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

Omar Nachawati PhD (co-founder and CTO)

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

Steven Jenkins PhD (Advisor)

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

Maged Elaasar PhD (Advisor)

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

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

Hans Peter de Koning (advisor)

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

Perpetual Labs Team

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

Problem Statement

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 <u>three</u> fundamental technical challenges

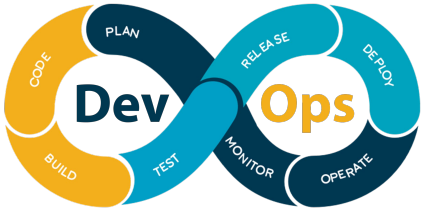
- **Version Control, Change Management and traceability**
  - Without an overarching strategy, a system description becomes a disorganized and untrustworthy collection of information artifacts.

Problem Statement

There are <u>three</u> 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

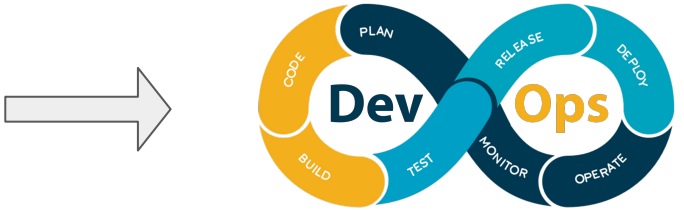


Problem Statement

There are <u>three</u> 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

git

Dev Ops

PLAN CODE BUILD TEST RELEASE DEPLOY MONITOR OPERATE

Semantic Web

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:

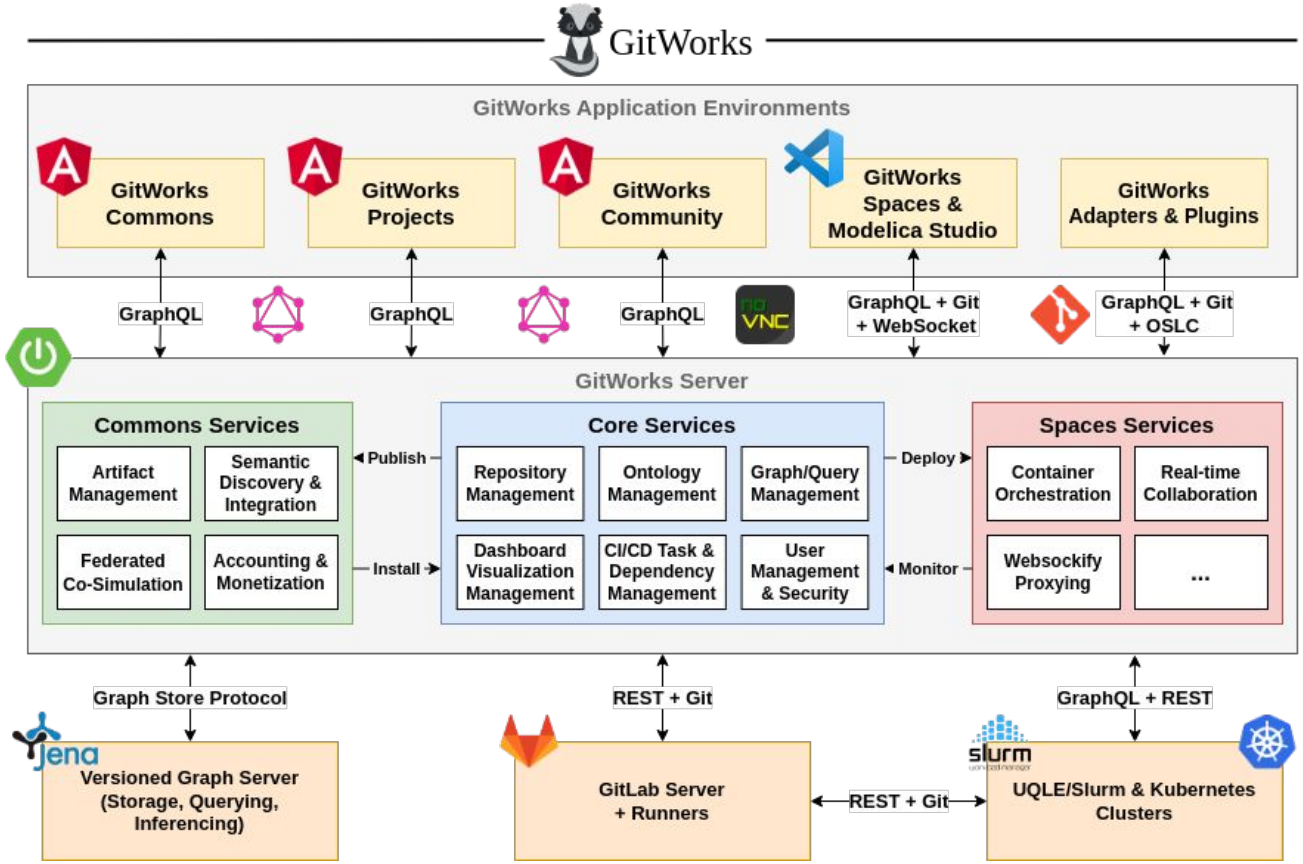
- Provide integrated development platform for Systems Engineering, Designers and Simulation Engineers to manage their digital artifacts, create smart documentation, verify and share their models at speed and with confidence

