Lessons Learned 3, 1 (2023)
Eingereicht am: 16.10.2022
Angenommen: 08.05.2023
DOI: https://doi.org/10.25369/ll.v3i1.63
ISSN: 2749-1293 (Print); 2749-1307 (Online)

Concurrent Engineering Software Tools – A Trade-Off for efficient Learning in Blended Teaching Scenarios

C. Bach*, C. Drobny, M. Tajmar

Professur für Raumfahrtsysteme, Institut für Luft- und Raumfahrttechnik, Fakultät Maschinenwesen, TU Dresden

Abstract

Concurrent engineering is an approach to the development of complex systems that is characterised by direct communication between the disciplines involved. Key to this approach is the access to the most current design data by all participants at all times. This can be done via a dedicated software solution, for which both commercial and open-source software tools are available. How these tools influence the outcome of the class itself, has been discussed extensively in a separate publication.
This contribution presents the experience that we gathered with different concurrent engineering 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. 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.

Concurrent Engineering ist ein Ansatz zur Entwicklung komplexer Systeme, der sich durch direkte Kommunikation zwischen den beteiligten Disziplinen auszeichnet. Der Schlüssel zu diesem Ansatz ist der Zugang zu den aktuellsten Konstruktionsdaten für alle Beteiligten zu jeder Zeit. Dies kann über eine spezielle Softwarelösung erfolgen, für die sowohl kommerzielle als auch Open-Source-Softwaretools zur Verfügung stehen. Wie diese Werkzeuge das Ergebnis des Kurses selbst beeinflussen, wurde in einer separaten Veröffentlichung ausführlich erörtert.
In diesem Beitrag werden die Erfahrungen vorgestellt, die wir mit verschiedenen Softwaretools für das Concurrent Engineering gesammelt haben. Ziel dieses Beitrags ist es, anderen Lehrenden und Studierenden einen Leitfaden für die Auswahl einer Softwarelösung für das Concurrent Engineering und deren Implementierung in die Lehrveranstaltung an die Hand zu geben, und zwar so, dass die Verwendung des Tools selbst nicht zur zentralen Lernherausforderung der Lehrveranstaltung wird. Die Ergebnisse könnten auch jenseits von Universitätskursen von Interesse sein, da einige Anforderungen, wie z.B. kurze Einarbeitungszeiten in die Software oder bestimmte Schnittstellenanforderungen, auch für andere Umgebungen in Forschung und Entwicklung gelten.

*Corresponding author: christian.bach1@tu-dresden.de

Acronyms / Abbreviations

CDF	*Concurrent Design Facility*
CDP	*Concurrent Design Platform*
CE	*Concurrent Engineering*
DLR	*German Space Agency*
ECSS	*European Corporation for Space Standardisation*
ESA	*European Space Agency*
EWM	*Engineering Workflow Manager*
IBM	*International Business Machines Corporation*
MBSE	*Model Based Systems Engineering*
OCDT	*Open Concurrent Design Tool*
TUD	*Technische Universität Dresden*

1. Introduction

Concurrent engineering (CE) is an approach to the development of space systems and mission. It is characterised by the 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 outputs have to be transmitted) might be just as important for students as learning about the individual specialised disciplines (e.g. propulsion, thermal, communication). At Technische Universität Dresden (TUD), students can learn this process by participation in student projects like the development of CubeSat missions or the development of experimental sounding rockets rocket. Furthermore, for engineering students in aerospace, there is also a dedicated course to introduce them to the CE philosophy [1] .

The CE process implementation is usually done with a dedicated infrastructure, which involves hard- and software. The 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.

A detailed overview of the educational aspects of the non-centralized course structure have been discussed extensively in a previous publication [1]. There, advantages and challenges of the course structure but also feedback provided by the participants of the study itself are discussed and evaluated. However, technical considerations of the used tools itself are mostly neglected.

This technical focus shall be discussed in more detail in this contribution. Therefore, following a summary of the software requirements in section 3, the tools will be described in section 4. The actual trade-off will be executed in section 5, before concluding the paper in section 6. Yet before, section 2 will present the educational framework of this analysis.

2. Educational Framework

The course "Spacecraft Design" is embedded in the specialisation module Space Systems Engineering of the diploma programme Mechanical Engineering, specialisation Aerospace Engineering, and now regularly takes place in the 8th semester. This course is one of two courses of the aforementioned module and is completed by a written report for examination. [1]

The students have already acquired detailed knowledge of the design of space systems in courses such as "Energy Systems for Spacecraft" or "Space Propulsion". The course combines the students' knowledge from all previous courses and showcases the high complexity and dependency of vastly different aspects when designing a space mission. The overall learning objectives of the course can be summarised as follows:

- By establishing criteria, weighting them and performing a trade-off, students can comparatively evaluate concepts for space missions to find the solution approach with the highest probability of success.
- By practically applying and combining the knowledge gained in the previous courses, students will be able to conceptualise space missions to develop an overall system to solve a specific engineering problem.

- By getting to know their characteristics as well as advantages and disadvantages, the students know different strategies and models for the development of technical systems and are able to classify and assess them in order to apply them in a targeted and justified manner.

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, a Moon rover, or a sounding rocket). For this purpose, a mission objective is issued by the teachers, who then assume the role of the customer/client for the rest of the course. The mission is first discussed by the students and initial solution concepts are postulated, which are then evaluated. The students inscribe themselves for 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).

