

Developing an Open Platform for Democratised MBSE

Presented by



22-11-2022 (Toulouse, France)

founders@perpetuallabs.io

Agenda:

- Introductions
- Motivations
- Platform Overview and Architecture
- Case Study
- Conclusions

Who are we?
&
Why are we here?

Perpetual Labs Team

Gianmaria Bullegas PhD (Founder and CEO)



Expertise

- 10+ years experience Aerospace and Defence
- 5+ years in early-stage ventures.
- PhD Aerospace Eng Imperial College

Maged Elaasar PhD (Advisor)



Expertise

- International leader in MBSE
- 25+ years experience as Principal Engineer Systems Engineering Division NASA JPL

Omar Nachawati PhD (co-founder and CTO)



Expertise

- 12+ years experience in Software architecture, development and operations
- PhD Computer Science George Mason University

Prof Peter G. Larsen (Advisor)



Expertise

- International leader in Model Based Design
- 30+ years experience in Industry and Academia
- Head of CPS lab at Aarhus University

Steven Jenkins PhD (Advisor)



Expertise

- International Leader in Digital Engineering
- 25+ years experience as Software Architect
- Software lead, IMCE program at NASA JPL

Hans Peter de Koning (advisor)



Expertise

- 25+ years experience in MBSE and MDE
- Previously Lead Systems Engineer at ESA ESTEC

David Toluhi PhD



Expertise

- Semantic Technologies and AI
- PhD Computer science University of Manchester

Andrey Vasilyev PhD



Expertise

- Systems Modelling and Controls
- PhD Systems and Controls, Loughborough University

Robin Kennedy-Reid



Expertise

- DevOps and HPC.
- Ms.C. Physics University of Bristol

Strahinja Gligovic



Expertise

- Front-end web development

Rebecca Thornton (CFO)



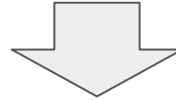
Expertise

- 12+ years experience in Finance
- Commercial Strategy
- Chartered Accountant

Why are we here?

Mission:

Make modelling collaborative and accessible to
as many people as possible



GitWorks
Community

Motivation

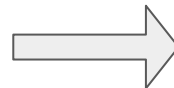
Challenges with Development of complex Engineering Systems

- Data and knowledge silos across supply chain
- Traceability and trust of design data
- The Digital Thread
- Lack of adoption of Model-based design
- Low design reusability
- Silos inhibit innovation and collaboration across organizational boundaries
- ...
- **This is nothing new! You've probably heard it a thousand times by now...**

Problem Statement

There are three fundamental technical challenges

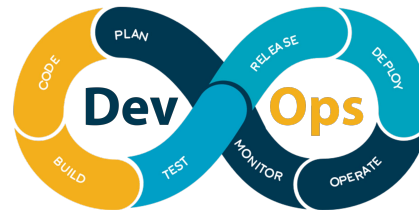
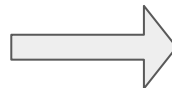
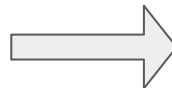
- **Version Control, Change Management and traceability**
 - Without an overarching strategy, a system description becomes a disorganized and untrustworthy collection of information artifacts.



Problem Statement

There are three fundamental technical challenges

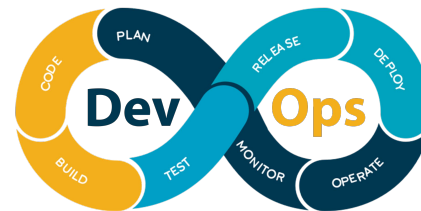
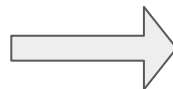
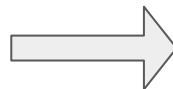
- **Version Control, Change Management and traceability**
 - Without an overarching strategy, a system description becomes a disorganized and untrustworthy collection of information artifacts.
- **Repeatability, durability and efficiency**
 - Without this, it is impossible to perform audits and repeat analyses. This is important to maintain confidence in the design and analysis over time and potentially reuse it