GitWorks Platform Architecture

> GitWorks Projects

> GitWorks Commons

> GitWorks IDEs



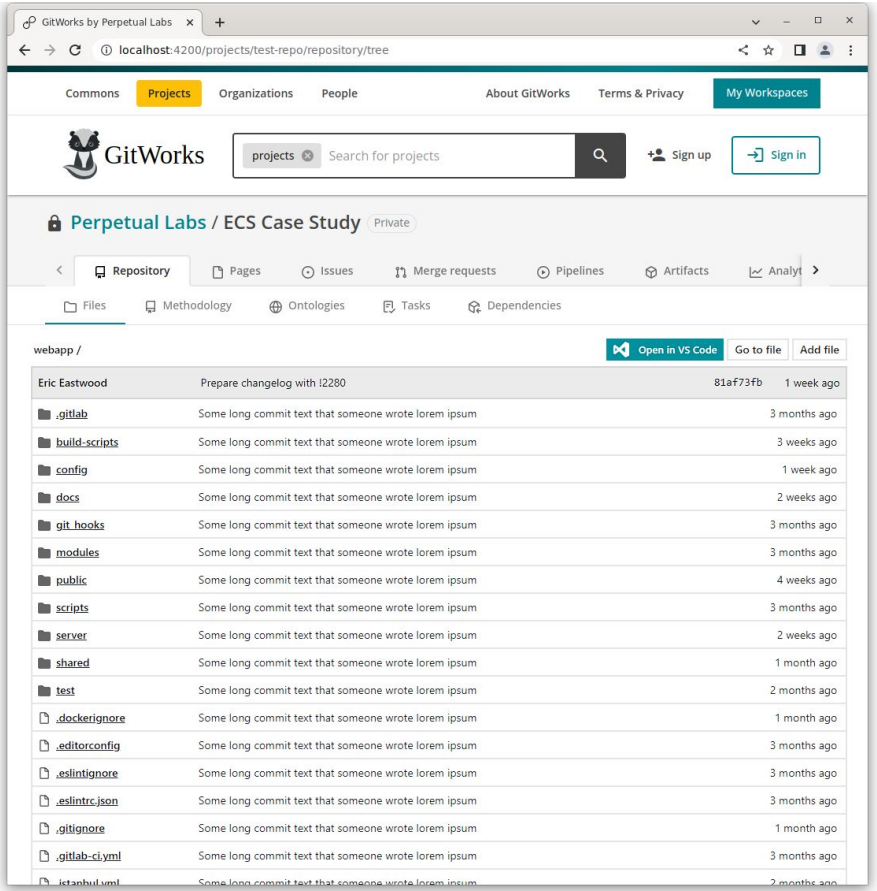
GitWorks Projects Environment

> 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



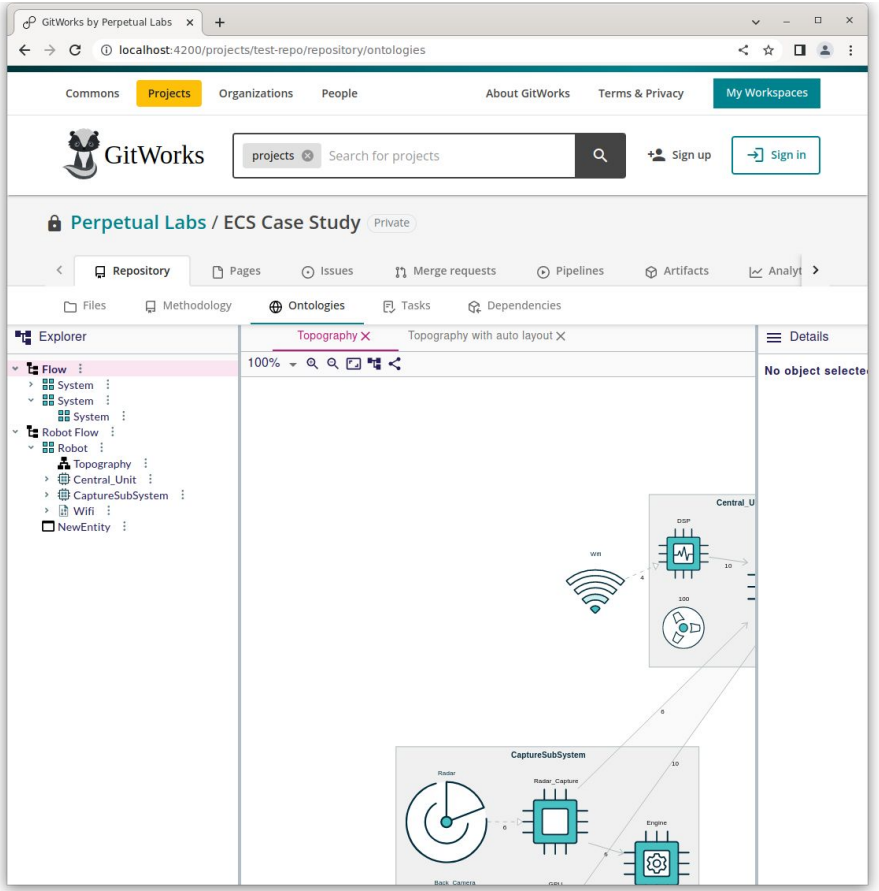