Until 2020, the course was held as a block course in a computer lab on three complete days, spread over a period of eight days. Afterwards, the course was transferred into a remote virtual format, which became necessary due to the restrictions associated with the COVID-19 pandemic. The core of the restructuring was the stretching of the course over the entire semester. The portfolio of utilised methods included screencasts to teach the basics at the beginning of the course, shifting the actual elaboration to self-study, and regular live consultations with short presentations by the students. With lower restrictions in the past semester, the course was adapted to a blended teaching version of this course, combining aspects from both previous versions: The course is still stretched over the entire semester, which allows a better focus of the students to the task at hand, be it the exchange of information during meetings, or the development of the responsible subsystem in between the meetings. For the introduction of the course, the available digital course material was utilized, but the consultation meetings were held in person at the institute, allowing for a much-improved room for open discussions to push the state of the design.

3. Basic Software Requirements

Designing any system with a certain complexity is usually not a straightforward process. This is particularly the case for space missions, due to the subsystem experts that may often be very disconnected from each other, both in terms of perspective and physical space. Therefore, a design process requires thorough and rigorous documentation. Appropriate software tools shall support data exchange and guarantee data consistency for everyone. Furthermore, it shall guarantee that the correct information is shared by a standardized definition of objects in the tool, since different subsystems may have vastly different ways of expressing their particular information.

In our course, the students shall experience the concurrent design approach in a first-hand manner, to learn the advantages and strengths, but also get to know limitations and challenges. For this, they shall get to know specific tools that may support a concurrent engineering approach and understand, how this can influence the approach on designing itself. The technical results of the design task itself is only of secondary relevance.

Consequently, the aspects to evaluate possible tools may vary significantly to any industrial or research-oriented approach. For instance, good accessibility and easy implementation of the tool are important, as we may not have a dedicated facility available and in times of remote teaching, students have to have access to the tool from their own personal computer. Since the tool itself is only one part of the course and utilisation of the tool shall not become the main learning challenge, it should be quite intuitive and not require extensive teaching and learning in order to get started. The fact needs to be respected that the students are neither experienced in the design approach itself, nor experts in the subsystem they will represent during the study, making it stressful to tackle too many unknown aspects at once.

Since the technical result of the study is only of secondary importance, the level of details of any information stored may not be decisive, same with the level of complexity, as it is not expected to have too many finalized interconnections between the subsystems. Rather, the system should feature possibilities to define direct relationships between parameters that can be automatically computed, since this can highly benefit the design approach.

4. Concurrent Engineering Tools

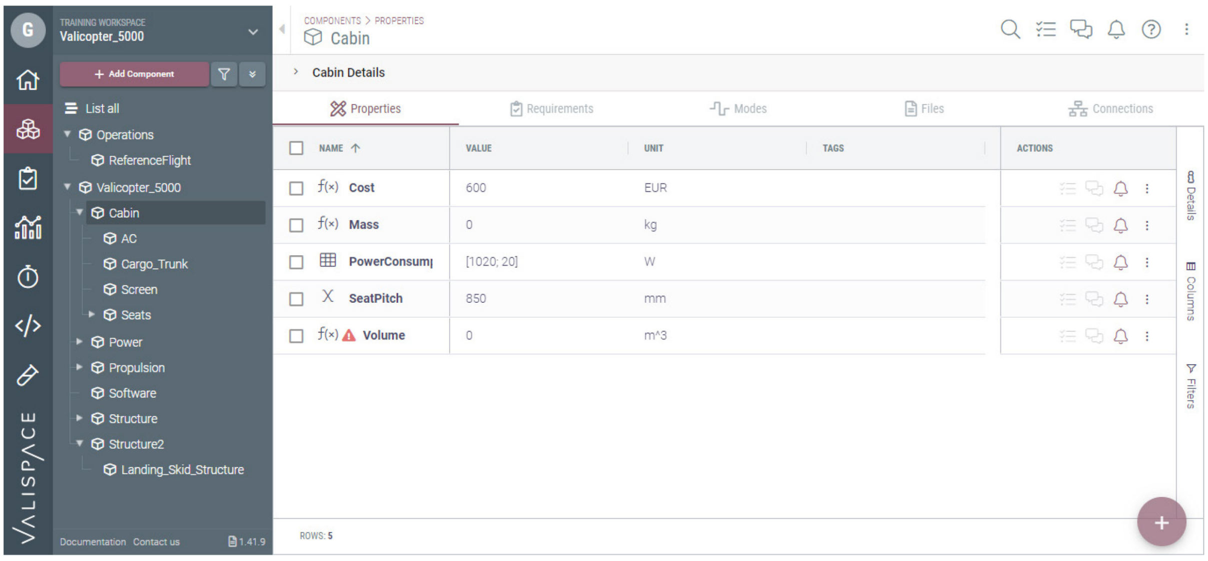
Numerous tools to aid the concurrent design process are available. The tools tested here were chosen due to previous experience with them from workshops, projects or similar usage. This list is not meant to be a complete overview of all software tools that could be utilised, but represents the tools that we actually investigated both theoretically (IBM 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].

Valispace

Valispace is a German-Portuguese start-up [4] that uses a browser-based web-interface to access a central database (so-called single source of truth) in which the actual design is 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 major goal of Valispace is the development of a design tool to allow “real time collaboration inside and across teams, even with suppliers and customers”[4]. It is designed to support any level of design ranging from early concept studies “through detailed design up to testing and documentation” [4], including a livid requirement engineering.

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 (see Figure 1).