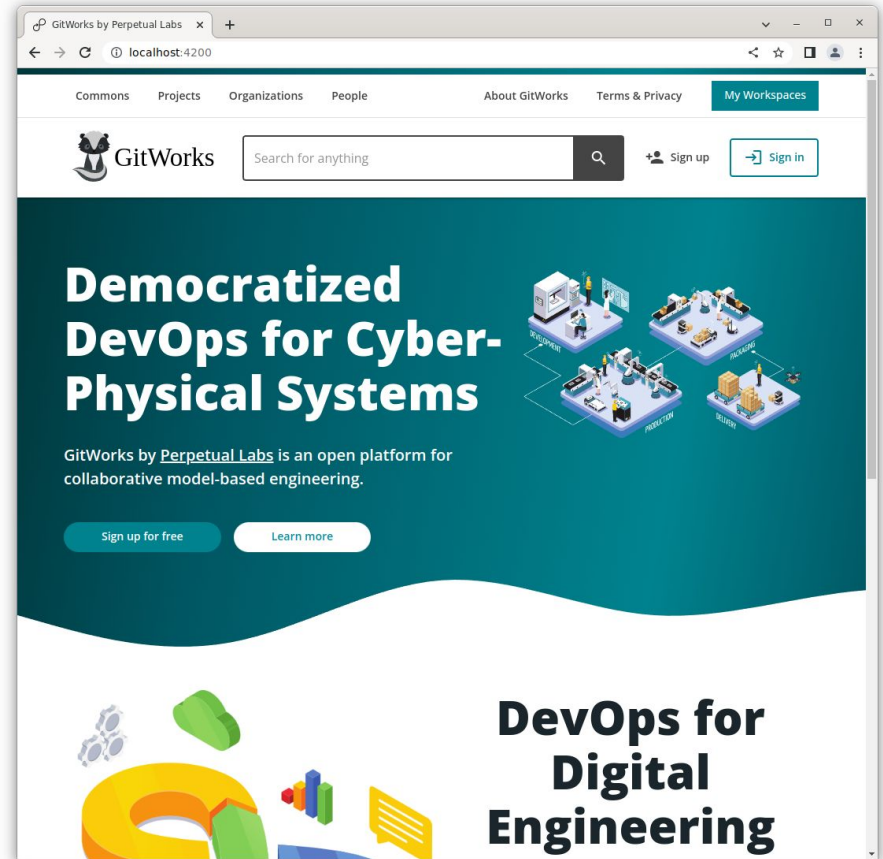
Problem Statement

There are three fundamental technical challenges

- **Version Control, Change Management and traceability**
 - Without an overarching strategy, a system description becomes a disorganized and untrustworthy collection of information artifacts.
- **Repeatability, durability and efficiency**
 - Without this, it is impossible to perform audits and repeat analyses. This is important to maintain confidence in the design and analysis over time and potentially reuse it
- **Interoperability**
 - System knowledge and data is scattered in many different artefacts (models, documents, emails) and often out of sync.
 - Artefacts have different syntax and tools have different schemas: i.e they don't speak the same language



Problem Statement



GitWorks Overview

GitWorks Platform Overview

What is it?

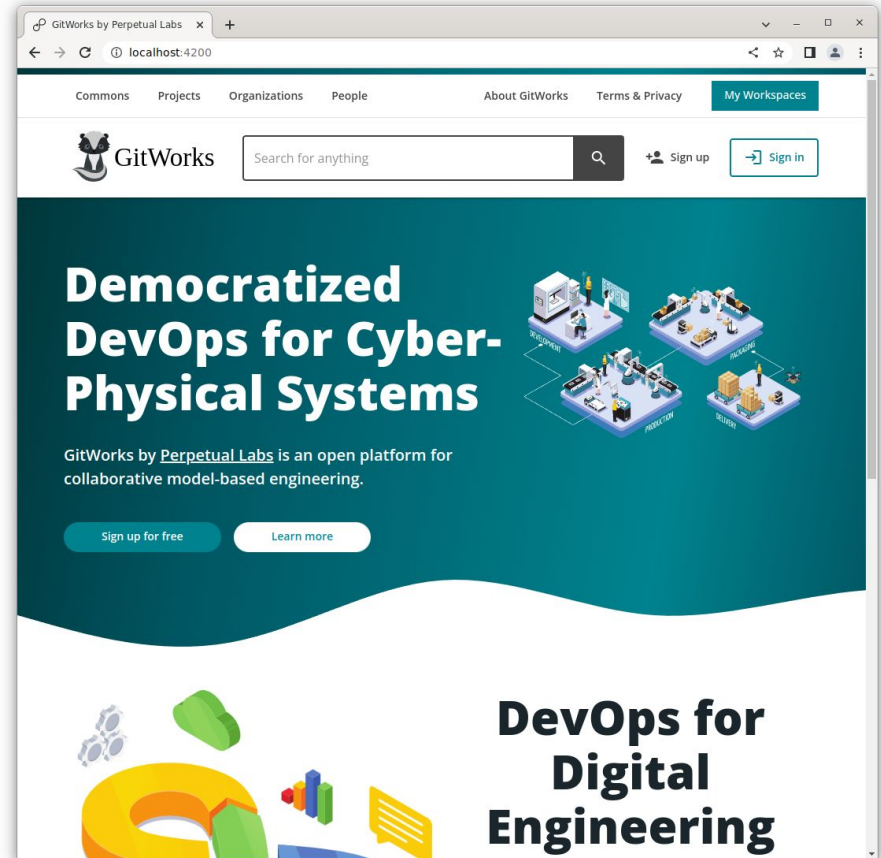
- DevOps for Digital Engineering: Think GitLab for Model-Based design of Cyber-Physical Systems (CPSs)



- Smart documentation environment: think Notion, Coda, Airtable backed by a Knowledge Graph

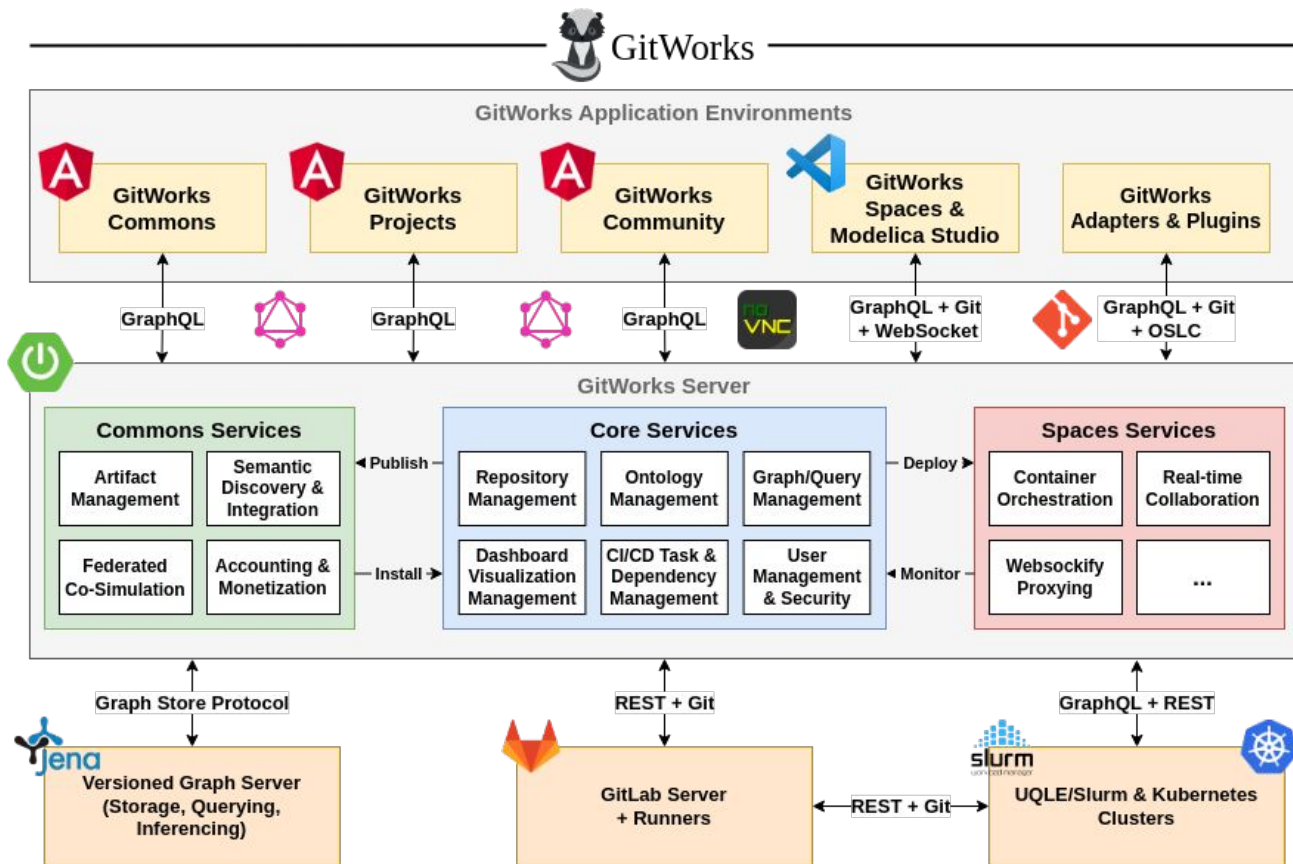
Objectives:

- Provide integrated development platform for Systems Engineering, Designers and Simulation Engineers to manage their digital artifacts, create smart documentation, verify and share their models at speed and with confidence



GitWorks Platform Architecture

- > GitWorks Projects
- > GitWorks Commons
- > GitWorks IDEs



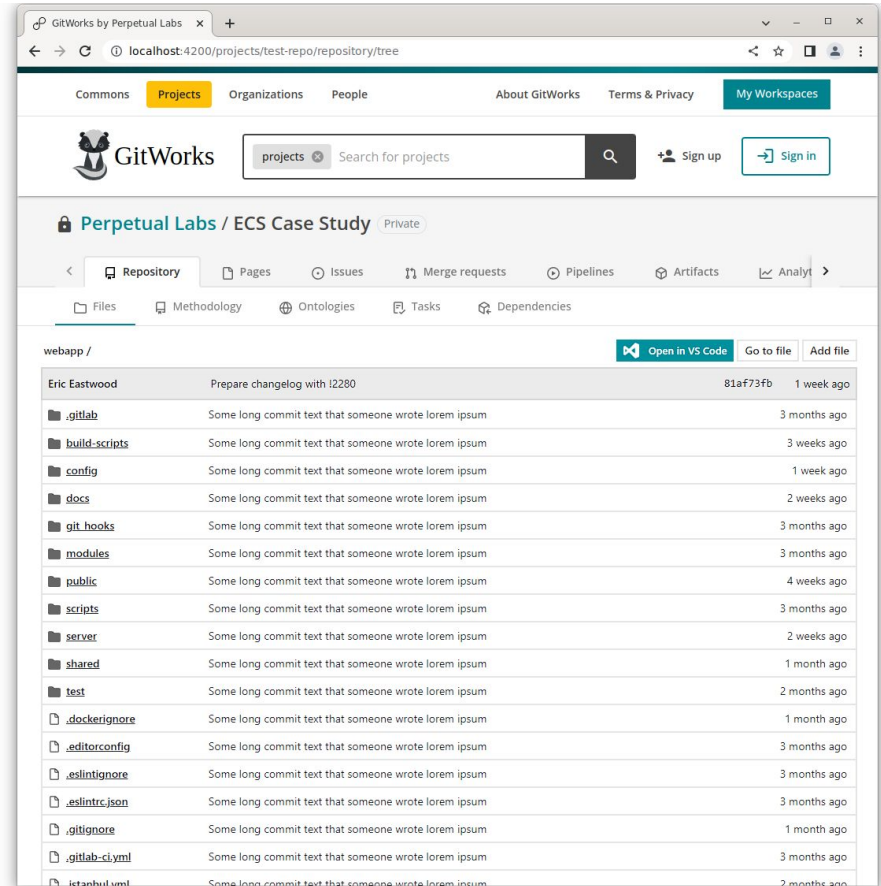
GitWorks Projects Environment

> GitWorks Projects

- **Versioned file management**
 - **On par with essential Git repository management features from GitHub/GitLab**
 - **e.g. branches, Issues, Merge Requests, ...**
- Semantic Twin
- DevOps task management

> GitWorks Commons

> GitWorks IDEs



The screenshot displays the GitWorks web interface. The browser address bar shows the URL `localhost:4200/projects/test-repo/repository/tree`. The navigation bar includes tabs for Commons, Projects (active), Organizations, People, About GitWorks, Terms & Privacy, and My Workspaces. A search bar is present with the text "projects" and a search icon. Below the navigation bar, the repository name "Perpetual Labs / ECS Case Study" is shown with a "Private" label. The main content area displays a file tree for the "webapp" directory. The tree includes folders like ".gitlab", "build-scripts", "config", "docs", "git_hooks", "modules", "public", "scripts", "server", "shared", "test", and files like ".dockerignore", ".editorconfig", ".eslintignore", ".eslintrc.json", ".gitignore", ".gitlab-ci.yml", and "istanbul.yml". Each item has a commit hash and a timestamp.