GitWorks Projects Environment

> 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



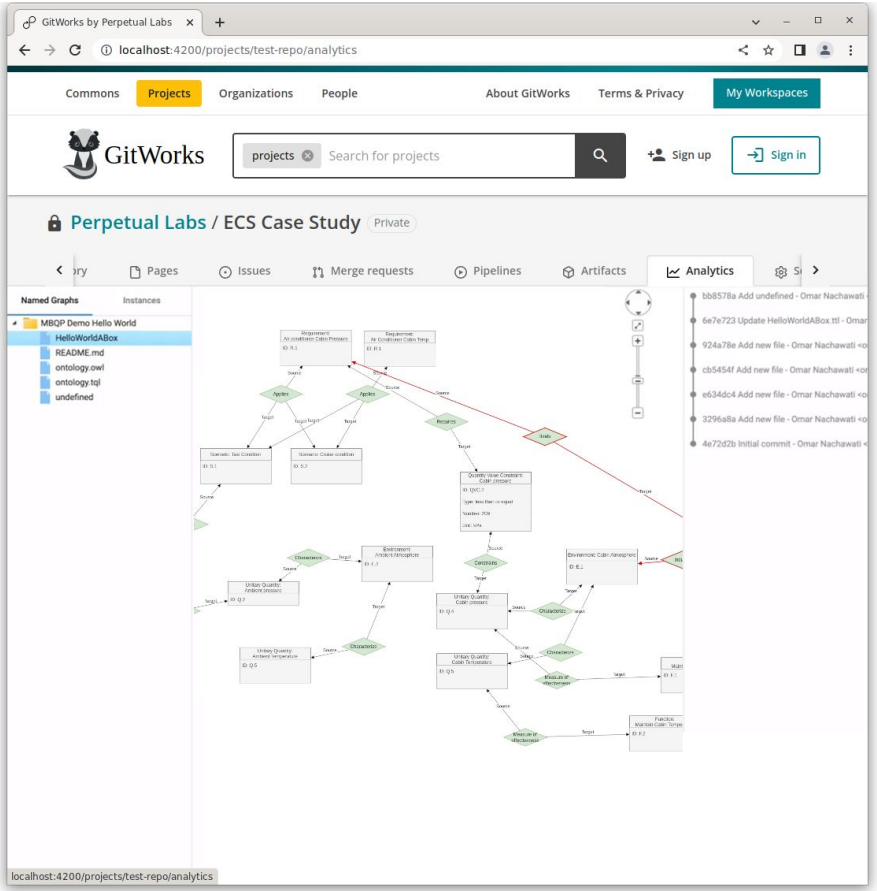
GitWorks Projects Environment

> GitWorks Projects

- Versioned file management
- **Semantic Twin**