TRAINING WORKSPACE
Valicopter_5000

COMPONENTS > PROPERTIES
Cabin

Cabin Details

- List all
- Operations
  - ReferenceFlight
- Valicopter_5000
  - Cabin
    - AC
    - Cargo_Trunk
    - Screen
    - Seats
  - Power
  - Propulsion
  - Software
  - Structure
  - Structure2
    - Landing_Skid_Structure

NAME	VALUE	UNIT	TAGS	ACTIONS
f(x) Cost	600	EUR		
f(x) Mass	0	kg		
PowerConsum|	[1020; 20]	W		
X SeatPitch	850	mm		
f(x) Volume	0	m^3		

ROWS: 5

Fig. 1: Screenshot of the Web-Interface of Valispace in the "Components" Section. On the left-hand side, the product tree with the implemented components and their hierarchic structure is shown. On the central and right-hand side, Valis (details) to the selected Component ("Cabin") are shown. [5]

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 parameters.

Furthermore, Valispace allows the implementation of alternatives of components that can be conveniently switched in between, for instance implementing different engines in a rocket. In addition to that, modes can be defined allowing the definition of the system at different states of the mission, for instance following the multistage behaviour of a rocket during ascent. Any change by any user will automatically be updated to anyone else accessing the database, allowing close-to real time changes in the model and exchange of data.

Valispace also has implemented numerous quality-of-life-features, including a complete unit implementation, that is able to handle and interchange many different unit systems, also including non-SI-units. Furthermore, a history graph for any parameter allows to follow the evolution of the parameter value over time. Datasets can be implemented to allow for defined interpolation of input values. Also, a general network of interactions between parameters can be plotted.

Many more features have been implemented in Valispace that revolve around the product tree and allow for a more convenient design procedure. All these features are able to link to a certain component or parameter in the product tree, so that it can always be up to date. For instance, the Analysis tool, in which data can be prepared in a document style, including automatically updated tables, graphs and budget lists that can be implemented in reports. There is also a simulation tool that allows for more complex calculations with multiple output parameters. Finally, there is an extensive tool to manage requirements, which can be automatically checked with multiple expressions against parameters of the product tree. Test necessities, procedures, protocols and results can be easily requested and stored accordingly.

Although featuring all these capabilities, Valispace strives to be lean in its user 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 [6], 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 the unexperienced user.

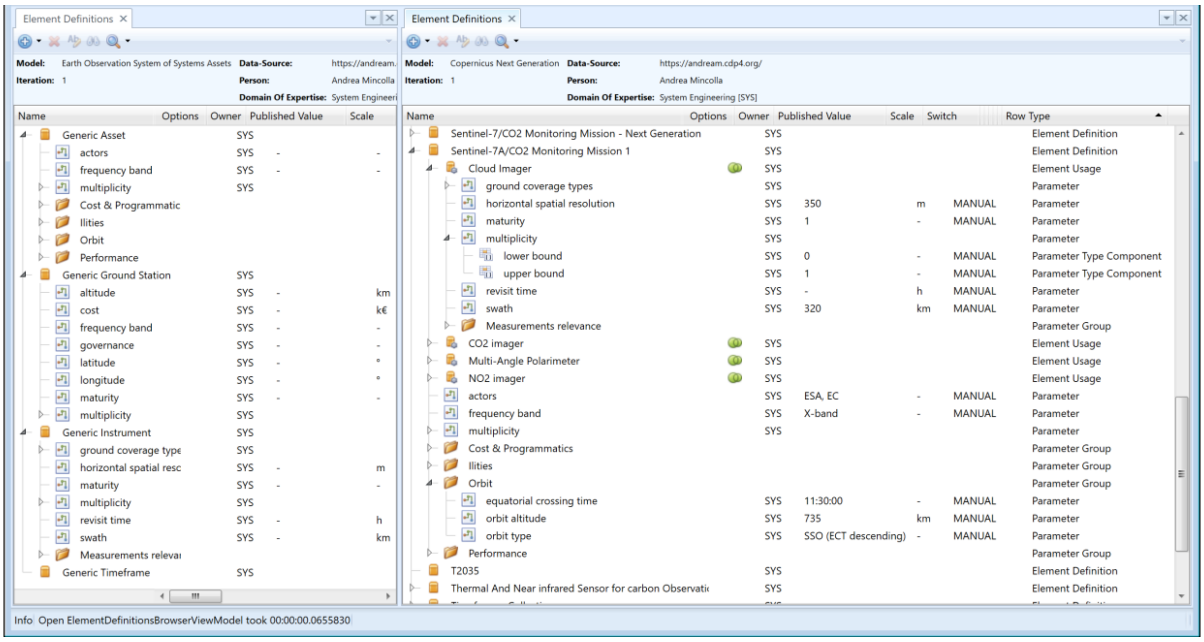
Rhea CDP

The Concurrent Design Platform (CDP) by Rhea [7] is a detailed design tool with high focus on implementation of space standards like the ECSS-E-TM-10-25A [8]. Here, we want to share our experience with mainly the CDP3.12 as well as the CDP4 version. However, please note that the tool has since been developed further and seen several releases, and is now available under the name “Comet”.

CDP is a standalone program that has to be installed first. It may require a certain Excel Version for some functionalities. In addition to that, a server routing may be necessary to open up a dedicated central database for the design itself. Some knowledge about server setup may be required. However, one can easily connect to any project one has access to, once everything is set up accordingly.

One unique aspect of the tool is the design procedure, which avoids real time changes in favour of a discrete approach of forwarding changes. If a user adds a parameter or changes the value of any existing parameter, these changes are stored in a dedicated routine. Although any user may see indications that changes have been done to a certain parameter, 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, in which fast publishing may neglect any problems occurring. 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 accordingly. 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.

