Towards an Open Platform for collaborative Model-Based Design of Cyber-Physical Systems

Presented by

26-10-2022 (Dallas, TX)

founders@perpetuallabs.io

Agenda:

- Introductions
- Motivations
- GitWorks
- ModelicaStudio
- Conclusions

Who are we?
&
Why are we here?

PL 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

PL 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, University of Sheffield

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

Wider context

Partnerships

R&D collaborations

Commercial Traction

INTO-CPS

openCAESAR

esa

BOEING

AARHUS UNIVERSITY

AUTODESK.

BAE SYSTEMS

LINKÖPING UNIVERSITY

GKN AEROSPACE

AIRBUS

BOEING HORIZONX

Hartree Centre
Science & Technology Facilities Council

Why are we here?

Mission:

Make modelling collaborative and accessible to as many people as possible

GitWorks Community

Motivation

Problem Statement

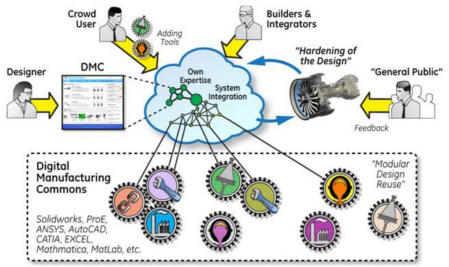
Challenges with Development of complex Engineering Systems

- 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

Many initiatives to Tackle these challenges

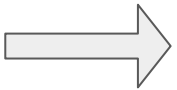
- **Digital Manufacturing Commons** (Beckmann 2016)
  - “The DMC is a key to digital manufacturing innovation because it allows for integration of data across the digital thread. The DMC facilitates system engineering by linking digital information and data across the entire product life cycle and supply chain. Within the DMC, experts and non-traditional contributors can design components or systems by reusing, remixing, and augmenting designs prepared by others. These designs can continue to evolve through a series of reiterative design loops.” (Beckmann 2016)
- **HUBCAP/INTO-CPS** (Larsen et al., 2016).” (Larsen 2020)
  - “In this position paper, we report on an approach that aims to make MBD more accessible to a range of businesses, but especially SMEs, involved in the development of cyber-physical products and systems. … Our goal is to provide a collaboration platform that enables users to access advanced CPS design and engineering solutions, including models and tools”
- **Many more**…



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.

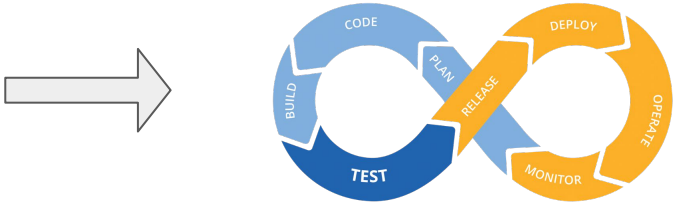


Problem Statement

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

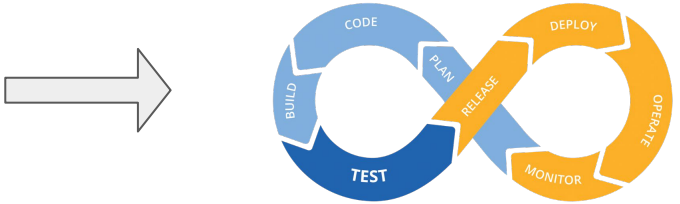


Problem Statement

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.

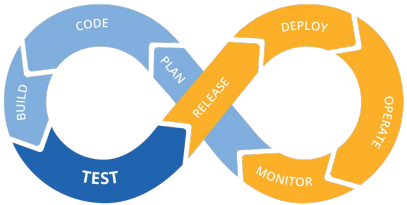
- **Repeatability, durability and efficiency**
  - Without this, it is impossible to perform audits and repeat analyses. This is important to maintain confidence in the design and analysis over time and potentially reuse it



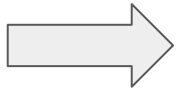
- **Interoperability**
  - System knowledge and data is scattered in many different artefacts (models, documents, emails) and often out of sink.
  - Artefacts have different syntax and tools have different schemas: i.e they don't speak the same language

Problem Statement

git



Semantic Web



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, Airtables 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 Overview