  - 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



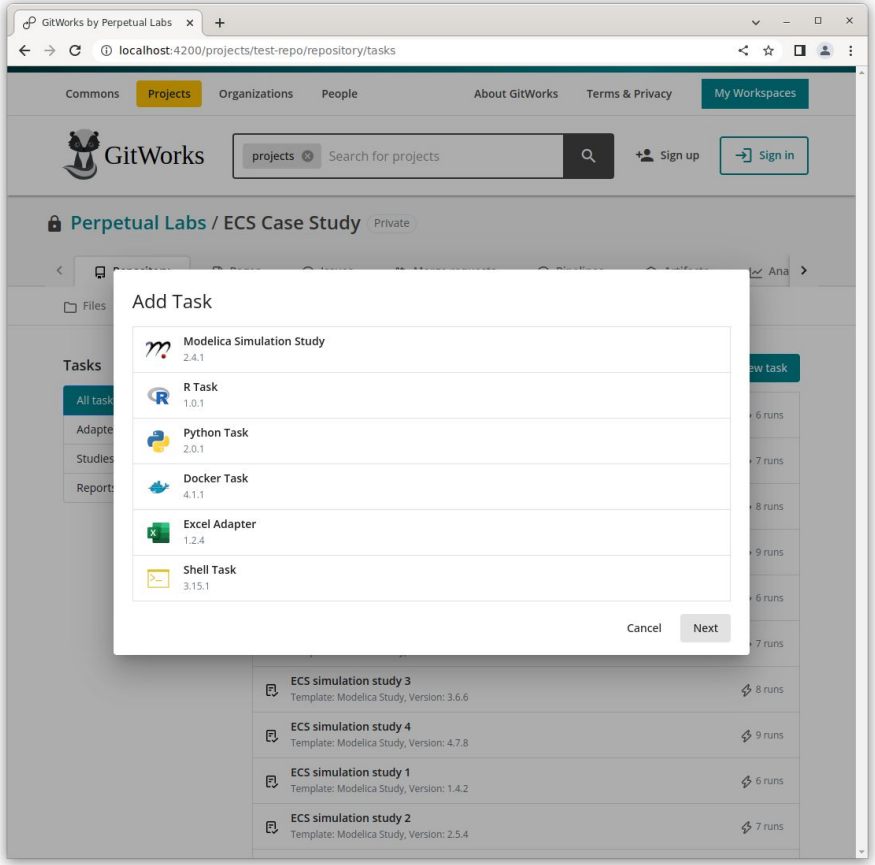
GitWorks Projects Environment

> 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

GitWorks IDEs

Light-weight IDEs via VSCode (for Web)