In addition to this discrete publishing approach, another level of setpoints can be used, being iterations. Here, a user with higher level of authority may set an iteration setpoint that basically copies the current design and freezes its status. These iteration setpoints may be used to analyse the evolution of certain parameters over the course of the study to analyse certain converging behaviour of the design.



Element Definitions

Model: Earth Observation System of Systems Assets **Data-Source:** https://andream.
Iteration: 1 **Person:** Andrea Mincolla
Domain Of Expertise: System Engineeri

Name	Options	Owner	Published Value	Scale
Generic Asset		SYS		
actors		SYS	-	-
frequency band		SYS	-	-
multiplicity		SYS		
Cost & Programmatic				
Ilities				
Orbit				
Performance				
Generic Ground Station		SYS		
altitude		SYS	-	km
cost		SYS	-	k€
frequency band		SYS	-	-
governance		SYS	-	-
latitude		SYS	-	°
longitude		SYS	-	°
maturity		SYS	-	-
multiplicity		SYS		
Generic Instrument		SYS		
ground coverage type		SYS		
horizontal spatial resc		SYS	-	m
maturity		SYS	-	-
multiplicity		SYS		
revisit time		SYS	-	h
swath		SYS	-	km
Measurements relevai				
Generic Timeframe		SYS		

Info Open ElementDefinitionsBrowserViewModel took 00:00:00.0655830

Element Definitions

Model: Copernicus Next Generation **Data-Source:** https://andream.cdp4.org/
Iteration: 1 **Person:** Andrea Mincolla
Domain Of Expertise: System Engineering [SYS]

Name	Options	Owner	Published Value	Scale	Switch	Row Type
Sentinel-7/CO2 Monitoring Mission - Next Generation		SYS				Element Definition
Sentinel-7A/CO2 Monitoring Mission 1		SYS				Element Definition
Cloud Imager	●	SYS				Element Usage
ground coverage types		SYS				Parameter
horizontal spatial resolution		SYS	350	m	MANUAL	Parameter
maturity		SYS	1	-	MANUAL	Parameter
multiplicity		SYS				Parameter
lower bound		SYS	0	-	MANUAL	Parameter Type Component
upper bound		SYS	1	-	MANUAL	Parameter Type Component
revisit time		SYS	-	h	MANUAL	Parameter
swath		SYS	320	km	MANUAL	Parameter
Measurements relevance						Parameter Group
CO2 imager	●	SYS				Element Usage
Multi-Angle Polarimeter	●	SYS				Element Usage
NO2 imager	●	SYS				Element Usage
actors		SYS	ESA, EC	-	MANUAL	Parameter
frequency band		SYS	X-band	-	MANUAL	Parameter
multiplicity		SYS				Parameter
Cost & Programmatics						Parameter Group
Ilities						Parameter Group
Orbit						Parameter Group
equatorial crossing time		SYS	11:30:00	-	MANUAL	Parameter
orbit altitude		SYS	735	km	MANUAL	Parameter
orbit type		SYS	SSO (ECT descending)	-	MANUAL	Parameter
Performance						Parameter Group
T2035		SYS				Element Definition
Thermal And Near infrared Sensor for carbon Observatic		SYS				Element Definition

Fig. 2: Working Space inside the Rhea CDP. Shown is the element definition workspace in the tree-list decomposition view for the Model Catalogue (left) and the Study Model (right). [9]

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 generating content. However, once getting used to this technique and understanding the important aspects, it is easy to very clearly define all aspects of any parameter, which makes it easier to later bundle different parameters all over the product tree into detailed budgets and overviews. This is further supported by a model catalogue, which allows the reuse of predefined components over multiple studies.

Although the CDP can be used for any concurrent design approach, it has a defined focus on space mission design. This stems from the implementation of various space standards like the ECSS-E-TM-10-25A [8]. In this standard, best practices for software aided design of space missions are comprised, defining nomenclature as well as data storage in order to enable compatibility between different tools.

Finally, the CDP has an interface to Excel, allowing the implementation of more complex calculations into the design. Therefore, the entire product tree will be exported and linked to an Excel file, which can be updated in both directions to publish changes accordingly. Complex budget calculations, pre-defined graphics as well as calculations can be conducted and exchanged respectively.

IBM Rhapsody

From the tools discussed in this paper, IBM Rhapsody [10] 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 many interesting features to enable the concurrent design approach. In our evaluation, SysML is used as modelling language.

Rhapsody itself is a standalone programme to be installed on one's device, which references itself to a file either on the computer or on a cloud. In order to enable concurrent engineering in a team of multiple users, additional software is required, e.g. the Engineering Workflow Manager (EWM), which can be set up to allow somewhat simultaneous work at the project. However, no actual real-time changes are shared but rather parts of the model are changed by the user in a local copy and afterwards uploaded to the common stream of data for everyone to see. From the university perspective, Rhapsody might be very interesting due to its educational support for educators and students. It is part of the academia initiative by IBM, making it highly accessible for educational purposes.

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.