File/Folder	Commit Hash	Timestamp
Eric Eastwood	Prepare changelog with 12280	81af73fb 1 week ago
.gitlab	Some long commit text that someone wrote lorem ipsum	3 months ago
build-scripts	Some long commit text that someone wrote lorem ipsum	3 weeks ago
config	Some long commit text that someone wrote lorem ipsum	1 week ago
docs	Some long commit text that someone wrote lorem ipsum	2 weeks ago
git_hooks	Some long commit text that someone wrote lorem ipsum	3 months ago
modules	Some long commit text that someone wrote lorem ipsum	3 months ago
public	Some long commit text that someone wrote lorem ipsum	4 weeks ago
scripts	Some long commit text that someone wrote lorem ipsum	3 months ago
server	Some long commit text that someone wrote lorem ipsum	2 weeks ago
shared	Some long commit text that someone wrote lorem ipsum	1 month ago
test	Some long commit text that someone wrote lorem ipsum	2 months ago
.dockerignore	Some long commit text that someone wrote lorem ipsum	1 month ago
.editorconfig	Some long commit text that someone wrote lorem ipsum	3 months ago
.eslintignore	Some long commit text that someone wrote lorem ipsum	3 months ago
.eslintrc.json	Some long commit text that someone wrote lorem ipsum	3 months ago
.gitignore	Some long commit text that someone wrote lorem ipsum	1 month ago
.gitlab-ci.yml	Some long commit text that someone wrote lorem ipsum	3 months ago
istanbul.yml	Some long commit text that someone wrote lorem ipsum	2 months ago

GitWorks Projects Environment

> GitWorks Projects

- Versioned file management
- **Semantic Twin**
 - **Versatile ontology editor based on Semantic Web. e.g. text, form, diagram, & table editors**
 - Knowledge graph exploration and reporting
- DevOps task management

> GitWorks Commons

> GitWorks IDEs

The screenshot displays the GitWorks web interface for a project named 'Perpetual Labs / ECS Case Study'. The interface is organized into several sections:

- Navigation Bar:** Includes 'Commons', 'Projects' (highlighted), 'Organizations', 'People', 'About GitWorks', 'Terms & Privacy', and 'My Workspaces'.
- Search and User Actions:** A search bar with the placeholder 'projects' and a 'Search for projects' button, along with 'Sign up' and 'Sign in' buttons.
- Repository View:** Shows a tree view of the repository structure under the 'Ontologies' tab. The tree includes folders like 'Flow', 'System', 'RobotFlow', 'Robot', 'Topography', 'Central_Unit', 'CaptureSubSystem', 'Wifi', and 'NewEntity'.
- Main Workspace:** Displays a diagram titled 'Topography X' with a zoom level of 100%. The diagram shows a network of components including 'Central_Unit', 'CaptureSubSystem', 'Engine', 'Radar', and 'Radar_Capture', connected by lines representing relationships.
- Details Panel:** Located on the right, it shows 'No object selected'.

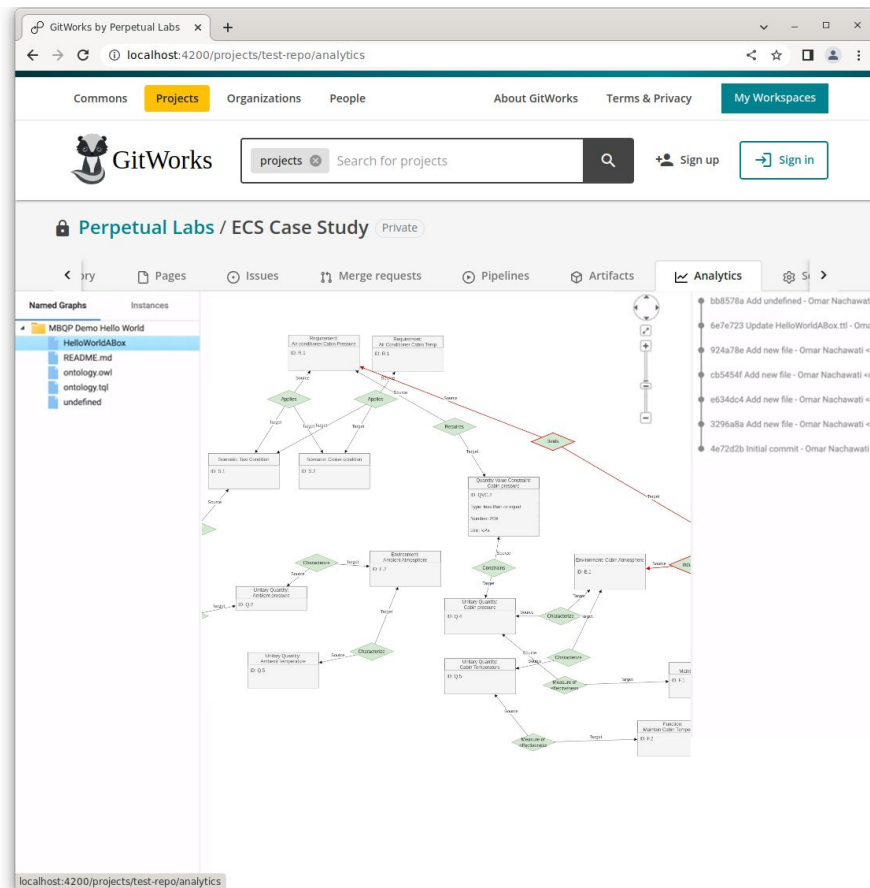
GitWorks Projects Environment

> GitWorks Projects

- Versioned file management
- **Semantic Twin**
 - Versatile ontology editor based on Semantic Web. e.g. text, form, diagram, & table editors
 - **Knowledge graph exploration and reporting**
- DevOps task management