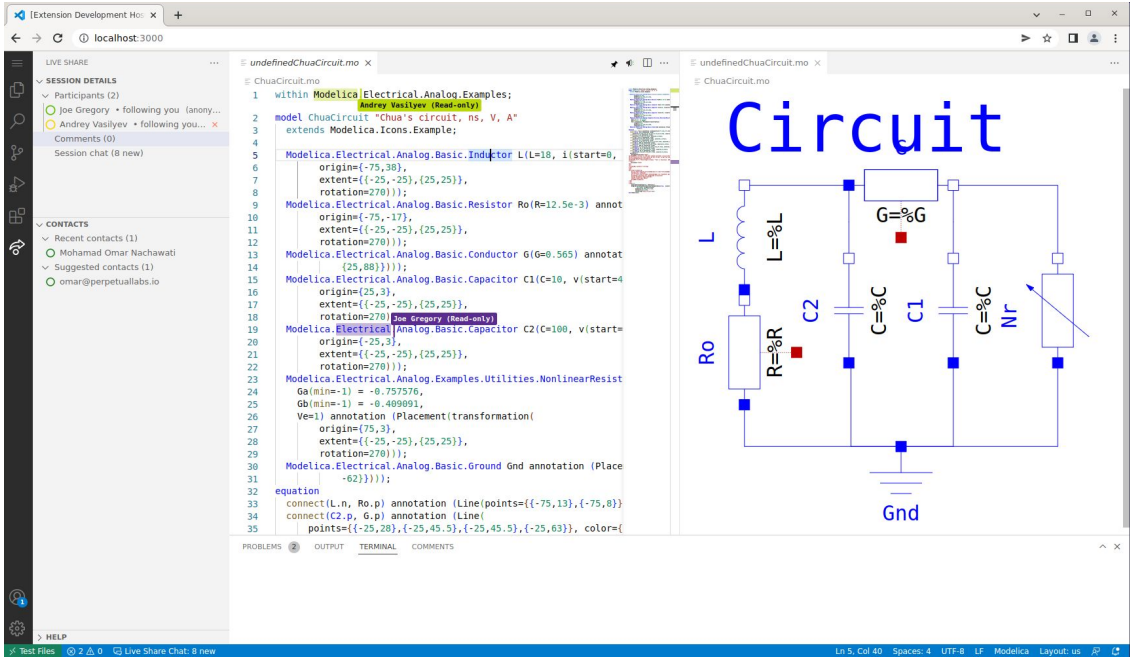
- Integrated Git version control
- Real-time collaboration (LiveShare)
- First Example: Web-based Modelica editor

Modelica Studio

- Browser-based Modelica Editor
  - Synchronized text and diagram editing
- Open core based on OMFrontend.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

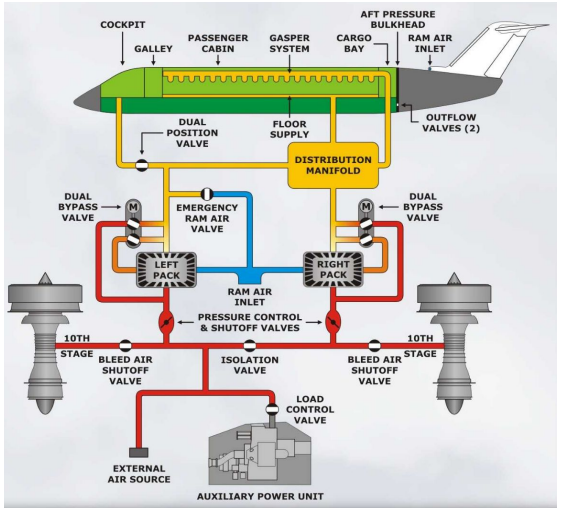
- Delivers containerized desktop applications over the Web



Case study

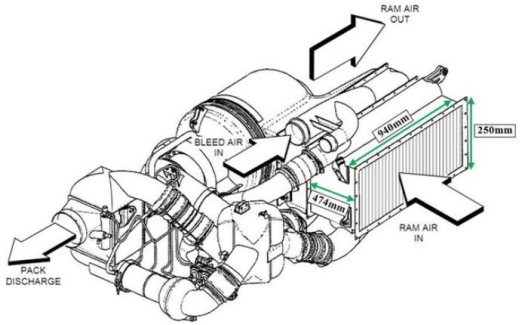
Introduction to the ECS Case Study

Organisation:
ECS-Customer

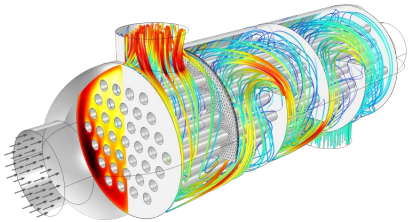


Environmental Control System (ECS)

