Structuring a Cross-Reality Environment to support a Concurrent Engineering Process for Space Mission Concept

6th International Conference on Systems & Concurrent Engineering for Space Applications

- SECESA 2014 -

08-10 October 2014

Vaihingen Campus, University of Stuttgart
Germany

Christopher Shneider Cerqueira[(1)], Ana Maria Ambrosio[(2)], Claudio Kirner[(3)] , Fabiano Luis de Sousa[(4)]

[(1)(2)(4)]National Institute for Space Research - INPE
[(1)]Space Engineering and Technology (ETE) – Space Systems Engineering and Management Post Graduation Program (CSE)
[(2)(4)]Space Engineering and Technology (ETE) – Space Systems Division (DSE)
Av. dos Astronautas, 1758. São José dos Campos – SP – Brazil
e-mail:christophercerqueira@gmail.com[(1)], ana.ambrosio@inpe.br[(2)], fabiano.sousa@inpe.br[(4)]

[(3)]Federal University of Itajubá – UNIFEI
Institute of Mathematics and Computer Science (IMC)
Av. BPS, 1303. Itajubá – MG - Brazil
e-mail: ckirner@gmail.com

INTRODUCTION

Space Mission Concept tasks requires the use of several modelling and simulations resources. The ECSS (European Cooperation on Space Standardization) Modelling and Simulation Technical Memorandum [1] indicates that simulations can support life-cycle tasks as: (i) design, (ii) feasibility and performance analysis, (ii) functional validation, (iv) software validation, (v) equipment validation, (vi) AIV (Assembly, Integration and Verification), and (vii) training & operation and further tasks. Such Memorandum also recommends that the multidisciplinary engineering team consolidate a mission and system concept using a virtual system model with a generic spacecraft, ground and environment models containing key parameters of each required domain, as: power, AOCS (Attitude and Orbital Control Systems), mission analysis, mechanical, thermal, etc.

Traditionally, each specialist engineer and discipline has a specific modelling and simulation tool set and environments, and these are placed into each engineer's computational workspace environment. These workspaces are not shared, in sessions or development, usually only a repository keeps the engineers shared information. This modelling and simulation toolset and computational environments might include: Matlab/Simulink, SciLab, STK, Orbiter, SINDA/FLUINT, CAD (Computer-Aided Design) applications, PSpice, Modellica, AMESim, LabView, MS. Excel, etc.; as well as their own nature of simulation: continuously, by events, electronic, mechanic, computational, etc.

The way each specialist engineer interacts with the tool set are, usually, based in WIMP (Windows, Icon, Menu and Pointer) metaphor, instead of a metaphor closer to the simulated system/sub-system. The WIMP metaphor extracts the abstraction of the desired system to an amount of tables, graphics, and in the best case diagrammatic representations, which allows navigation and exploration in two dimensions, for example, creating a Simulink model consists in search for functional blocks, to drag and drop into a main workspace area, interconnect the blocks, point and click to attribute setting by menus and text fields. On the other hand, a CAD based simulation (thermic, mechanic, CFD - Computational fluid dynamics, etc.) has a metaphor closer to the real system, as it represents the necessary three-dimensional system model, allowing point and click in the "three-dimensional" space in order to attribute setting.

NUI (Natural User Interfaces) researchers defines that collaborative works, as concurrent engineering, demands virtual and transparent software coupling, in real time work, using a shared visualization media [2]. So each specific engineer

uses a personalized workspace which controls its toolset and shows the shared information flow, to all collaborates at the same time. The technology that allows this type of scenario is called Cross-Reality [3].

This work shows an intuitive human computer interface environment structure proposal to increase cohesion among the Concurrent Engineering participants using NUI. This work proposes an infrastructure to interconnect several specific tools/environments through a generic component model, devices to create the Cross-Reality user interface environment, the user interaction methodologies with the visualization of the links from the shared information.

Such approach could be very useful in concurrent engineering environments, as the ESA Concurrent Design Facility or the NASA JPL's Team-X. It is planned to be applied at the current PJESOProM, a Project for Simultaneous Engineering and Multidisciplinary Design Optimization, in Portuguese, carried on INPE´s Space Systems Division.

This work is divided in six sections, which (i) contextualizes the scope of Concurrent Engineering and Model-Driven Architecture; (ii) creates a Simulation Center scenario; (iii) defines the information componentization; (iv) defines Cross-Reality and mixed realities; (v) shows the interaction strategies and usability assessments; and (vi) provides the conclusions.