> GitWorks Projects

> GitWorks Commons

> GitWorks IDEs

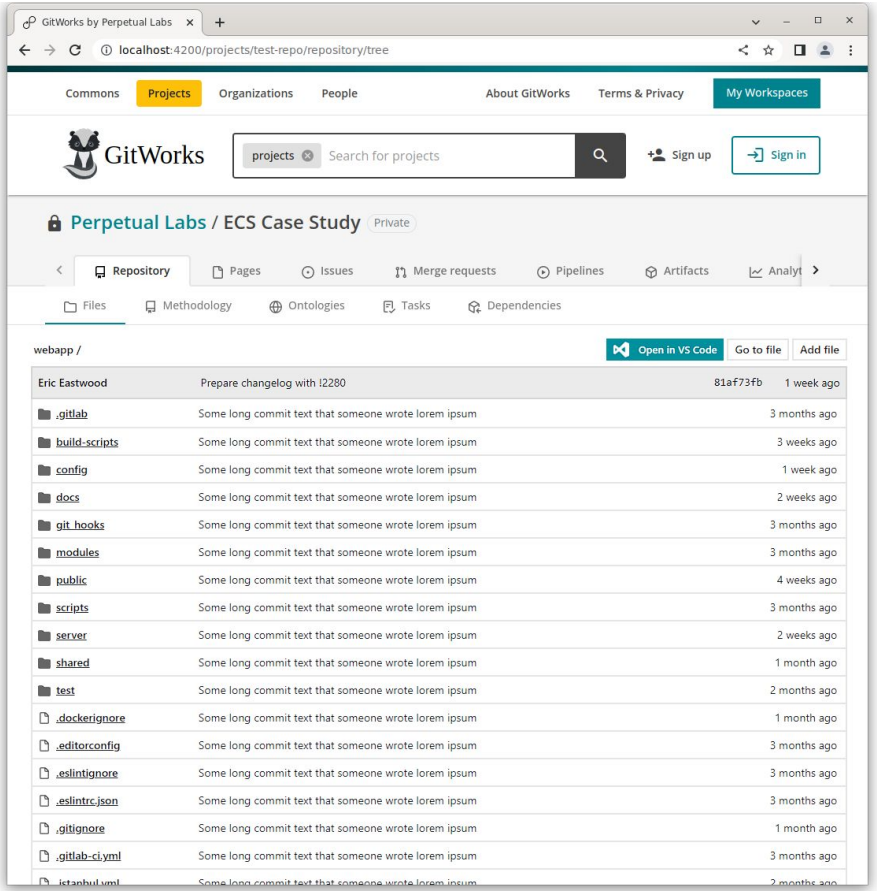
GitWorks Platform Overview

> GitWorks Projects

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

> GitWorks Commons

> GitWorks IDEs



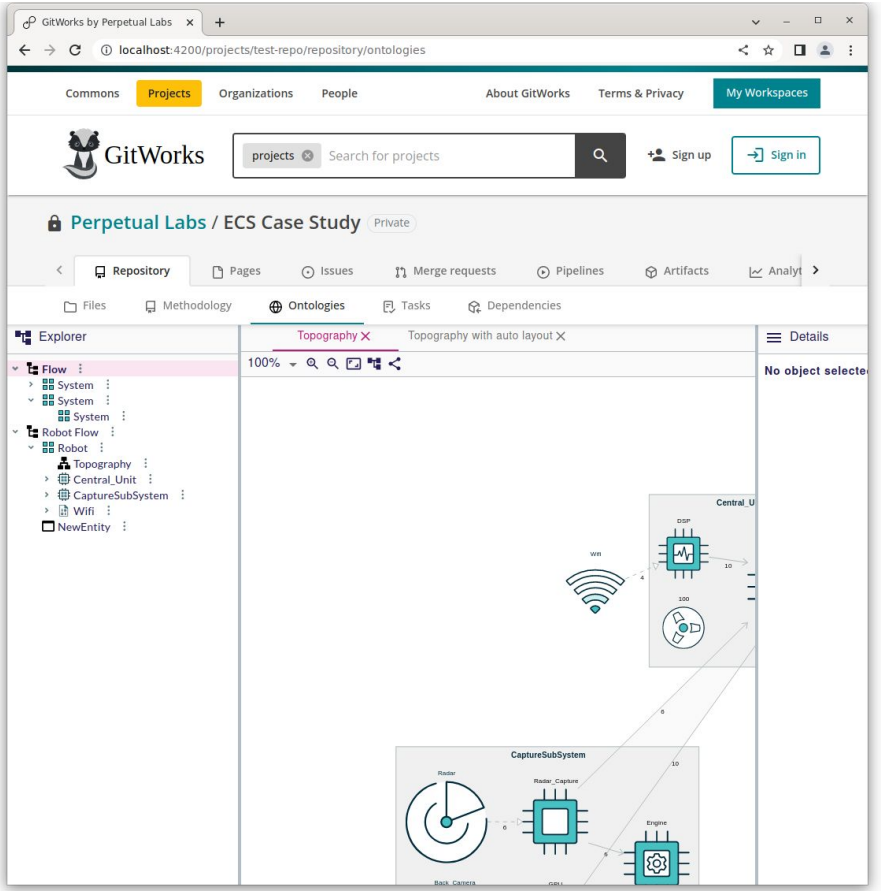