Organisation:
ECS supplier



Passenger Air Conditioner



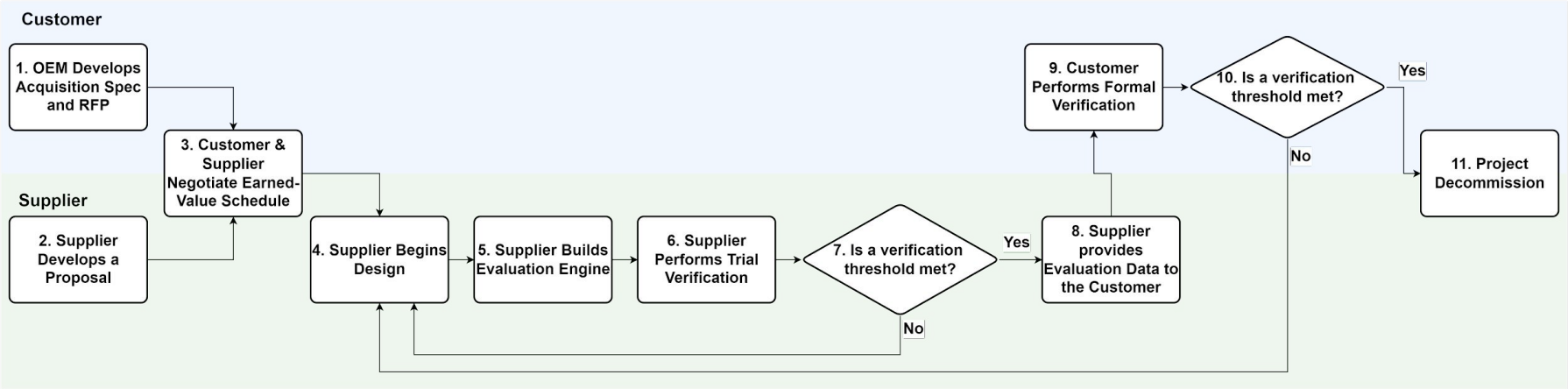
Heat Exchanger

1

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

- 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

```
@rdfs:label "air conditioner"
ci airConditioner : ecs-cust-vocab:AirConditioner [
    base:hasIdentifier "C.1"
    base:hasCanonicalName "Air Conditioner"
    base:hasDescription "The Air Conditioner conditions the cabin atmosphere."
]

@rdfs:label "air conditioner mass"
ci acMass : vim4:IndividualUnitaryQuantity [
    base:hasIdentifier "Q.1"
    base:hasCanonicalName "air conditioner mass"
    base:hasDescription "The mass of the air conditioner."
    vim4:instantiates iso-80000-4.1:mass
    vim4:characterizes airConditioner
]

@rdfs:label "air conditioner induces cabin atmosphere"
ri acInducesCabinAtmosphere : mission:Induces [
    from airConditioner
    to cabinAtmosphere
]

@rdfs:label "ambient atmosphere influences air conditioner"
ri acInhabitsCabinAtmosphere : mission:Influences [
    from ambientAtmosphere
    to airConditioner
]
```

ECS Case Study - Step 1 - System Definition

Semantic Web + 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 ECS Supplier Description in OML