CONCURRENT ENGINEERING AND MODEL DRIVEN ARCHITECTURE

Space Mission Analysis and Design (SMAD) is an iterative process of gradually refine requirements, functions and function allocations to achieve the requested needs of a space mission [4]. The analysis and design results come from initial needs and goals, through possible mission concepts and architectures, which comprises costs and trades-offs, in order to provide a set of scenarios to the decision-takers stakeholders.

An approach to create the set of Scenarios is the Concurrent Engineering, which is an agile based method, to distribute the study and create system configurations [5]. Traditionally CE case studies use Microsoft Excel for the initial phases, building Integrated Design Models (IDM). Reference [6] presented advantages and drawbacks of Excel based design versus Model Driven Design, discussing some of the software infrastructure requirements to a CE Environment, as:

- General Requirements: team management, requirements management, scheduling module, documentation and data management.
- Collaboration related Requirements: distributed work, authentication and authorization, component breakdown, parameter management, calculation management, central data, version control, import/export, intuitive use and documentation.

As an evolution from the Excel based IDM, [7] and [8] proposes virtually modelled space mission systems. The models are abstract representation of the reality, from: earlier missions, new designs and/or Excel's translation. Such myriad of model allows to create a dynamic simulation catalogue that helps the engineers to create the scenario concept and architecture. This approach is called Model-Driven Architecture (MDA), where the system is platform independent. The engineers can model the logic, functionality – attached to goals and requirements, and use automatic task tools as auto-code and auto-doc. The models can be saved or described by metafiles, usually Extensible Mark-up Language (XML), which describes its characteristics and can evolve and gain granularity through the life-cycle.

A SIMULATION CENTER

Based on the [8][9] reference, we propose a Simulation Center Architecture able to access multiple data, from multiple tools, control a data repository, etc., in order to fulfil the uses not only in the CE process to analysis and design definition, but as well as in the all life cycle. We propose a classification of the architecture elements as follows:

- *Design Elements*: Each engineer designs (in any phase) with its own set of tools, he can´t be forced to use a unknown toolset that will supress his skills, however most of them allows to export/import commonly used file formats. The Design Controls have to allow the interchangeability of file formats, functions to create the metafiles, and functions to access other simulations instances to co-simulate front others' designs.
- *Simulation Elements*: The simulation has to provide an infrastructure to connect multiple models, and co-simulators software, as well as Hardware-In-The-Loop elements (emulations or real equipment under evaluation/test). The Simulation controls have to provide simulations instantiations with configurable infrastructure components, execution controls, parameters configuration, time/step configuration, logs,

simulation visualization and information services (to execution-time publish and request of parameter conditions).

- *Operation Elements*: The operation phase uses an operational simulator implementation that uses the life-cycle results of modelling to simulate the real spacecraft that was placed in orbit, as well as accommodation of models by update changes during the operation. The Operations Controls accesses a current picture of the spacecraft from the simulation controls in order to test procedures and follow spacecraft conditions through the received telemetries.
- *Repository Elements*: The repository has to allow version controlled meta information, that indicates models, sources of models, interchangeable formats, documentation as well as auto-code to the desired tools, simulation architecture and embedded hardware apps; and auto-documentation using standard compiled documentation, as Latex, or XML-based documents as MS. Excel (.xlsx), MS. Word (.docx) and MS. PowerPoint (.pptx).
- *Concurrent Session Elements*: In Concurrent Sessions the engineers restore their design results from the meta information and interconnect with other's designs, allowing them to compare docs and run stored simulation scenarios to show the progress and the conceived concepts and architectures. The Concurrent Controls have to allow access of multiple engineers, and allow multiple viewpoints of the scenario, so each discipline engineer feel familiar to his workspace.

Fig. 1 shows the interconnection of the major elements of the proposed Simulation Center Architecture.