GitWorks Platform Overview

> 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
- CI/CD task management

> GitWorks Commons

> GitWorks IDEs



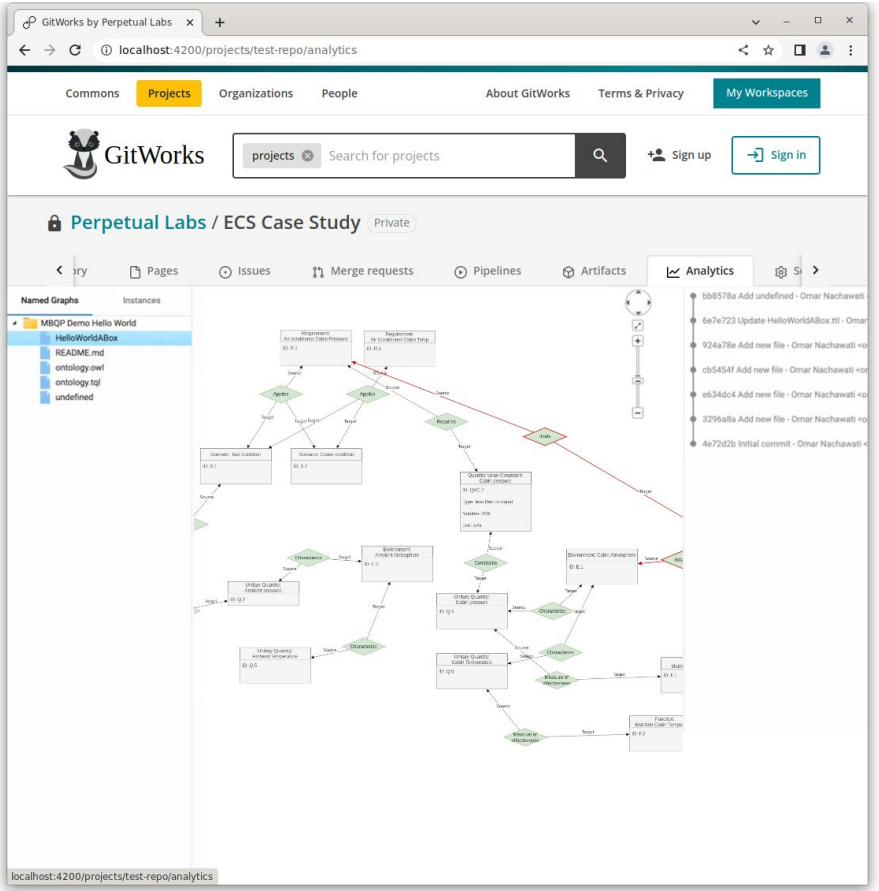
GitWorks Platform Overview

> 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**
- CI/CD task management