> GitWorks Commons

> GitWorks IDEs



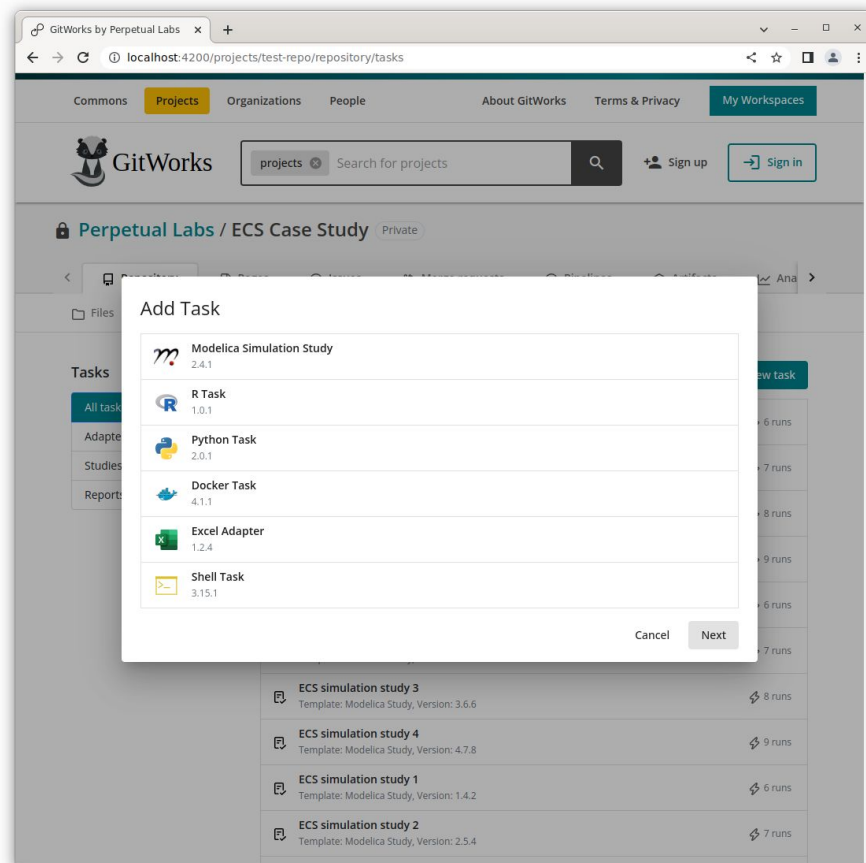
GitWorks Projects Environment

> GitWorks Projects

- Versioned file management
- Semantic Twin
- **DevOps task management**
 - **Adapters for semantic integration of disparate engineering artifacts**
 - **Built-in HPC support for computationally intensive studies (e.g. simulation-based UQ)**

> GitWorks Commons

> GitWorks IDEs



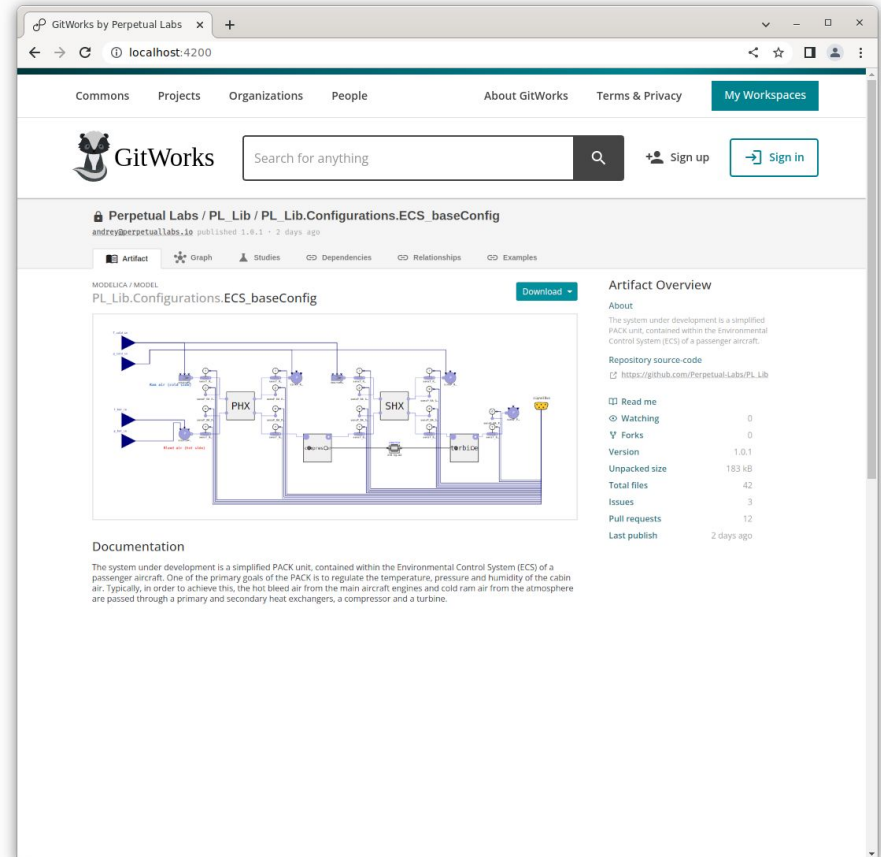
GitWorks Commons Environment

> GitWorks Projects

> GitWorks Commons

- On par with essential package registry features from NPM/Maven Central (e.g. publishing, dependency management, ...)
- Semantic search and integration of published artifacts (RDF/SPARQL)
- IP-protected cosimulation (e.g. FMU as a service)
- Includes monetization strategy, effectively creating marketplace for models, tools and services related to CPS development.

> GitWorks IDEs



The screenshot displays the GitWorks Commons environment interface. The browser address bar shows 'localhost:4200'. The navigation bar includes 'Commons', 'Projects', 'Organizations', 'People', 'About GitWorks', 'Terms & Privacy', and 'My Workspaces'. The main content area shows the project page for 'Perpetual Labs / PL_Lib / PL_Lib.Configurations.ECS_baseConfig', published by 'andrey@perpetuallabs.io' on '1.9.1' two days ago. The page features a search bar, 'Sign up', and 'Sign in' buttons. Below the project title, there are tabs for 'Artifact', 'Graph', 'Studies', 'Dependencies', 'Relationships', and 'Examples'. The main content area displays a diagram titled 'PL_Lib.Configurations.ECS_baseConfig' with a 'Download' button. The diagram shows a complex system with components like PHX, SHX, and FLOW, connected by lines. To the right of the diagram is an 'Artifact Overview' section with 'About', 'Repository source code', and a table of statistics.