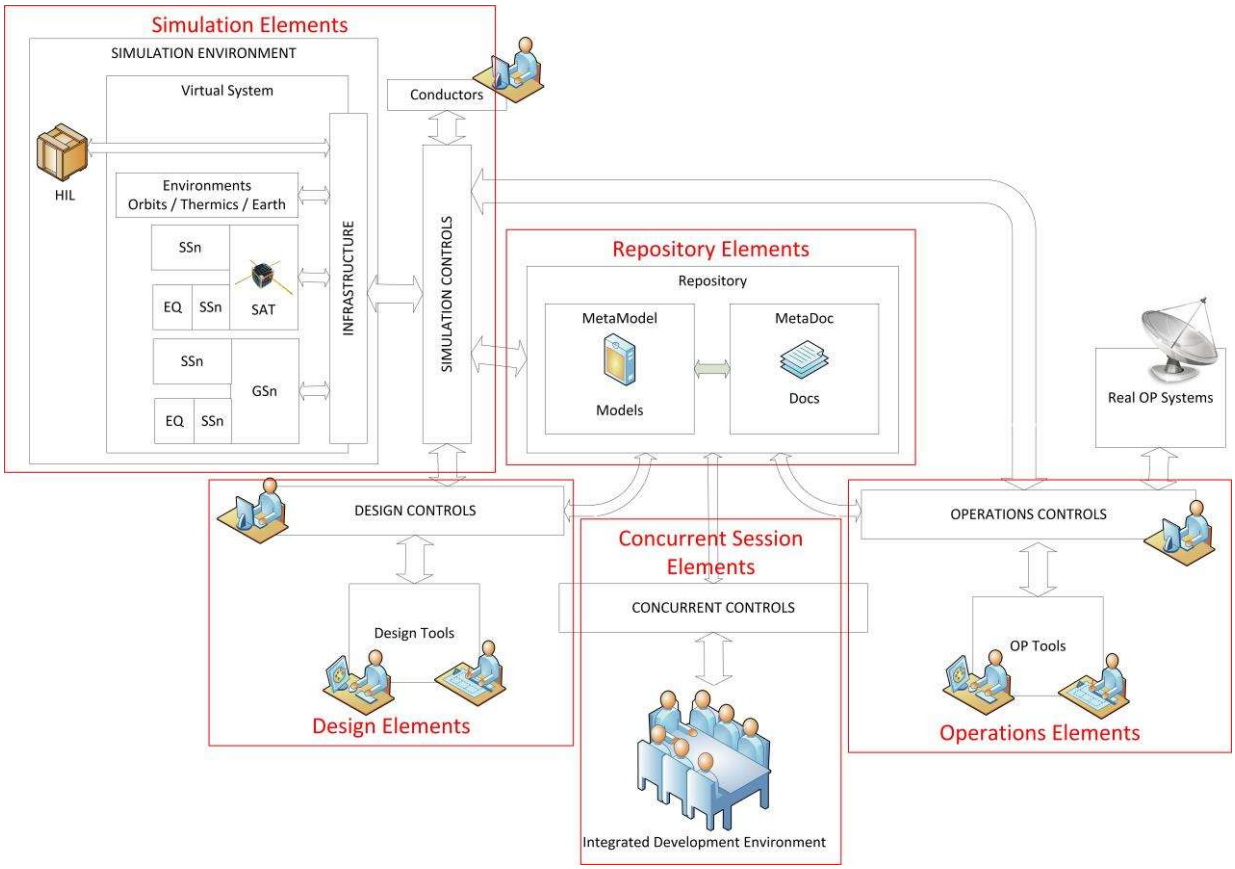


Fig. 1. Proposed Simulation Center Architecture.

INFORMATION COMPONENTIZATION

The references [8] [9] [10] discuss the information componentization in XML files. The meta files in XML are useful to represent shared information through the applications and allow translations through the activities. The XML description strategy was tested at an operational simulator [10]. In this attempt each node (seen in Fig. 2a) describes the identification, related parameters, dynamics, geometric information, and subcomponents relationship, as seen in Fig. 2b, and into the code snippet of Figure 2c. Later, the operational simulator XML strategy attempt allowed dividing the

software in three layers: Infra-structure – containing the application codes, communication issues; Structure – containing the XML files that are read by the infra-structure software to mount the application; and the Content - containing the content pointed by the XML files to the user-interface (3D models) and to the simulator (Dynamics).

