.-E. ECSS Secretariat, *[ECSS-E-TM-E-10-25A] Space Engineering - Engineering design model data exchange (CDF)*, First Issu. ESA Requirements and Standards Division, 2010.

[8] "IBM Rhapsody Website." [Online]. Available: https://www.ibm.com/products/systems-design-rhapsody.

[9] "OCDT Website." [Online]. Available: https://ocdt.esa.int/.