Read me	
Watching	0
Forks	0
Version	1.0.1
Unpacked size	183 kB
Total files	42
Issues	3
Pull requests	12
Last publish	2 days ago

Light-weight IDEs via VSCode (for Web)

- Integrated Git version control
- Real-time collaboration (LiveShare)
- First Example: Web-based Modelica editor

Modelica Studio

- Browser-based Modelica Editor
 - Synchronized text and diagram editing
- Open core based on OMFronend.js:
 - Standalone JavaScript library for incremental, error-tolerant parsing & lazy flattening of Modelica models
 - Diagram/Icon SVG generation
 - Also used for implementing the Modelica/OML Semantic Web adapter

Heavy-weight workspaces via VDI over noVNC

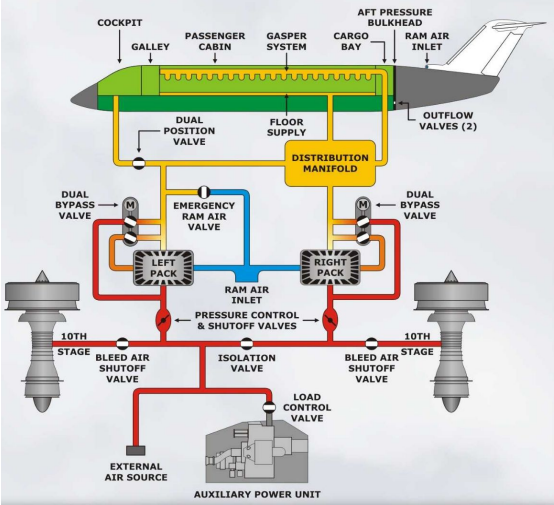
- Delivers containerized desktop applications over the Web

The screenshot displays a web-based IDE interface. On the left, there is a sidebar with 'LIVE SHARE' and 'SESSION DETAILS' sections. The main area is split into two panes. The left pane shows a code editor with Modelica code for a circuit model, including components like Inductor, Resistor, Conductor, and Capacitor. The right pane shows a corresponding circuit diagram with components labeled L, R, G, C, and Gnd. The diagram includes a voltage source, an inductor, a resistor, a conductance, and two capacitors connected in a network.

Case study

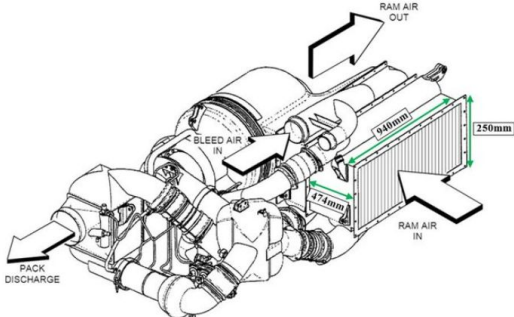
Introduction to the ECS Case Study

Organisation:
ECS-Customer

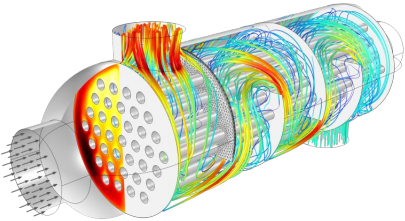


Environmental Control
System (ECS)

Organisation:
ECS supplier



Passenger Air Conditioner

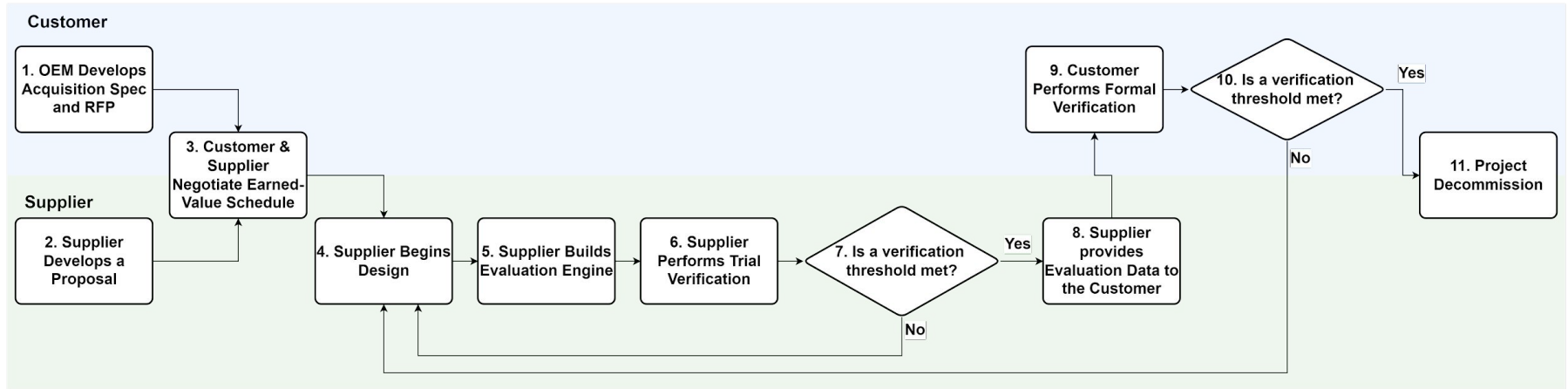


Heat Exchanger

1: figure reproduced from Landis, Albert & Dixon-Hardy, Darron & Heggs, Peter & Al-Damook, Moustafa. (2018). CFD Analysis of RAM Air Flow in an Aircraft Air Conditioning System. 10.13140/RG.2.2.29149.56802.

ECS Case Study Workflow

In this case study GitWorks facilitates the following customer/supplier workflow:



Summarised in three steps:

1. System definition
2. Building the evaluation engine (simulation model)
3. Evaluation and verification

ECS Case Study - Step 1 - System Definition



ontological
modeling
language

- Customer defines a vocabulary and descriptions (Acquisition Spec) that capture:
 - Operational and functional constraints
 - Characteristic quantities
 - Requirements
 - System boundaries
- Elaborates the system design by defining:
 - Proposed constituent components
 - Proposed connections
 - Proposed configurations, controls, inputs, etc.