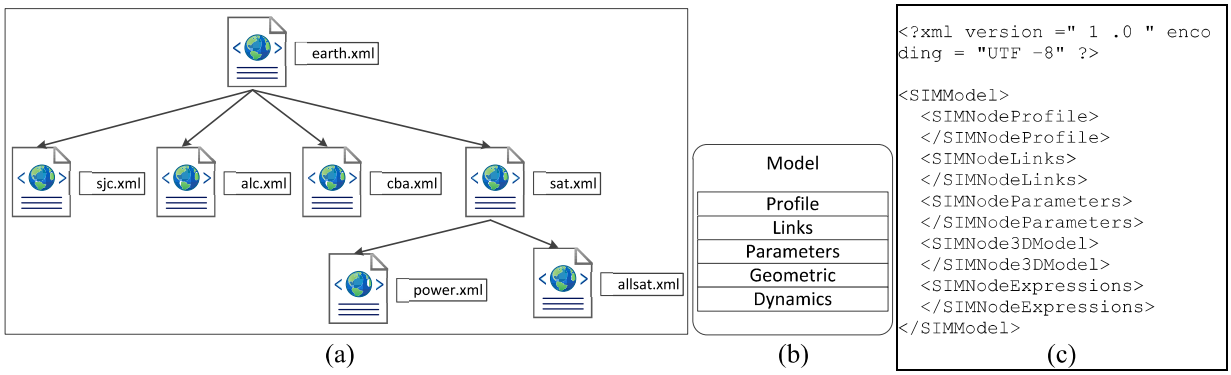


Fig. 2. XML scheme to an operational simulator attempt. (a) XML relation, (b) XML composition, (C) XML tag scheme example.

Componentizing allows splitting the disciplines goals/allocated functions in components that are interconnected to create a desired architecture, allowing fast changes, easy component testing and traceability. Abstract components can allocate broad definitions as systems or subsystems and more detailed in equipment and its physical components, as well as the inter-relation of association (communication link among the components by its parameters), heritage (a component is a father of other component, sharing its characteristics) and container aggregation (a component has internal components) – specializing as required. Fig. 3a shows the proposed component with its interfaces: (i) Profile – containing name, identification, allocated requirements, file names, links to files, link to the model catalogue; (ii) Parameters – input and output parameters as well as internal parameters used in the component; (iii) Dynamics – if it's a known simulation code, a white box model, to run directly by the simulation structure, the component dynamics will be shown, otherwise the links to the co-simulator; (iv) The State-machine – simple event-driven representation of the component, to inspect white and black box dynamics, and to help in the automatic generation of test-cases in the assembly life-cycle phase; (v) Flags - common flags indicating if the component is active, visible, writeable, etc.; (vi) Geometric – information about 3D models, mass, and the local (component to superior component)/global(component to global system) positioning on the simulated model. Fig. 3b shows the proposed association of parameter inside a component container which aggregate the models that derived from generic system components, colours can be used to match with the state machine status.

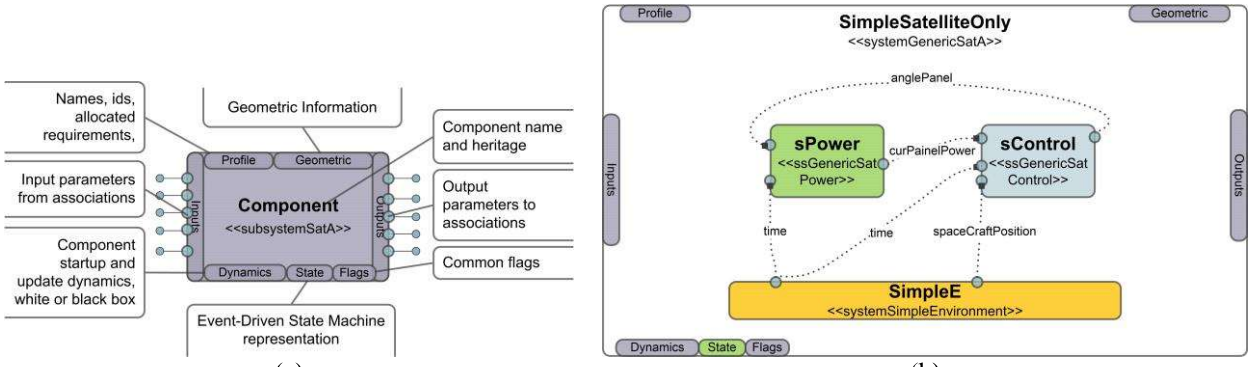


Fig. 3. Example of component layout. (a) Generic component and its elements. (b) Example association of components inside of a higher level component.