> GitWorks Commons

> GitWorks IDEs



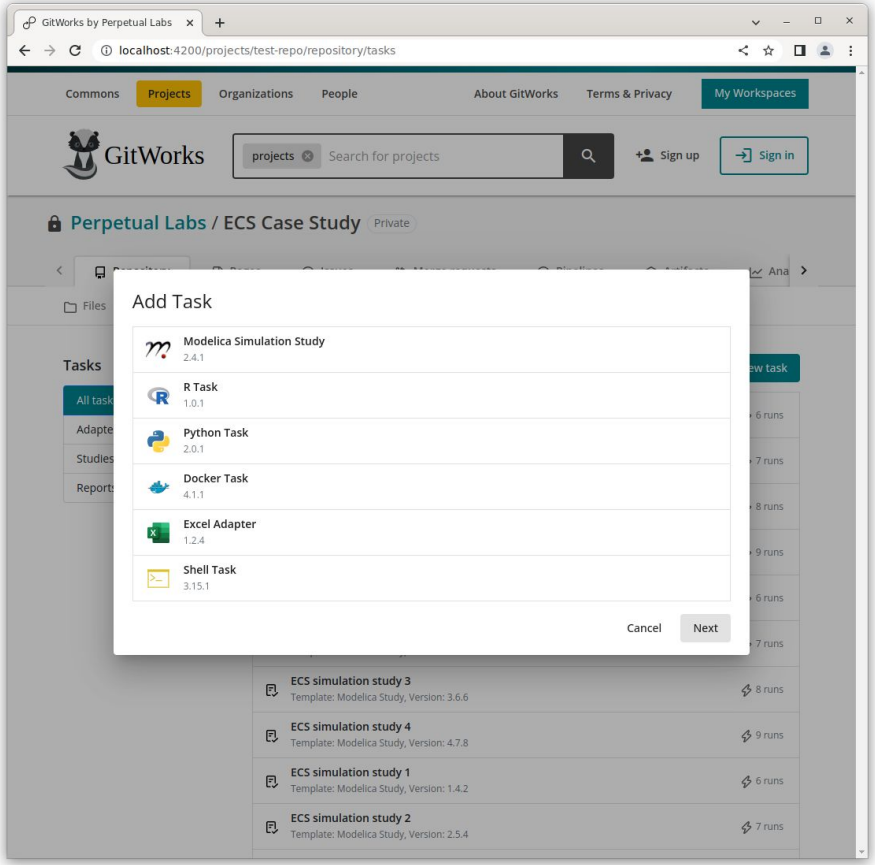
GitWorks Platform Overview

> GitWorks Projects

- Versioned file management
- Semantic Twin
- **CI/CD 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 Platform Overview

> GitWorks Projects

> GitWorks Commons

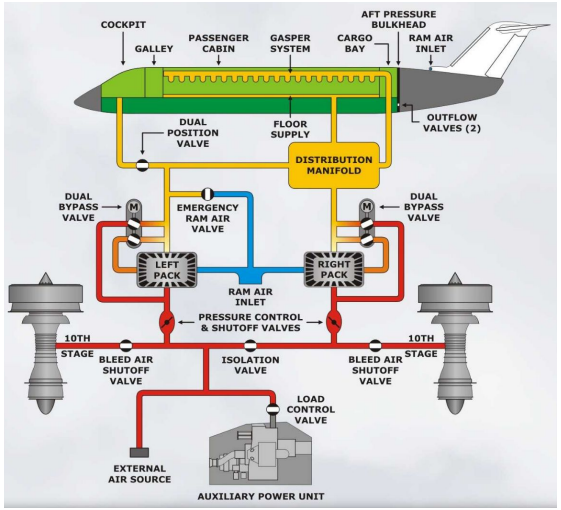
- **On par with essential package registry features from NPM/Maven Central (e.g. publishing, dependency management, …)**
- **Semantic search and integration of published artifacts**
- **IP-protected cosimulation (e.g. FMU as a service)**
- **Includes monetization strategy, effectively creating marketplace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

> GitWorks IDEs

Case study

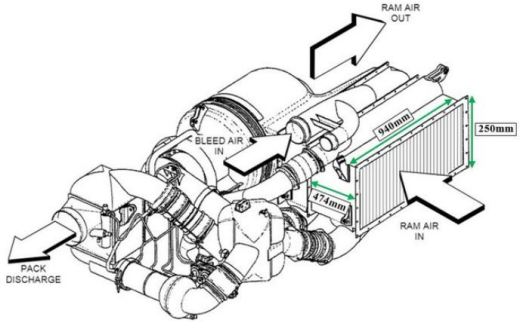
Introduction to ECS Case Study

Organisation 1 (OEM):
ECS-Customer



Environmental Control
System (ECS)

