VehicleForge.mil

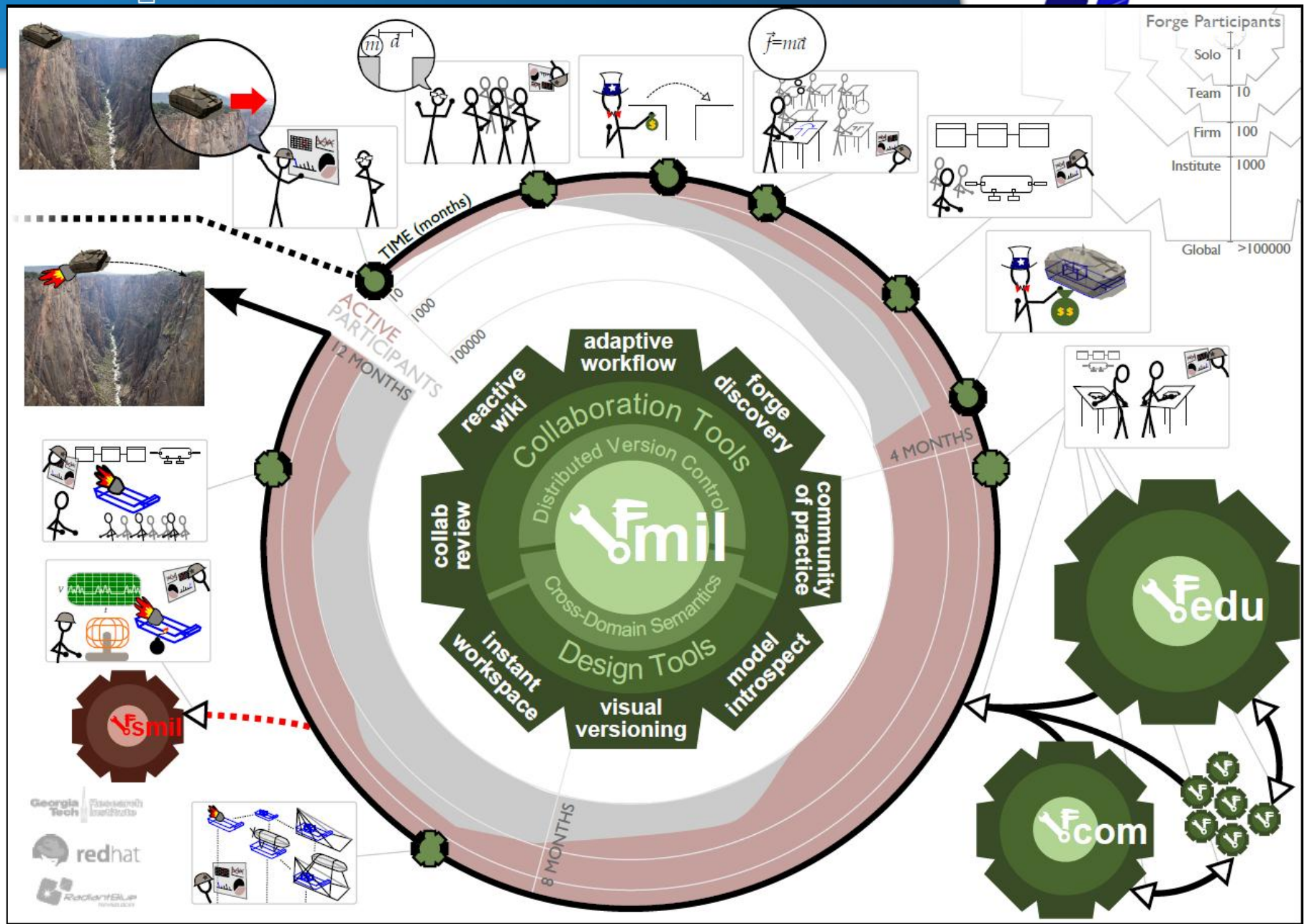
A distributed, semantically-aware framework to support the needs of the open hardware community

Jack Zentner
Senior Research Engineer
Georgia Tech Research Institute
Atlanta, GA 30332
jack.zentner@gtri.gatech.edu

Nick Bollweg
Research Scientist
Georgia Tech Research Institute
Atlanta, GA 30332
nicholas.bollweg@gtri.gatech.edu

John Scott
Senior Systems Engineer
RadiantBlue Technologies
Chantilly, VA 20151
jscott@radiantblue.com