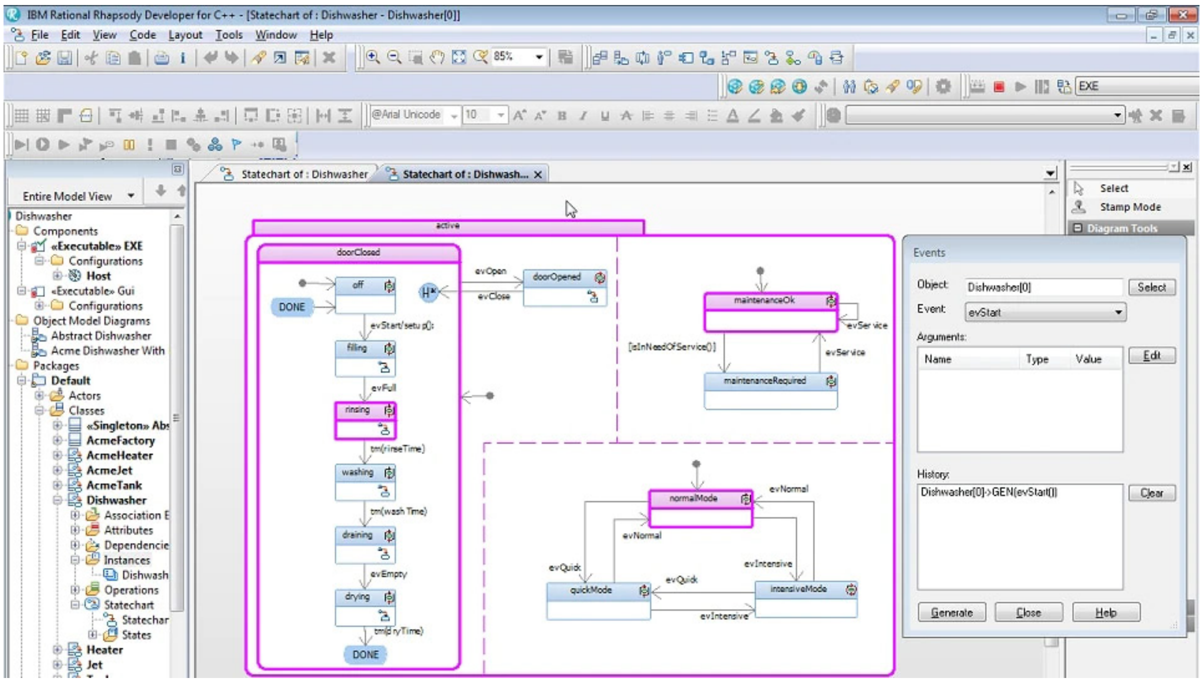


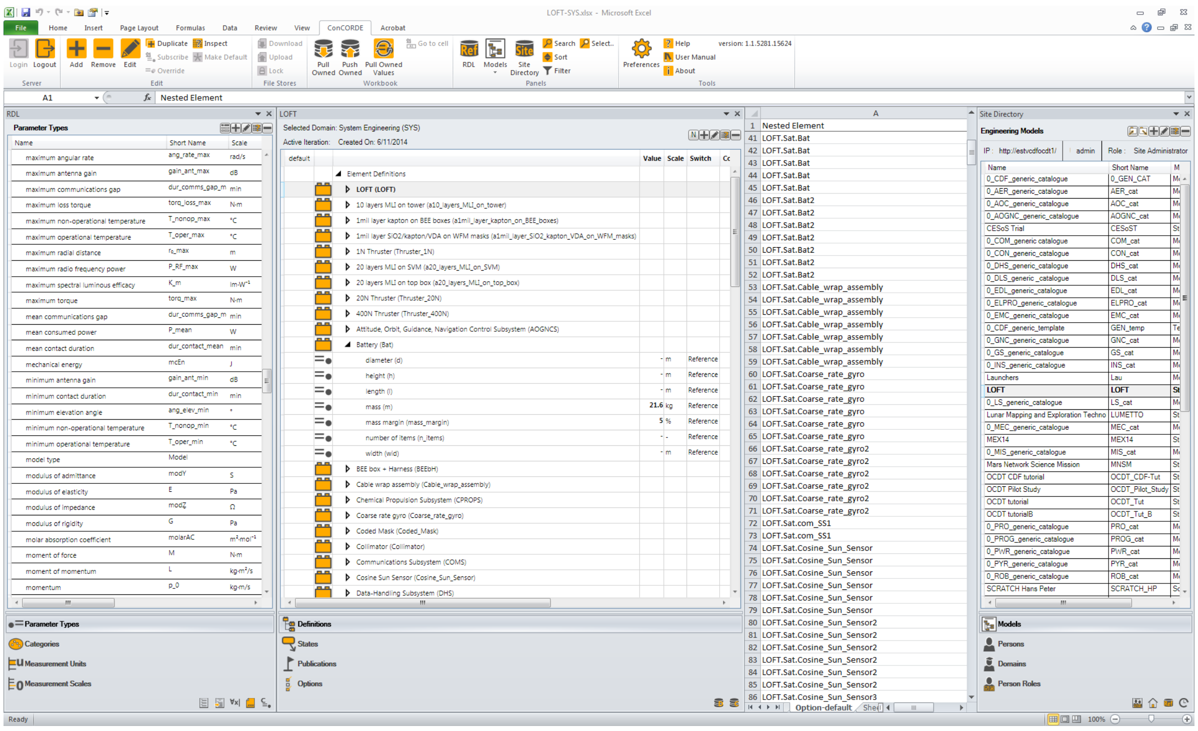
Fig. 3: Screenshot for IBM Rhapsody, showing the tree structured Model View (left) as well as the visual representation of different states of a system (Dishwasher) in an Diagram (central). [11]

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 modelized 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 programme 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 setup 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.

Since the possible approaches to modelling a system and all the options for the appearance are vastly different, guidelines have been defined for related topics to establish conventions of naming and structure as well as to guide the eye by similar layout and appearance. For instance, for space mission related topics, the “ESA SysMLProfile” guideline has been defined by the European Space Agency (ESA) [12]. This compendium is a guideline for instance how to structure the model and how to navigate in-between, or which colour code to use, which makes it easier to access multiple projects once you are generally familiar to the appearance.