CROSS-REALITY AND MIXED REALITIES

There are several interaction natures used by a human being to manipulate real elements that are augmented by virtual layers. The elements of the real and virtual world cross themselves with computational empowered elements such as sensors and actuators, creating a myriad of words such as: real tangible world, augmented reality, virtual reality, distributed virtual worlds, ubiquitous worlds and Cross-Reality. Fig. 4 shows the crossing of real world objects – touchable and tangibles, with virtual objects (creating Augmented Reality), or inserting computational and electronic intelligence (creating ubiquitous worlds). The real world can be completely replaced by a virtual world - through

immersive devices, allowing interaction with virtual objects only (creating virtual reality), and collaborative and distributed tasks (creating distributed virtual worlds). The intersection among real, virtual and computational objects, with bidirectional communication among the worlds is called Cross-Reality. [3]

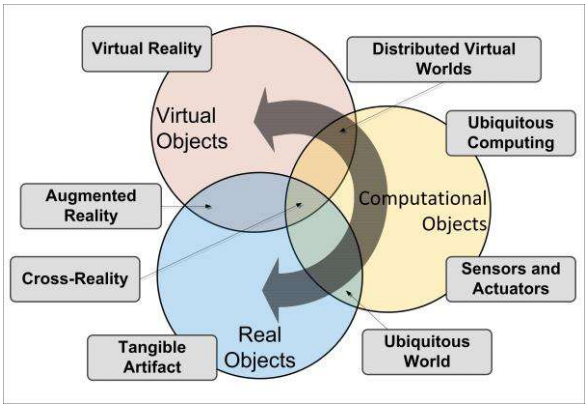
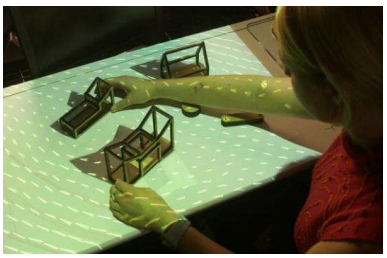


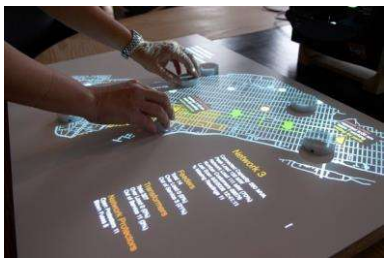
Fig. 4. Kirner's Diagram of multiple realities. Source: [3]

The Cross-Reality environment is formally defined by [3] as "a ubiquitous mixed reality environment that comes from the from the fusion of a network of sensors and actuators (which collects and sends data related to the real world) with shared virtual worlds, using an augmented reality user interface, where the exchange of information is bidirectional between the real and virtual world". Cross-Reality can have two types: Non Overlapped Cross-Reality – the virtual elements are not overlapped with the real elements; Overlapped Cross-Reality – the virtual elements are overlapped with real elements.

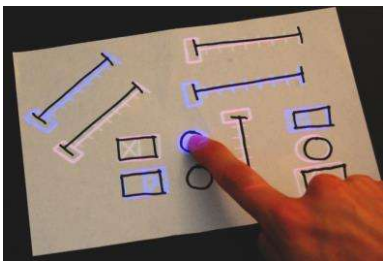
Fig. 5 shows examples of projection based cross-reality environment interactions, where: (a) is the Luminous Room [11], which allows natural interaction with proportional objects facing wind and natural light simulations to architecture design; (b) is the Sensetable [12], which allows natural interaction based in handlers that interact based in proximity and rotational movements to control maps, and simulation parameters, and (c) is the Sketch Synth [13], which allows natural manipulation of hand drawn interfaces augmented with the digital simulation elements.



(a)



(b)



(c)

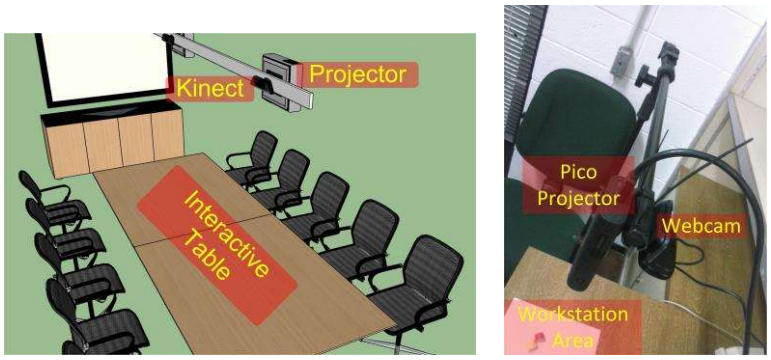
Fig. 5. Examples of Projective Cross-Reality Environment interactions. (a) Luminous Room example, (b) Sensetable example, and (c) Sketch Synth example.