Gunnar Hellekson
Chief Technology Strategist
Red Hat
Raleigh, NC 27606
gunnar.hellekson@redhat.com


Forge Participants
Solo 1
Team 10
Firm 100
Institute 1000
Global >100000
TIME (months)
ACTIVE PARTICIPANTS
12 MONTHS
4 MONTHS
8 MONTHS
10
1000
100000
adaptive workflow
forge discovery
community of practice
model introspect
visual versioning
instant workspace
collab review
reactive wiki
Collaboration Tools
Distributed Version Control
Design Tools
Cross-Domain Semantics
fmil
fedu
fcom
fsmil
redhat

The needs of open source hardware

- Design of cyber-electro-mechanical systems vs the design of software
- Ensuring data provenance while enabling sharing
- Concurrent design and how to ensure design intent across distributed design teams with versioning
- Semantic search, discovery, introspection and linking of design artifacts
- IP-rights and varying governance models
- Multi-classification level enclaves

Hardware vs Software Design

- Hardware designers are not software developers
  - What does synthesis and sizing mean for software?
  - Formal design languages such as SysML only now beginning to be leveraged in Hardware design
- Hardware systems are not the same as software systems
  - Hardware designs are merely abstractions of the system
- Different tools, different artifacts, different needs

Ensuring data provenance while enabling sharing

- Data provenance is a key enabler to help ensure IP rights
- Knowing the provenance of the artifacts and data associated with any project enables better reuse metrics
- Strong data provenance and version control supports the verification stages of systems design

Concurrent design and how to ensure design intent?

- Software developers use integrated unit testing to ensure design intent in collaborative development
- Integrated unit testing for hardware design would, in general, require automated execution of engineering codes
- Ideally each designer would be able to use a different suite of tools

Semantic search, discovery, introspection and linking of design artifacts

- Interface oriented design for hardware development is not generally supported in code integration tools
- Semantic linking of codes/artifacts enables tool agnostic design and development
- Semantic search and discovery enables better re-use and faster differentiation across possible solutions