Organisation 2 (SME):
ECS supplier

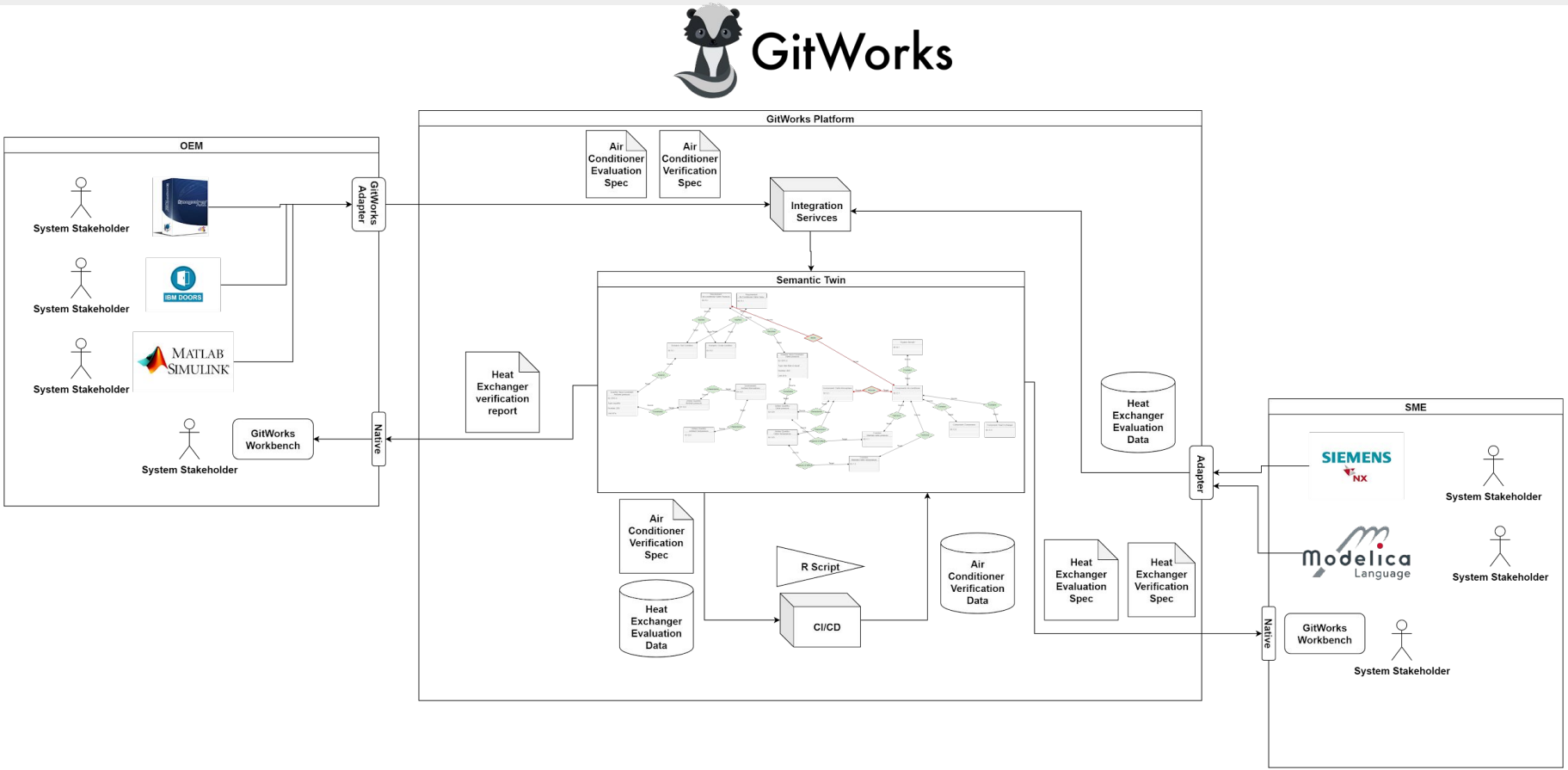


Passenger Air Conditioner

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.

GitWorks workflow



Platform architecture

GitWorks Implementation Architecture

GitWorks Commons

- Package Distribution (GitLab Packages)
- Semantic Discovery & Integration (RDF/Sparql)
- Monetization