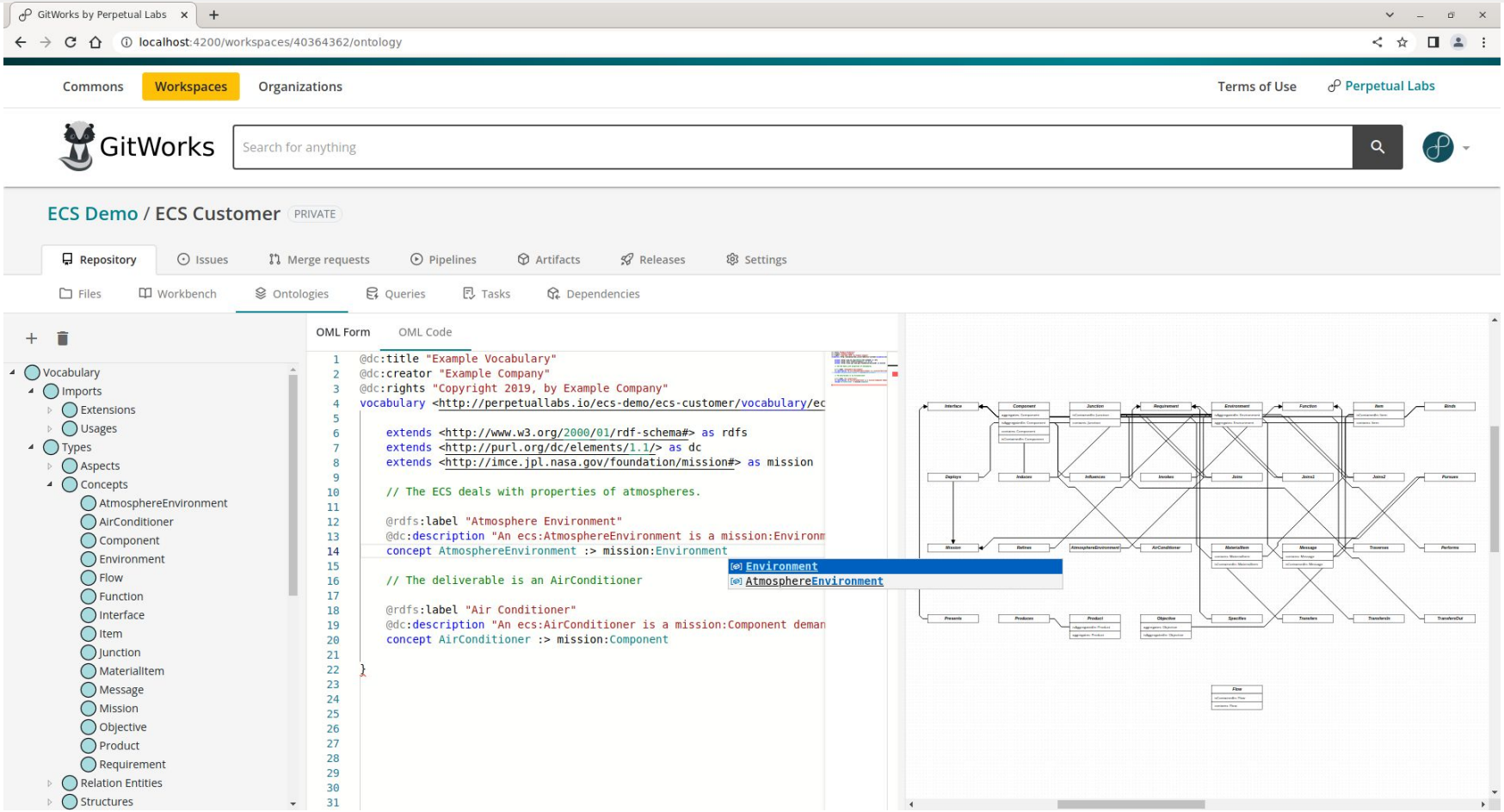
```
ci compressor : ecs-supp-vocab:Compressor [
    base:hasIdentifier "C.1.1"
    base:hasCanonicalName "Compressor"
    base:hasDescription "A compressor."
    mission:presents compressorIntermediateAirInlet
    mission:presents compressorConditionedAirOutlet
]

ci heatExchanger : ecs-supp-vocab:HeatExchanger [
    base:hasIdentifier "C.1.2"
    base:hasCanonicalName "Heat Exchanger"
    base:hasDescription "A heat exchanger."
    mission:presents heatExchangerBleedAirInlet
    mission:presents heatExchangerOusideAirInlet
    mission:presents heatExchangerRejectedAirOutlet
    mission:presents heatExchangerIntermediateAirOutlet
]

ref ci ecs-cust-desc:airConditioner [
    base:contains compressor
    base:contains heatExchanger
]
...
ci bleedAirPressure : vim4:IndividualUnitaryQuantity [
    base:hasIdentifier "Q.6"
    base:hasCanonicalName "bleed air pressure at heat exchanger inlet"
    base:hasDescription "The pressure of the bleed air at the inlet of the heat exchanger."
    vim4:instantiates iso-80000-4.15:pressure
    vim4:characterizes bleedAir
]

ci bleedAirTemperature : vim4:IndividualUnitaryQuantity [
    base:hasIdentifier "Q.7"
    base:hasCanonicalName "bleed air temperature at heat exchanger inlet"
    base:hasDescription "The temperature of the bleed air at the inlet of the heat exchanger."
    vim4:instantiates iso-80000-5.1:thermodynamic-temperature
    vim4:characterizes bleedAir
]
```