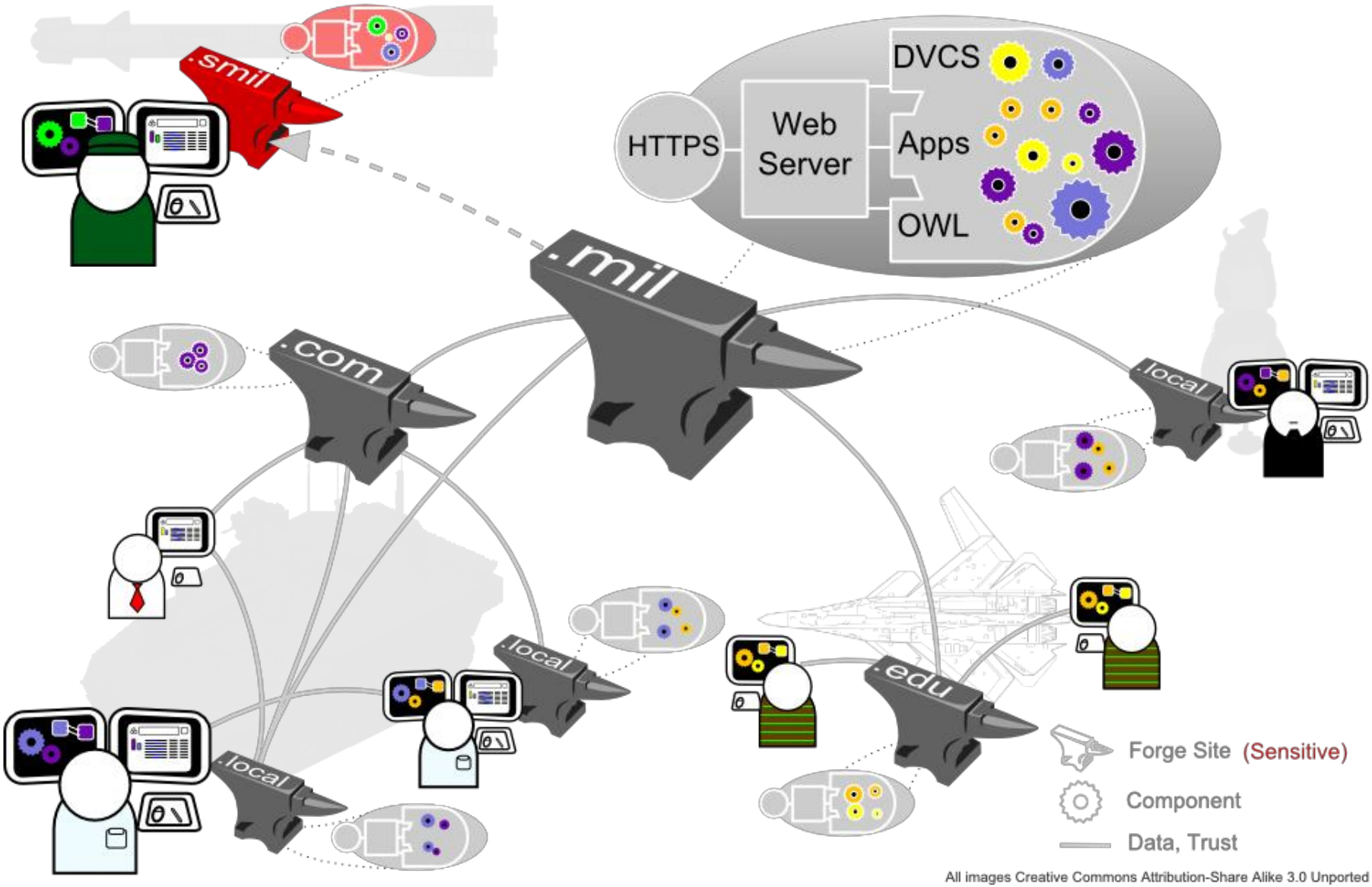
IP-rights and varying governance models

- Why IP-rights and governance is Important
  - Well-understood ground rules make it easier to gain participants in the short term.
  - A vibrant collaborative hardware community reduces costs and eases maintenance for everyone.
  - Limit confusion and liability.
- Hardware IP management is different than software
  - Hardware governed by patents
  - Software by copyrights
  - Open source hardware usually not patented

Multi-classification level enclaves

- Open -> Proprietary -> ITAR Controlled -> Classified
  - Ideally information/designs would seamlessly flow from low to high but have strict controls in the other direction
  - How to ensure US citizen status for open yet ITAR controlled projects?
- The whole goal of the AVM project is to enable a 5x reduction in time to develop military vehicles

VehicleForge.mil Concept



All images Creative Commons Attribution-Share Alike 3.0 Unported

What is VehicleForge?

- A web-centric framework to enable collaborative hardware design
- Built on the enterprise-grade, open source technology stack used on 30k projects by 300k users on SourceForge.net
- Bootstrapped by $1.4M in DARPA funding to support the Adaptive Vehicle Make program and accredited to subset of NIST 800-53
- Open source (MIT License variant) and fully extensible and customizable

Required Software Components

Core Services

Web Server Stack
DVCS & WebDAV
Cryptographic Services
Indexing Engine
Index Search

Component Interoperability

Basic Data Ontology
Semantic Triple Store
Semantic Search

Extensibility

App Engine
Hook Script Engine

META Integration

SysML App
AADL App
Modelica App

Designer Collaboration

Wiki with Forums
Integrated Chat Client
Tasking App

What does VehicleForge Do?

1. **Revision control**: git, mercurial, rug, svn, ...
2. **Federated search:** forge-to-forge, global, project
3. **Change tracking**: ticketing, branching, merging, artifact/asset diff
4. **Collaboration:** wiki, discussion, design review
5. **Context-awareness**: syntax hi-lite, CAD view
6. **Access control**: roles, permissions
7. **Notification:** check-ins, comments, tickets, ...
8. **One-click project provisioning**

How we use VehicleForge to Develop VehicleForge

- VehicleForge used for all development tasking, tracking and version control
- VehicleForge development workflow:
  - Two month AVM development cycle
  - SysML to determine use cases, requirements, development activities and subsystems needed to fulfill a given behavioral\functional feature
  - SysML functional architecture mapped to lettuce scenarios\BDD unit tests and linked to VFI tasking
  - Results of BDD units tests linked back to SysML requirements and features to determine requirements coverage

How VehicleForge Supports Technology Transfer

- VehicleForge is the embodiment of technology transfer
- The crowdsourcing vision of AVM relies on technology transfer to achieve its 5x goals
- VehicleForge is built all on open source software and is itself open source


Build It. Better.
VEHICLEFORGE