Overall, Rhapsody is very complex and it may have limited support to guide the mathematical computation of a design task. However, its strength lies in the representation of an actual complex engineering model, down to the very detail, while the different diagrams of the model allow for a quick and easy overview of the system. This way, the modelling may be very complex, but the information that can be stored is extensive. At the same time, the highest levels of the model may be very visually appealing and intuitive, to get a great overview of the model design.



Parameter Types

Name	Short Name	Scale
maximum angular rate	ang_rate_max	rad/s
maximum antenna gain	gain_ant_max	dB
maximum communications gap	dur_comms_gap_m	min
maximum loss torque	torq_loss_max	N·m
maximum non-operational temperature	T_nonop_max	°C
maximum operational temperature	T_oper_max	°C
maximum radial distance	r_0_max	m
maximum radio frequency power	P_RF_max	W
maximum spectral luminous efficacy	K_m	lm·W^{-1}
maximum torque	torq_max	N·m
mean communications gap	dur_comms_gap_m	min
mean consumed power	P_mean	W
mean contact duration	dur_contact_mean	min
mechanical energy	mcEn	J
minimum antenna gain	gain_ant_min	dB
minimum contact duration	dur_contact_min	min
minimum elevation angle	ang_elev_min	°
minimum non-operational temperature	T_nonop_min	°C
minimum operational temperature	T_oper_min	°C
model type	Model	
modulus of admittance	modY	S
modulus of elasticity	E	Pa
modulus of impedance	modZ	Ω
modulus of rigidity	G	Pa
molar absorption coefficient	molarAC	m^2·mol^{-1}
moment of force	M	N·m
moment of momentum	L	kg·m^2/s
momentum	p_0	kg·m/s

Selected Domain: System Engineering (SYS)

Active Iteration: Created On: 6/11/2014

default	Value	Scale	Switch
Element Definitions			
LOFT (LOFT)			
10 layers MLI on tower (a10_layers_MLI_on_tower)			
1mil layer kapton on BEE boxes (a1mil_layer_kapton_on_BEE_boxes)			
1mil layer SiO2/kapton/VDA on WFM masks (a1mil_layer_SiO2_kapton_VDA_on_WFM_masks)			
1N Thruster (Thruster_1N)			
20 layers MLI on SVM (a20_layers_MLI_on_SVM)			
20 layers MLI on top box (a20_layers_MLI_on_top_box)			
20N Thruster (Thruster_20N)			
400N Thruster (Thruster_400N)			
Attitude, Orbit, Guidance, Navigation Control Subsystem (AOGNCS)			
Battery (Bat)			
diameter (d)	-	m	Reference
height (h)	-	m	Reference
length (l)	-	m	Reference
mass (m)	**21.6**	kg	Reference
mass margin (mass_margin)	**5**	%	Reference
number of items (n_items)	-	-	Reference
width (wid)	-	m	Reference
BEE box + Harness (BEEbH)			
Cable wrap assembly (Cable_wrap_assembly)			
Chemical Propulsion Subsystem (CPROPS)			
Coarse rate gyro (Coarse_rate_gyro)			
Coded Mask (Coded_Mask)			
Collimator (Collimator)			
Communications Subsystem (COMS)			
Cosine Sun Sensor (Cosine_Sun_Sensor)			
Data-Handling Subsystem (DHS)			

	A
1	Nested Element
41	LOFT.Sat.Bat
42	LOFT.Sat.Bat
43	LOFT.Sat.Bat
44	LOFT.Sat.Bat
45	LOFT.Sat.Bat
46	LOFT.Sat.Bat2
47	LOFT.Sat.Bat2
48	LOFT.Sat.Bat2
49	LOFT.Sat.Bat2
50	LOFT.Sat.Bat2
51	LOFT.Sat.Bat2
52	LOFT.Sat.Bat2
53	LOFT.Sat.Cable_wrap_assembly
54	LOFT.Sat.Cable_wrap_assembly
55	LOFT.Sat.Cable_wrap_assembly
56	LOFT.Sat.Cable_wrap_assembly
57	LOFT.Sat.Cable_wrap_assembly
58	LOFT.Sat.Cable_wrap_assembly
59	LOFT.Sat.Cable_wrap_assembly
60	LOFT.Sat.Coarse_rate_gyro
61	LOFT.Sat.Coarse_rate_gyro
62	LOFT.Sat.Coarse_rate_gyro
63	LOFT.Sat.Coarse_rate_gyro
64	LOFT.Sat.Coarse_rate_gyro
65	LOFT.Sat.Coarse_rate_gyro
66	LOFT.Sat.Coarse_rate_gyro2
67	LOFT.Sat.Coarse_rate_gyro2
68	LOFT.Sat.Coarse_rate_gyro2
69	LOFT.Sat.Coarse_rate_gyro2
70	LOFT.Sat.Coarse_rate_gyro2
71	LOFT.Sat.Coarse_rate_gyro2
72	LOFT.Sat.com_SS1
73	LOFT.Sat.com_SS1
74	LOFT.Sat.Cosine_Sun_Sensor
75	LOFT.Sat.Cosine_Sun_Sensor
76	LOFT.Sat.Cosine_Sun_Sensor
77	LOFT.Sat.Cosine_Sun_Sensor
78	LOFT.Sat.Cosine_Sun_Sensor
79	LOFT.Sat.Cosine_Sun_Sensor
80	LOFT.Sat.Cosine_Sun_Sensor2
81	LOFT.Sat.Cosine_Sun_Sensor2
82	LOFT.Sat.Cosine_Sun_Sensor2
83	LOFT.Sat.Cosine_Sun_Sensor2
84	LOFT.Sat.Cosine_Sun_Sensor2
85	LOFT.Sat.Cosine_Sun_Sensor2
86	LOFT.Sat.Cosine_Sun_Sensor3