INTERACTION STRATEGIES

Toolset to the environment

Given the proposed Simulation Center Architecture, this works structures a projection based cross-reality environment. To create this kind of environment it is necessary a set of hardware and software to deal with computer vision interaction, and projection of rich manipulable content.

The devices available by the time of writing this work are: the MS. Kinect (mainly because of the depth camera facilities), a common projector (however a stereoscopic projector is highly desirable) and a computer to process the interactions. The physical environment project is shown in the diagram of Fig. 6a. A simpler single workspace case-study hardware environment is shown on Fig. 6b, it uses a 3M Pico projector and a Logitech HD webcam.



(a) (b)

Fig. 6. Hardware environment. (a) Proposed hardware environment. (b) Case study environment.

The software framework that is being used to create the CE environment is the openFrameworks (oF) [10]. oF is an open source interoperable C++ layer that implements a set of libraries to create and control rich content applications. oF allows that the same code can be compiled through Linux, Windows and Mac operational systems and as well as embedded platforms as Android, iOS and embedded Linux. oF shows as a more appropriated choice, among some possible frameworks as Java (Processing) or Action Script (Flash) counterparts. The oF community helps to extends the main functionality with add-ons that handle specific items as: animations, hardware, computer vision, user interfaces, graphics, machine learning, physics, sound, utilities, networking, and etc.

Identifying the data visualization and manipulation in CE

From the PJESOProM initiative and references [4] [6] [7] [8], was possible to identify seven initial major types of visualization/manipulation structures that can be useful to create the CE scenario: trees, component relationship diagram, state-machine diagram, Cartesian charts, text editors, tables/menu, and 3D objects. These structures can be achieved using, multiple views, reconfigurable disk trees, scene graphs, panels and 3D models, some of those structures are show in Fig. 7 shows some examples, which was explored in the [10]: (a) multiple views, (b) reconfigurable disk tree, (c) scene Graphs, (d) auto-created panels, and (e) 3D models.

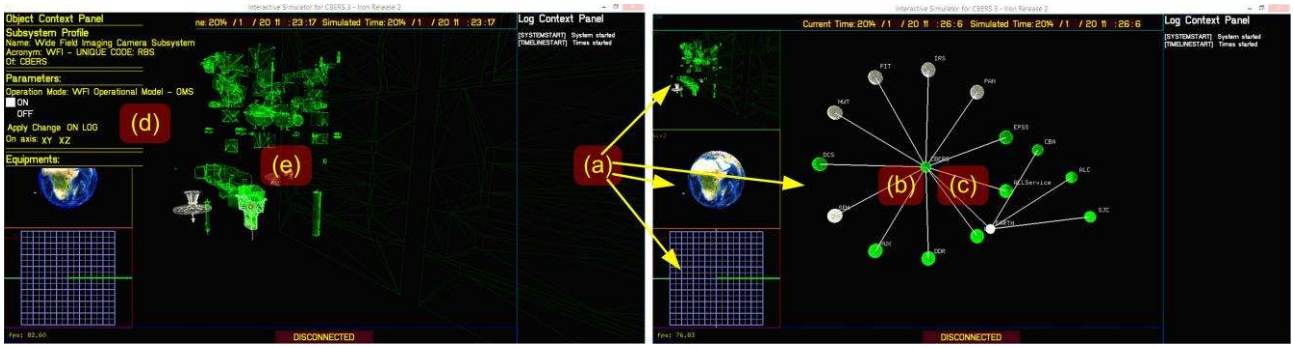


Fig. 7. Examples of implementation structures. [10]

Using NUI projected objects demands rather bigger interactive objects than on desktops as it depends on the projector and tracking camera resolution. Note: All these objects also work well on desktop and touchscreen environments.

Interaction Strategies Discussion

There are some issues to be analysed concerning the interaction concepts applied to CE, as: single and multiple workspaces, creating scenarios and session operations.