Snippet of the Customer's ECS Description in OML

```
1 @rdfs:label "air conditioner"
2 ci airConditioner : ecs-cust-vocab:AirConditioner [
3   base:hasIdentifier "C.1"
4   base:hasCanonicalName "Air Conditioner"
5   base:hasDescription "The Air Conditioner conditions the cabin atmosphere."
6 ]
7
8 @rdfs:label "air conditioner mass"
9 ci acMass : vim4:IndividualUnitaryQuantity [
10  base:hasIdentifier "Q.1"
11  base:hasCanonicalName "air conditioner mass"
12  base:hasDescription "The mass of the air conditioner."
13  vim4:instantiates iso-80000-4.1:mass
14  vim4:characterizes airConditioner
15 ]
16
17 @rdfs:label "air conditioner induces cabin atmosphere"
18 ri acInducesCabinAtmosphere : mission:Induces [
19   from airConditioner
20   to cabinAtmosphere
21 ]
22
23 @rdfs:label "ambient atmosphere influences air conditioner"
24 ri acInhabitsCabinAtmosphere : mission:Influences [
25   from ambientAtmosphere
26   to airConditioner
27 ]
```

ECS Case Study - Step 1 - System Definition



- Customer defines a vocabulary and descriptions (Acquisition Spec) that capture:
 - Operational and functional constraints
 - Characteristic quantities
 - Requirements
 - System boundaries
- Elaborates the system design by defining:
 - Proposed constituent components
 - Proposed connections
 - Proposed configurations, controls, inputs, etc.

Snippet of the ECS Supplier Description in OML

```
1 ci compressor : ecs-supp-vocab:Compressor [
2   base:hasIdentifier "C.1.1"
3   base:hasCanonicalName "Compressor"
4   base:hasDescription "A compressor."
5   mission:presents compressorIntermediateAirInlet
6   mission:presents compressorConditionedAirOutlet
7 ]
8
9 ci heatExchanger : ecs-supp-vocab:HeatExchanger [
10  base:hasIdentifier "C.1.2"
11  base:hasCanonicalName "Heat Exchanger"
12  base:hasDescription "A heat exchanger."
13  mission:presents heatExchangerBleedAirInlet
14  mission:presents heatExchangerOusideAirInlet
15  mission:presents heatExchangerRejectedAirOutlet
16  mission:presents heatExchangerIntermediateAirOutlet
17 ]
18
19 ref ci ecs-cust-desc:airConditioner [
20   base:contains compressor
21   base:contains heatExchanger
22 ]
23 ...
24 ci bleedAirPressure : vim4:IndividualUnitaryQuantity [
25   base:hasIdentifier "Q.6"
26   base:hasCanonicalName "bleed air pressure at heat exchanger inlet"
27   base:hasDescription "The pressure of the bleed air at the inlet of the heat exchanger."
28   vim4:instantiates iso-80000-4.15:pressure
29   vim4:characterizes bleedAir
30 ]
31
32 ci bleedAirTemperature : vim4:IndividualUnitaryQuantity [
33   base:hasIdentifier "Q.7"
34   base:hasCanonicalName "bleed air temperature at heat exchanger inlet"
35   base:hasDescription "The temperature of the bleed air at the inlet of the heat exchanger."
36   vim4:instantiates iso-80000-5.1:thermodynamic-temperature
37   vim4:characterizes bleedAir
38 ]
```

ECS Case Study - Step 1 - System Definition

The screenshot displays the GitWorks interface for an ECS Demo repository. The main content area shows the OML Form editor with the following code:

```
1 @dc:title "Example Vocabulary"
2 @dc:creator "Example Company"
3 @dc:rights "Copyright 2019, by Example Company"
4 vocabulary <http://perpetuallabs.io/ecs-demo/ecs-customer/vocabulary/ecs-customer/
5
6 extends <http://www.w3.org/2000/01/rdf-schema#> as rdfs
7 extends <http://purl.org/dc/elements/1.1/> as dc
8 extends <http://imce.jpl.nasa.gov/foundation/mission#> as mission
9
10 // The ECS deals with properties of atmospheres.
11
12 @rdfs:label "Atmosphere Environment"
13 @dc:description "An ecs:AtmosphereEnvironment is a mission:Environment"
14 concept AtmosphereEnvironment :-> mission:Environment
15
16 // The deliverable is an AirConditioner
17
18 @rdfs:label "Air Conditioner"
19 @dc:description "An ecs:AirConditioner is a mission:Component demand"
20 concept AirConditioner :-> mission:Component
```

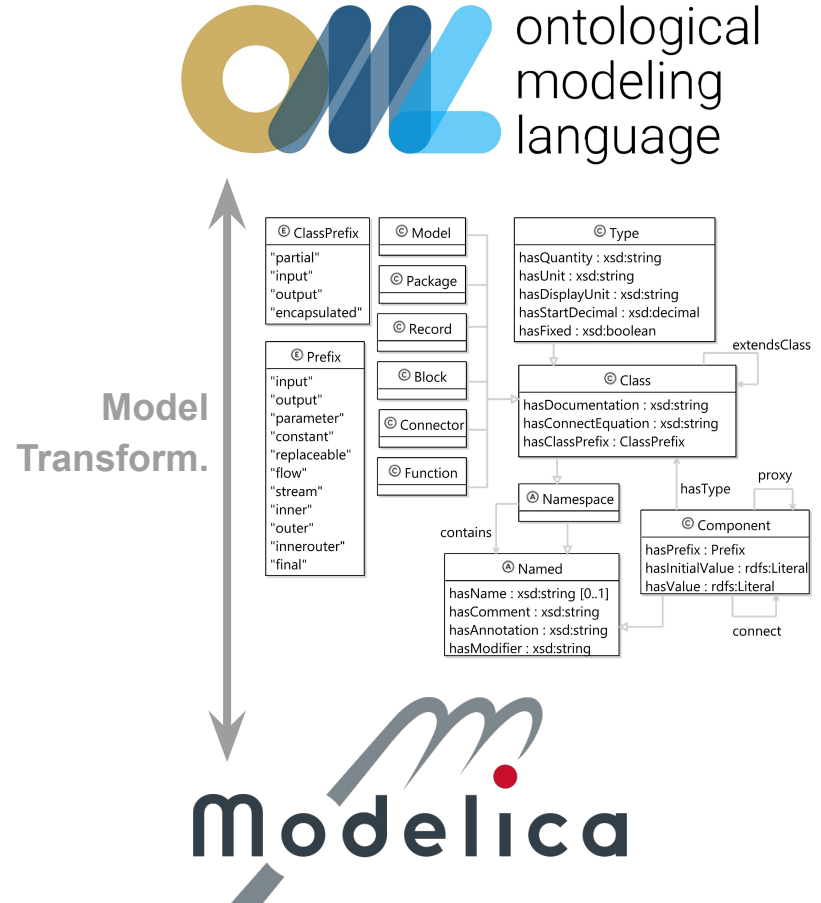
The diagram on the right visualizes the relationships between various entities. A legend at the bottom right identifies the entities: Component, Environment, Mission, AtmosphereEnvironment, AirConditioner, Atmosphere, Message, Product, Objective, Specifics, Transfer, TransferIn, TransferOut, and Flow. The diagram shows a complex network of relationships between these entities, with arrows indicating the direction of the relationships.