ECS Case Study - Step 1 - System Definition



ECS Case Study - Step 2 - Building the Evaluation Engine

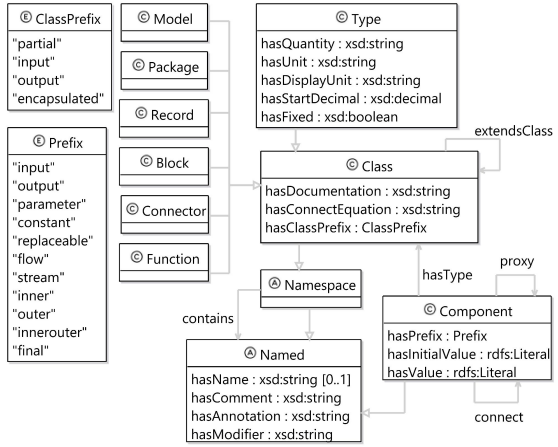
Semantic Web + DevOps

ontological modeling language

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

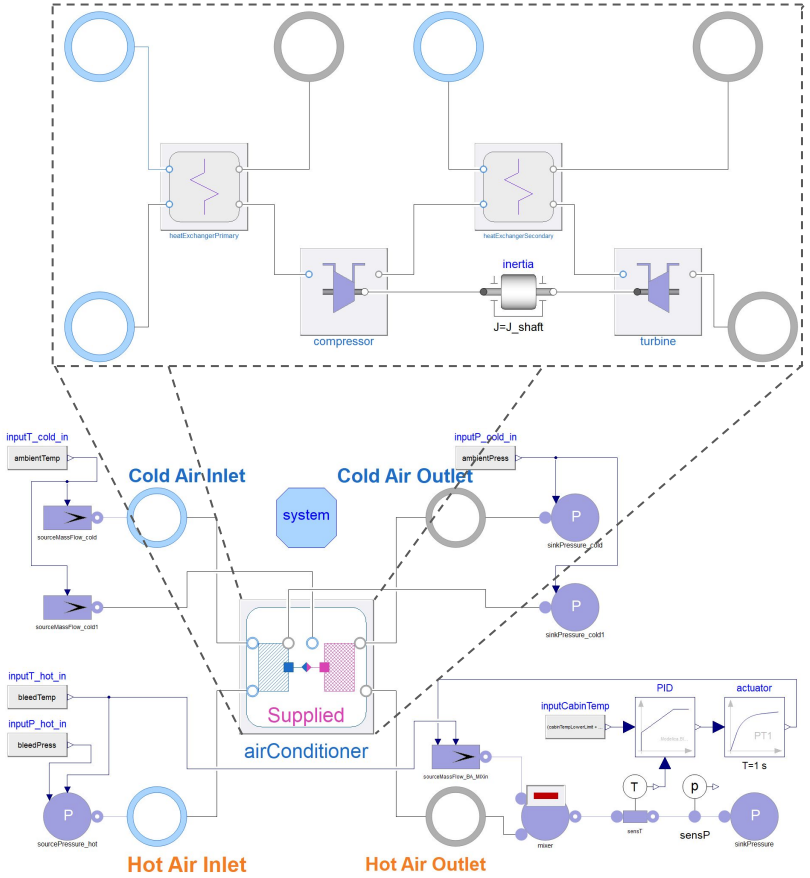
Model Transform.



Modelica

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	✓
		R.2	✓	S.1	✓	Q.4	✓	≥	9.8000e+04	pascal	0	264	✓
								≤	1.0400e+05	pascal	0	264	✓
				S.2	✓	Q.4	✓	≥	9.8000e+04	pascal	0	260	✓
								≤	1.0400e+05	pascal	0	260	✓
		R.3	✗	S.1	✗	Q.5	✗	≥	2.9200e+02	kelvin	2	262	✗
								≤	2.9400e+02	kelvin	16	248	✗
				S.2	✗	Q.5	✗	≥	2.9200e+02	kelvin	0	260	✓
								≤	2.9400e+02	kelvin	5	255	✗
		R.4	✓	S.1	✓	Q.18	✓	≤	1.8000e+02	second	0	264	✓
				S.2	✓	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.

Future work

- 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?