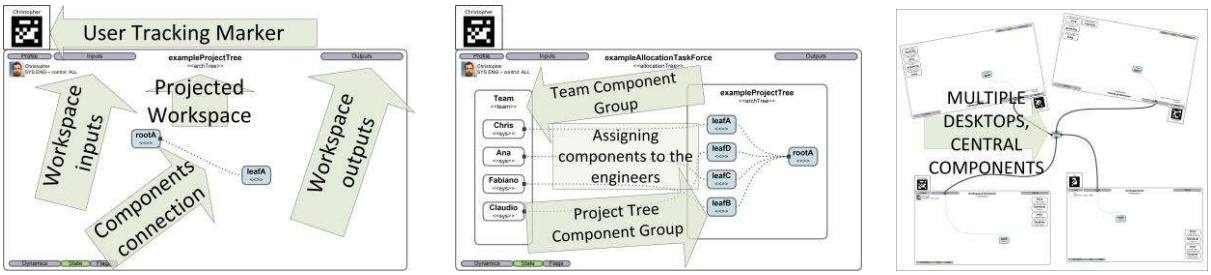
Single and Multiple workspaces

Two modes of use cases are allowed: single and multiple workspaces. Single workspaces allow a single engineer to work in concepts, architectures, trade-offs and designs of his study discipline. Multiple desktops allow multiple

engineers to work in blended study disciplines, to locally optimize dependable trades and prune trade trees, or to present the study results with co-simulated scenarios in sessions.

Creating a scenario

The initial task on the CE process is to create the scenario elements in the tool; the system engineer creates the first component node and assigns a component template or a documentation template. Component structures the architecture and documentation structures the needs, objectives, constraints and requirements. Documentations derive from needs to requirements through objectives and constraints creating trees/graphs, components derive from the higher study abstraction to the final allocation function through systems, subsystems (or disciplines) and equipment also creating trees/graphs. As the basic layout is created, the tree/graph branches are assigned to the discipline expert engineers. So they use their own design tools and export to the repository or prototype using given tools, on desktops. Figure 8a shows the storyboard of some of the possible creating scenario processes: (a) components aggregation, (b) expert assignment, (c) throwing to the engineers on multiple workspaces. These standardize component/function assignment may help to distribute functions through engineers, to document the models into the repository´s designs catalogue, control co-simulations, and prepare discipline input/output relations to the sessions review.

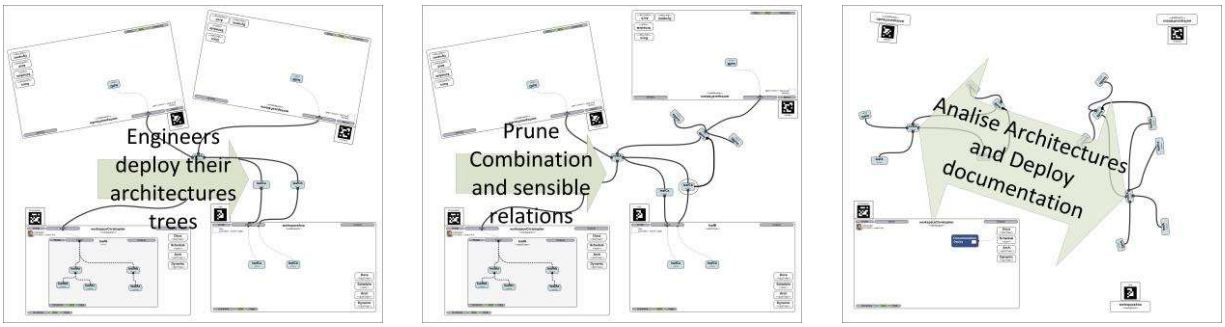


(a) (b) (c)

Fig. 8. Creating a scenario storyboard.

Sessions

In a session meeting the engineers show the trade trees of documentation and components, sequentially using a planned criteria they incorporate it to get the full domain study picture ([4] indicates to follow orbit, payload, service bus, launcher and orbit transfer, and operations). As each engineer attaches the studied branch, the workspaces attach in each other by the higher levels architecture layouts, so each expert engineer prune the sensible combinations and adjust the possible dependable trades. Deciding the concepts/architecture options, the Simulation Center Architecture provides the auto-doc from the meta information files containing the component dynamics and structure, and the requirement tree options to the decision-takers. Figure 9a shows the storyboard of a session operation, where: (a) attaches the first study branch, (b) prune the combinations and sensible relations, (c) close the session and request auto-doc. This automatic interconnection of disciplines in the session time may help the engineers: (i) to prune their integrated trade trees, (ii) to find dependable trades, (iii) to zoom in/out desirable components, and (iv) to simulate and observer the integrated evolution of a concept/architecture.