Engineering Models

IP : http://estvcdfocdt1/ admin Role : Site Administrator

Name	Short Name	M
0_CDF_generic_catalogue	0_GEN_CAT	M
0_AER_generic_catalogue	AER_cat	M
0_AOC_generic_catalogue	AOC_cat	M
0_AOGNC_generic_catalogue	AOGNC_cat	M
CESoS Trial	CESoST	St
0_COM_generic catalogue	COM_cat	M
0_CON_generic_catalogue	CON_cat	M
0_DHS_generic_catalogue	DHS_cat	M
0_DLS_generic _catalogue	DLS_cat	M
0_EDL_generic_catalogue	EDL_cat	M
0_ELPRO_generic_catalogue	ELPRO_cat	M
0_EMC_generic_catalogue	EMC_cat	M
0_CDF_generic_template	GEN_temp	Te
0_GNC_generic_catalogue	GNC_cat	M
0_GS_generic_catalogue	GS_cat	M
0_INS_generic_catalogue	INS_cat	M
Launchers	Lau	M
LOFT	**LOFT**	**St**
0_LS_generic_catalogue	LS_cat	M
Lunar Mapping and Exploration Techno	LUMETTO	St
0_MEC_generic_catalogue	MEC_cat	M
MEX14	MEX14	St
0_MIS_generic_catalogue	MIS_cat	M
Mars Network Science Mission	MNSM	St
OCDT CDF tutorial	OCDT_CDF-Tut	St
OCDT Pilot Study	OCDT_Pilot_Study	St
OCDT tutorial	OCDT_Tut	St
OCDT tutorialB	OCDT_Tut_B	St
0_PRO_generic_catalogue	PRO_cat	M
0_PROG_generic_catalogue	PROG_cat	M
0_PWR_generic_catalogue	PWR_cat	M
0_PYR_generic_catalogue	PYR_cat	M
0_ROB_generic_catalogue	ROB_cat	M
SCRATCH Hans Peter	SCRATCH_HP	Sc

Fig. 4: Screenshot of the ESA OCDT implementation. [14]

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. However, OCDT is openly distributed under an ESA community open source software licence, which allows qualified community members not only the usage of the software, but also its further development. The centre of this community is the OCDT website[13] with a plethora of information, on which this section is based.

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.

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.

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.

Users who are familiar with Excel, will find a relatively easy entrance to OCDT. Apart from that, the common advantages and disadvantages of Excel apply. The open source character of the software can surely be seen beneficial in terms of accessibility, particularly in the frame of teaching, where the potential disadvantage of relying on community support might not play a big role. Another advantage lies in the fact that this software is used by ESA, which can motivate students to engage more with the software.

Tool Comparison

The tools presented here do have some features in common, but vary in how these are applied. Other Features on the other hand are unique for one or the other tool. One major aspect common for all tools is the representation of the design in a product tree. This hierarchic structure is the baseline for storing the design. For Valispace, OCDT as well as for CDP the use of the product tree is the baseline for understanding the system and is used to navigate through the design to ad or extract information. With Rhapsody on the other hand, the diagrams representing views onto the product tree are a fundamental feature allowing a much more sophisticated understanding of the design of the model. Depending on the design philosophy used for the model as well as personal preferences, many tasks of the work can be achieved only in those diagrams.

Significant for engineering work is the handling of units. In particular for space mission, due to the international character of the industry with its numerous preferences in units as of today and even more so in the past. Although all tools allow a definition of units, Valispace is the only one able to compute with them conveniently. Units of the metric as well as of the imperial system can be added without problem, and once used in formulas are automatically converted. In addition to that, calculations will return error messages if the units are not compatible with in itself. This is great, as it allows the engineer to focus on designing the system, and not care about conversion factors and rules. Although basic unit conversion may be implemented for other tools as well, it is by for not as convenient as with Valispace.

An interesting aspect of the design may be the time resolved evolution of parameters over the course of the study. Valispace has here implemented a historic graph for any Vali, which shows a time resolved development. In the CDP, the evolution of objects may be plotted over several iteration steps. These iteration steps can at the same time also be used as “Save points” of the design, to which one could always could go back, for instance when deciding to go into a different direction with the design or if the design itself may be corrupted. Also, this allows a view onto the design to a certain point in time, which may be beneficial for any review process. A similar feature can be used in the EWM with Rhapsody, where Baselines and Snapshots can be defined. For the OCDT, a version history may be achieved by simply copying the file on the system.

The main objective of our course is to get hands on experience to the concurrent engineering process, so the tools have to compare on how they support this aspect. With Valispace, all users access the same Database, and changes are redirected in real time to any other user accessing the database. Therefore, fast and direct exchange is possible. With CDP, Rhapsody and OCDT, the approach is somewhat more streamlined. While in CDP and OCDT the changes have to be published by a user, in Rhapsody (using EWM) the certain part of the model containing changes has to be uploaded. This upload may generate conflicts that have to be resolved. Even though this approach may be a bit slower, it significantly reduces noise to the user, and allows for an easier performance optimisation of the tool.