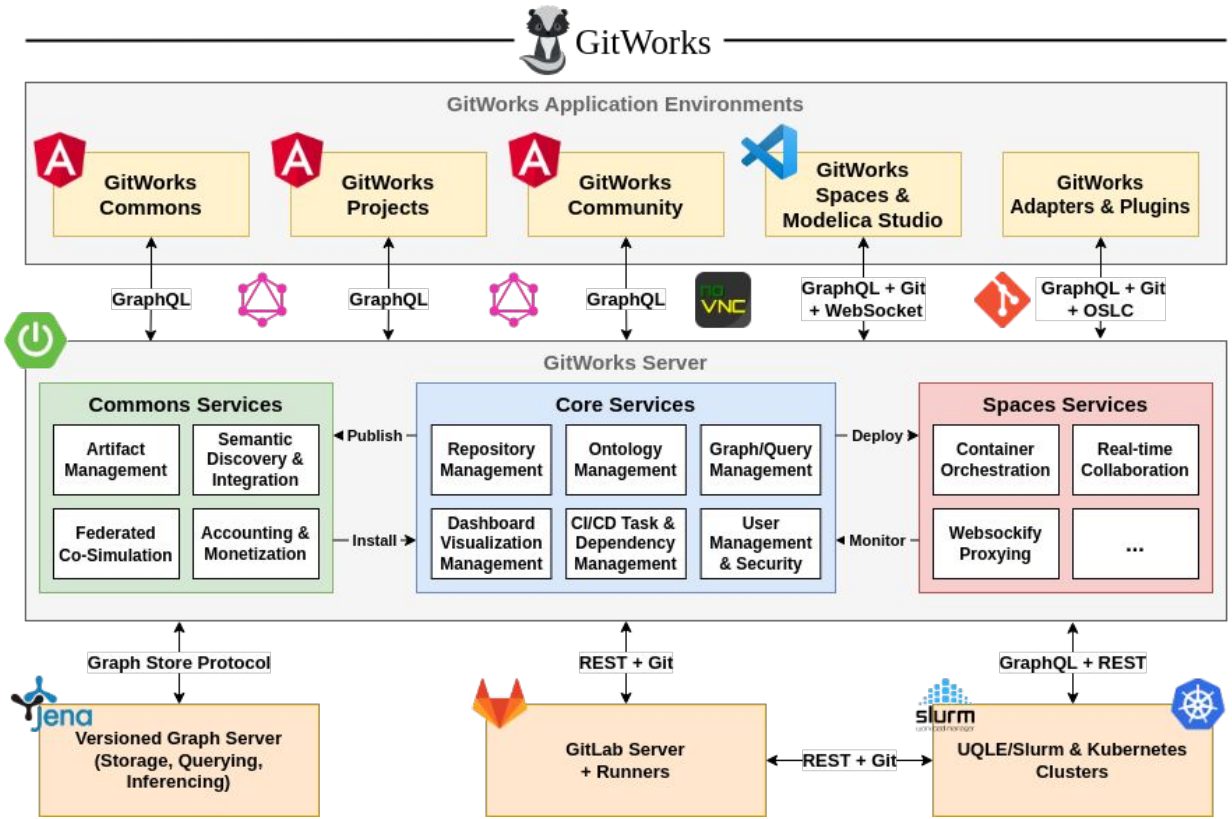
GitWorks Projects

- Git Repository & DevOps Functionality (GitLab/GitLab Runner)
- Ontology Management (OML.js)
- Versioned Triple Store and Querying (Jena + Cytoscape.js)
- Integrated HPC cluster based on Slurm (UQLE)

GitWorks Adapters & Plugins

- Modelica/OML Semantic Web Adapter

GitWorks IDEs & Modelica Studio



GitWorks IDEs and Modelica Studio

Light-weight IDEs via VSCode (for Web)

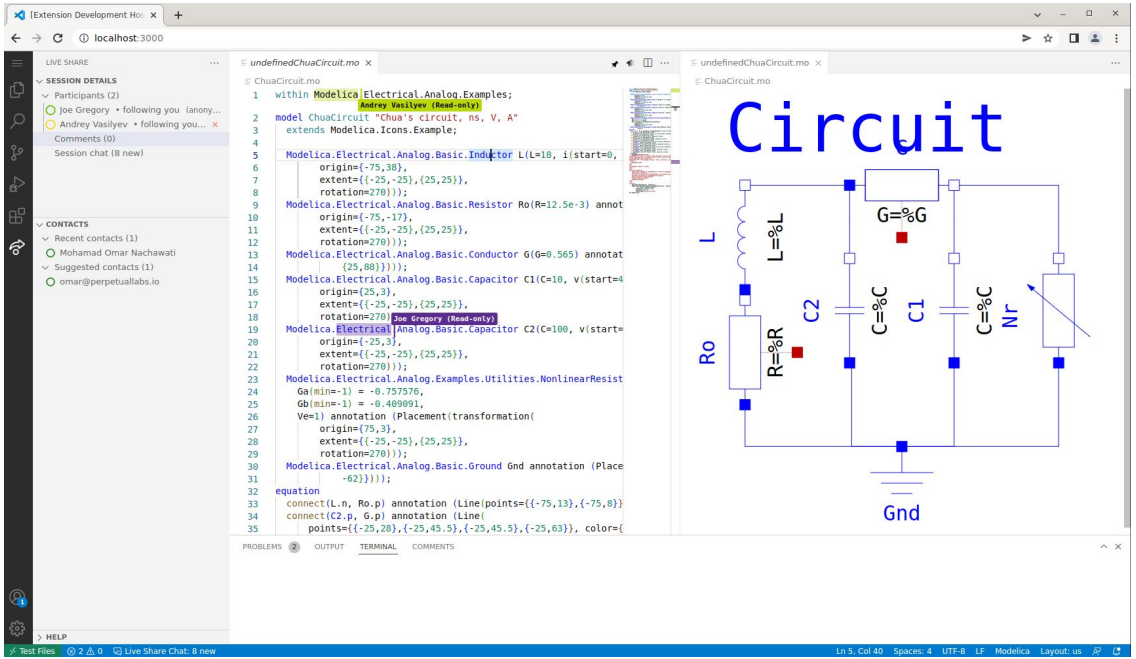
- Integrated Git version control
- Real-time collaboration (LiveShare)
- First Example: Web-based Modelica editor