ECS Case Study - Step 2 - Building the Evaluation Engine



Semantic-Twin powered model editing is enabled by the corresponding model transformation algorithm:

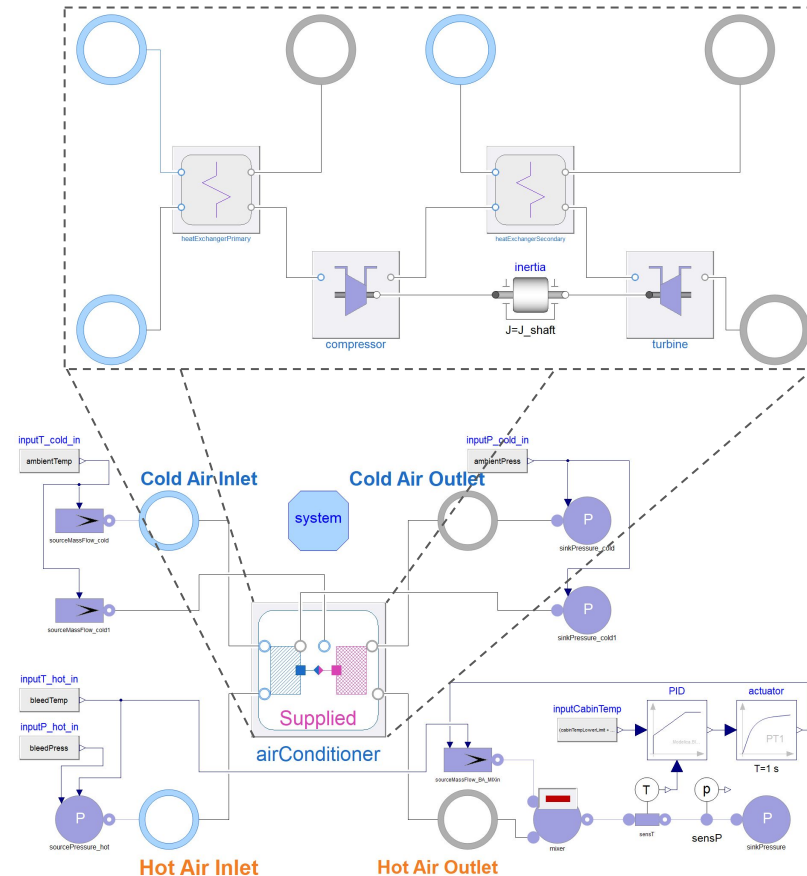
- For each component, selects an appropriate Modelica model based on the type of the component, the number and types of its interfaces, exchanged flows, etc.
- For each quantity involved in verification ensures that that quantity is computed and exposed on output



ECS Case Study - Step 2 - Building the Evaluation Engine



- Simulation Engineers can use any IDE to complete the model including ModelicaStudio with LiveShare
- Design variants can be independently developed on different branches of the repository and automation pipelines can be set-up independently.
- All model revisions are version controlled and dependencies managed appropriately



ECS Case Study - Step 3 - Evaluation & Verification



- A completed simulation model is used to generate the evaluation data and compile verification reports through an automation pipeline (CI/CD)
- Supplier can publish the Modelica model itself or provide the evaluation data
- Verification results are compared to a milestone criteria
- Design iterations continue until all milestones are completed

Component	Requirement	Scenario	Quantity	Constraint	Unit	Number of Observations		
C.1	R.1	✓ always	✓ Q.1	✓ ≤ 1.0000e+02	kilogram	0	1	✓
		✓ S.1	✓ Q.4	✓ ≥ 9.8000e+04	pascal	0	264	✓
	S.2	✓	✓ Q.4	✓ ≥ 9.8000e+04	pascal	0	264	✓
			✓ Q.4	✓ ≥ 9.8000e+04	pascal	0	260	✓
		✗	✗ Q.5	✗ ≥ 2.9200e+02	kelvin	2	262	✗
			✗ Q.5	✗ ≥ 2.9400e+02	kelvin	16	248	✗
	R.3	✗ S.1	✗ Q.5	✗ ≥ 2.9200e+02	kelvin	2	262	✗
			✗ Q.5	✗ ≥ 2.9400e+02	kelvin	16	248	✗
✓ S.2		✓ Q.18	✓ ≤ 1.8000e+02	second	0	264	✓	
		✓ Q.18	✓ ≤ 1.8000e+02	second	0	260	✓	

Conclusions

Conclusions and Future work

- We have presented our vision for the GitWorks platform to enable the collaborative model-based design of cyber-physical systems.
- GitWorks is designed from ground up to address many MBSE challenges by employing a novel combination of DevOps and Semantic Web Technologies.
- The demonstrated case study showcased how GitWorks can be used in practice by different collaborators along the supply chain.

- Plans for future work include:
 - Further development of adaptors and IDEs to increase the number of different modeling paradigms and COTS tools supported by the platform
 - Increase the maturity of the user interface for the web applications
 - Demonstrate the application to other use cases including satellite systems design and manufacturing.

Thank you for your attention!

Any questions?