File repositories are common for all tools. For Valispace there is also a discussion function connected to Valis itself, including notifications once a Vali is updated, keeping discussion limited to users concerning a certain Vali and saving these discussions for later references.

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 share disconnected information by noting it 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 designs, this may be a valid option, since students don't have to get acquainted with a new tool and can focus solely on the design of the respective subsystem they are responsible for and its interaction with other subsystems/disciplines.

However, since the design will get complex by itself in no time, it would quickly get unorganized. In addition to that, for any non-centralized design study over a longer period of time, this cannot be an option.

Nevertheless, we wanted to include this option, as some people might find it favourable in their conditions, where they might have no time to introduce a dedicated software or the means to effectively utilise one. While this option surely has rather narrow limitations, it remains a viable option if the software utilisation itself is not one of the learning objectives. When students of our course avoided the software, we didn't enforce its use. The CD methodology could still be learned well up to a certain model complexity.

5. 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. This means that 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 and what not.

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 includes the availability 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 were problems with the stability of the software or serious latency in the data synchronisation, the acceptance drops. Thus, the software and its 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.

Next to these five criteria (usability, complexity, interface, performance and manageability) the analysis could be extended by further aspects. This could involve the supported interfaces for the implementation of further software solutions (such as design and simulation software). However, we consider this very user-specific and thus did choose not to include it.

Another aspect might be the requirements of the software towards the hardware infrastructure. We didn't include this as the options we consider in this work don't show significant differences that would allow a meaningful differentiation.

Moreover, some might consider available licenses and corresponding prices important. While we agree, we excluded this point as everyone will have their own threshold and prices change frequently.

Lastly, one might consider how widely the different software tools are used within a certain domain. Clearly, learning the utilisation of a more commonly used software would overall have a higher impact on the students than little known software. Nevertheless, this can also change over time and the level of expertise the students can gain on any software solution during the course is rather limited. We also invite the students to check out other tools outside the course to find their own preference.

The five criteria presented are all significant in their very own aspect, which concludes that the failure to fulfil 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 on these evaluation criteria.

Evaluation

Due to the limited time available during our 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 the hang of 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 very 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. Once this one is covered, it may seem to 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 and connected, to be stored in the model. From our experience, the level of expertise and therefore the level of usage will differ much stronger for the CDP and Rhapsody than for Valispace and OCDT, simply due to the different background and interest of the students. This higher difference makes it more challenging for the tool to be actually used for exchange between the students in the course.

The OCDT, Valispace as well as CDP are designed to aid the design of space related missions. Although other studies may also be conducted, numerous features are implemented to supporting this general field of study, including for instance 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 point in time. 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 students. In particular with Rhapsody, many information has to be added up front, but an experienced user may be able to present this information visually very appealing as well as sorted, using different types of diagrams.

In the description of the tools, we distinguished the functionality of updating the model. 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 changing a single parameter can result in the update of a multitude of parameters, which may be resourceful and take more and more time with increasing model complexity.

For the CDP, the model will only be updated by a top-level user. This makes the system more discrete, but also requires less data to be 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 educators' 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 much more flexibility in planning the courses and allowing the students to work from home. 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.

A basic evaluation of the criteria's is summarized in table 1, which provides a general overview of the viability of these softwares for the requirements discussed initially. For the evaluation, a simple grading system of [++, +, 0, -, --] was used, were [++] represents the best implementation of the criteria, and [--] the worst.

Since the requirements imposed on the tool will drastically influence the results of the evaluation, no summation of our grades is included and the reader is invited to adapt the evaluation to their individual requirements and setting. The grades presented shall be understood as indications for a single semester course at university level.

Tab. 1: Evaluation of software tools for the discussed criterias, based on the requirements imposed by the course structure

	Valispace	**CDP**	**Rhapsody**	**OCDT**
Usability	++	0	--	+
Complexity	+	+	--	0
Interface	++	0	-	+
Performance	-	++	+	+
Manageability	+	+	-	0

6. Conclusion

Multiple tools have been used by the authors to conduct concurrent design studies in a university level course with students. The authors' experience with the software tools is obviously 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. Furthermore, the tools are under constant development, which means that certain aspects may have already changed since their evaluation.

For the course at hand, the software implementation by Valispace is our preferred solution as of right now. The tool grants easy access and requires only a minimum of initial training, which also can be self-taught with the available tutorials, 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 the reporting tool are additional benefits for our course. From our experience, the tool provided the best introduction to the general concurrent engineering approach for the students, and resulted in the greatest amount of data shared with such a tool.

Acknowledgements

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

Literature

[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 Help Desk.” [Online]. Available: https://docs.valispace.com/vhd/Components-Module.1505230901.html.

[6] “Valispace Tutorial.” [Online]. Available: https://docs.valispace.com/vhd/Fan-Tutorials.1512243215.html.

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

[8] 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.

[9] A. Mincolla, “Space Systems of Systems Generative Design Using Concurrent MBSE,” Stoockholm, Sweden, 2020.

[10] “IBM Rhapsody Website.” [Online]. Available: https://www.ibm.com/products/systems-design-rhapsody.

[11] “IBM Webpage.” [Online]. Available: https://www.ibm.com/de-de/products/uml-tools.

[12] M. Kretzenbacher, “Generic ESA SysML Metamodel and Toolbox for Space Systems Modelling [EUCL-EST-TN-1-014],” 2017.

[13] “OCDT Website.” [Online]. Available: https://ocdt.esa.int/.

[14] H. P. de Koning, “Experiences from Developing COncurrent Multi-Disciplinary MBSE,” Noordwijk, The Netherlands, 2015.