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



GitWorks IDEs and Modelica Studio

Light-weight IDEs via VSCode for Web

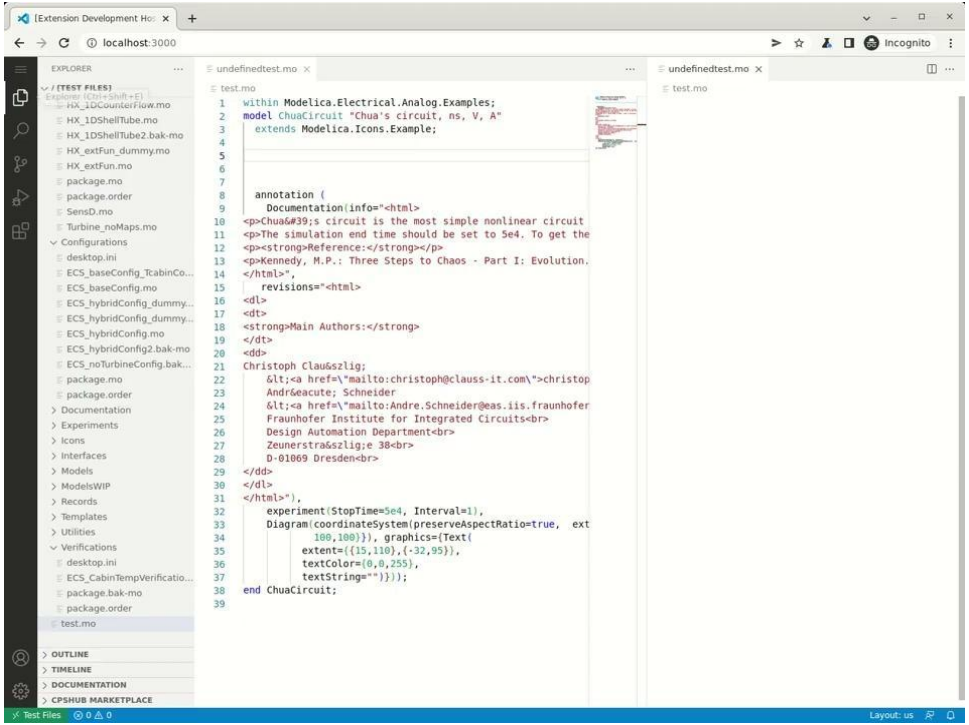
- Integrated Git version control
- Real-time collaboration (LiveShare)
- First Example: Web-based Modelica editor

Heavy-weight workspaces via VDI over noVNC

- Delivers containerized desktop applications over the Web

Modelica Studio

- Browser-based Modelica Editor
  - Synchronized text and diagram editing
- Open core based on OMFrontend.js:
  - Standalone JavaScript library for parsing/flattening of Modelica models
  - Diagram/Icon generation
  - Modelica Semantic Web adapter



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.
- We have proposed a system architecture for GitWorks and have developed a prototype implementation focused around enabling the use of Modelica in the larger MBSE process.
- We have conducted a preliminary case study that has demonstrated the use of GitWorks for the federated design and engineering of a passenger air conditioner system.
- 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, and demonstrate the application to other use cases including satellite systems design and manufacturing

Thank you for your attention!

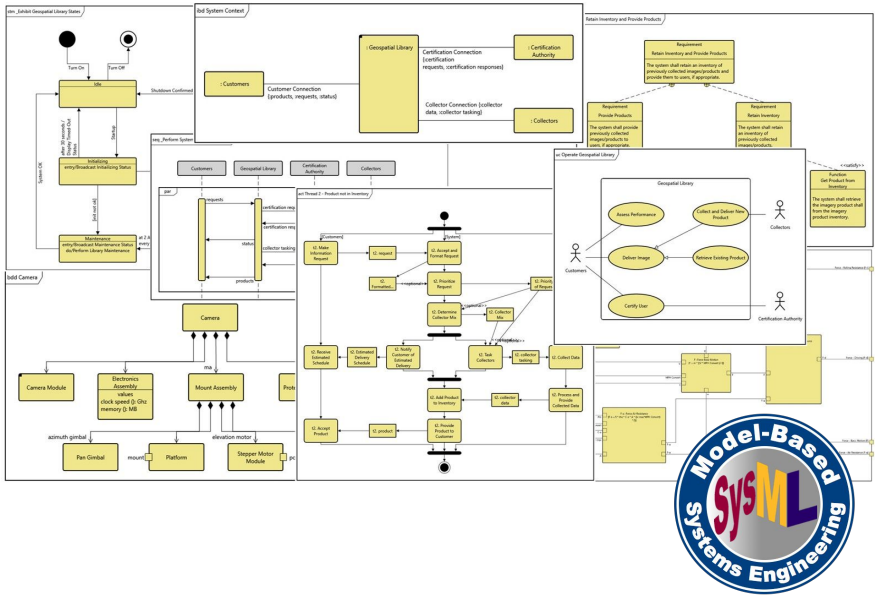
Any questions?