(a) (b) (c)

Fig. 9. Session Operation Storyboard.

CONCLUSIONS

This work presented elements that may comprise a future CE user interface and discussed the problem in a natural interface perspective for a Simulation Center Architecture perspective to provide the backing tools to componentize and co-simulate early concept designs.

This natural user interface exploration has its basis in usability tests, including potential stakeholders (design engineers, simulator developers, operation engineers, and system modellers) at INPE, using a prototype operational simulator to collect requirements and to measure the interaction, interfaces, sensorial, association, representation and motivation aspects of a natural user interface. These potential stakeholders prefer diagrammatic representations instead of 3D equipment's placement to keep up with the simulation data, the natural interaction in contrast to the WIMP interaction was highly desirable and described as easier and intuitive. This exploration work is available at [10].

We expect that the cross-reality environment structure and concepts can contribute to PJESOProM disciplines simulations and parameters interconnection during concept designs and sessions. This environment might help the expert engineers to look, touch and control their designs in order to choose better concepts/architectures rapidly, prune the study trees, and generated the desired documentation to decision takers approval.

REFERENCES

[1] European Cooperation for Space Standardization (ECSS), ECSS-E-TM-10-21A - Space engineering - system modelling and simulation. Noordwijk, The Netherlands: ESA-ESTEC, 2010.

[2] D. Wigdor and D. Wixon. 2011. Brave NUI World: Designing Natural User Interfaces for Touch and Gesture (1st ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[3] C. Kirner, C. S. Cerqueira, T. G. Kirner. Using augmented reality cognitive artifacts in education and virtual rehabilitation. In: Virtual Reality in Psychological, Medical and Pedagogical Applications. InTech, 2012. p. 247–270. ISBN 9789535107.

[4] J.R. Wertz and W. J. Larson. Space Mission Analysis and Design. Springer Netherlands.

[5] I. Sommerville. 2010. Software Engineering (9th Ed.). Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

[6] H.Schumann, H. Wendel, A. Braukhane, A. Berres, A. Gerndt, and A.Screiber. Concurrent Systems Engineering in Aerospace: From Excel-based to Model Driven Design. 8th Conference on Systems Engineering Research. Hoboken, NJ.

[7] P. Fritzen, S.Kranz, P. Plas, M. Arcioni. Rapid Simulation Development for Concurrent Engineering. SpaceOps 2006 Conference. AIAA 2006-5912.

[8] V. Schaus, P. M. Fischer, D. Lüdtke, A. Braukhane, O. Romberg, A. Gerndt. Concurrent Engineering Software Development at German Aerospace Center -Status and Outlook- 4th International Workshop on Systems & Concurrent Engineering for Space Applications (SECESA), Lausanne, Switzerland, October 17–19, 2010

[9] European Cooperation for Space Standardization (ECSS). ECSS-E-TM-10-25A - Space engineering – Engineering design model data exchange (CDF). Noordwijk, The Netherlands: ESA-ESTEC, 2010.

[10] C. S. Cerqueira. Virtual Reality Techniques Employed Into Satellite Operational Simulators Interaction. 2014. 179 p. Dissertation - Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2014. Available at: <http://urlib.net/8JMKD3MGP5W34M/3G28FG2>

[11] J. Underkoffler; D. Chak, D. Santos, H. Ishii. I/O Bulb and Luminous Room – Tangible Media Group. 1998. Available at: http://tangible.media.mit.edu/project.php?recid=73.

[12] J. Patten, G. A. Pangaro, M. Reynolds, J. Alonso, J. Panganiban, J. Hines, H. Ishii. Sensetable – Tangible Media Group. 2002. Available at: http://tangible.media.mit.edu/project/sensetable/

[13] B. Keyes. SketchSynth: A Drawable OSC Control Surface. Carneggie Mellon University. 2012. Available at: http://golancourses.net/2012spring/05/13/billy-keyes-final-project-sketchsynth/

View publication stats