Don’t miss our workshop:
Integrating Modelica in MBSE workflow

Cheat Slides

Quick review: Semantic Systems Engineering

- What about SysML (v.1)?
  - Lacks a number of foundational concepts in Systems Engineering such as SBS and WBS
  - No standard exchange format.
  - Monolithic, does not natively support federation of information
  - Most importantly, does not have formal semantics which prevents automation
  - Bonus points: unnecessarily complicated and with steep learning curve
- An aside on the term MBSE:
  - Systems Engineering is, first and foremost, an engineering discipline
  - All Engineering is “model-based”.
  - Hence MBSE is redundant and, we think, misleading.



“ Drawing boxes and lines may be *modelling*, but is it useful? ”

Steve Jenkins

Why should we do this?

The need comes from the limitations of legacy approaches to Systems Engineering (SE) and Product Lifecycle Management (PLM):

- PLM is still practised largely as a linear, monolith, waterfall process. In theory, it is well understood that an iterative approach to development with early integration and prototyping is a much more effective approach than trying to engineer the complete solution up front. In practice, this has proven very difficult to achieve for complex systems that comprise both Hardware and Software components.

- Integration of heterogeneous data-sources created throughout the Digital Thread (ex: Requirement Models, Architecture Models, CAD, CAE and CAM.) within a rigorous MBSE methodology is extremely difficult to achieve because models from domain-specific tools often lack a common notation and the methodology itself is often not codified in a formal language

- Collecting, aggregating and exchanging information at the system level is often a complex manual activity and highly dependent on the skill level of expert Systems Engineers. This creates bottlenecks that slow down the development process and introduce significant risk of errors and inconsistency. These issues result in very long and rigid development cycles and the proliferation of “Bugs” (the term here is used generically to indicate errors in any kind of engineering model, not just software) throughout the Digital Thread which often lead to extremely costly integration problems later on in the system lifecycle or, even worse, undetected critical defects, risks or vulnerabilities in deployed systems. Ultimately, this situation translates to the exponential increase in the development costs that is currently plaguing industries such as Aerospace, Automotive and Robotics.

- The lack of integration of advanced system simulation and analysis with MBSE methodologies also limits the ability to analyse system-level requirements (such as performance and dependability) in the early design phases of the system life cycle causing the postponement of design decisions to later phases. This, in turn, reduces possible opportunities to study alternative solutions and validation of fitness for purpose. It also increases the costs (in terms of time and skill) of design refinements if irreversible consecutive design decisions are made in the early stages of the development (Stirgwolt, Mazzuchi, and Sarkani 2022). Ultimately, this translates into more conservative design choices, slower technological development and, in general, systems designs that don’t reach their full potential in terms of optimization of performance, emissions or best utilisation of manufacturing resources.

- Furthermore, the complexity and steep learning curve of current MBSE methodologies and their poor integration with the Digital Thread contribute to an increased barrier-to-entry for the widespread adoption of Model-Based Engineering (MBE). Whilst large enterprises can afford the high level of expertise and effort necessary to set-up and manage ad-hoc integration frameworks, SME often lack the human and capital resources to do so. This limits the benefits that could be gained from a wide-spread adoption of MBE across the supply chain.

- In conclusion, development time, cost and risk are increasing exponentially with system complexity. This prevents traditional Engineering Enterprises from being able to effectively deliver new products and capabilities in an environment with continuously evolving requirements.

Why should we do this?

- The problem of think about Systems Engineering in terms of ROI.
  - It's not about ROI, it's about risk management
    - Applying Scientific and Mathematical Rigor is what engineers do
    - It's what makes the results trustworthy and avoids catastrophic failures
  - It's not really possible to do a fair quantitative "before" and "after" comparison
    - Example of Agile in Software development:
      - Empirical, quantitative evidence that adopting agile practices and values improves the effectiveness of software development is mixed and hard to find in the literature today
      - Yet, Agile is the defacto standard for Software development