A Foundation for the DoD Digital Transformation

Walter W. Wilson
Lockheed Martin
January 28, 2021

1. The Challenge

Recent documents give a vision for a DoD digital transformation. The 2018 "Digital Engineering Strategy" [Digital-Engineering-Strategy_Approved.pdf (sercuarc.org)] has goals that include replacing documentation with digital models, end-to-end digital representation, and rapid innovation in the supporting environments. Dr. Will Roper's papers "Take the Red Pill" [Take_the_Red_Pill-Digital_Acquisition.pdf (af.mil)] and "There is no Spoon [https://software.af.mil/wp-content/uploads/2020/10/There-Is-No-Spoon-Digital-Acquisition-7-Oct-2020-digital-version.pdf] advocate an open-architecture government-owned "tech stack", digital threads that connect lifecycle phases, and digital twins that connect models with real-world systems.

This paper argues that a successful digital transformation needs a good data representation. It should be a textual, human-readable language that can also serve as good documentation. It should have the power of programmability. (The same language can be used to define the supporting software.) A single language would be more efficient for digital threads than a multitude of file formats and interface standards. (We spend a lot converting data from one format to another.) A single programming language for the software tools would support better integration. A digital representation should support formal verification – proving programs correct. In a pervasive digital world, bugs can be catastrophic: https://spectrum.ieee.org/riskfactor/computing/it/coding-error-leads-293-subaru-ascents-to-the-car-crusher, Mars Probe Lost Due to Simple Math Error - Los Angeles Times (latimes.com). Proof may be our best/only hope for cyber security.

Probably the biggest component of the tech stack is the CAD system. My vision for CAD [http://www.axiomaticlanguage.org/A_Vision_for_CAD_released.pdf] aligns with Dr. Roper's call for an open architecture. Commercial CAD holds our design data hostage, leads to huge migration costs, and makes design innovation difficult due to its black-box nature. Also, the long-term accessibility of our design data could be in doubt.

2. A Proposal

This paper advocates a type of logic programming called "axiomatic language" [http://www.axiomaticlanguage.org/] as an ideal foundation for representing digital data and its supporting software. It is a pure specification language, which should provide software engineering benefits. Specifications should be smaller, more readable, more reusable, and more likely to be correct than algorithmic code. The language would provide both programmability and human-readability. The extensibility and metalanguage capability of the language makes it a “universal language” – able to incorporate and subsume the features and expressiveness of other languages. It would be a good host for embedded domain specific languages (DSLs) – i.e., "language-oriented programming". Axiomatic language would also be well-suited to formal verification [http://www.axiomaticlanguage.org/proof.htm].

CAD data would be defined in a textual engineering design DSL defined within axiomatic language. The open-source geometric engine would provide accessible mathematics and powerful scripting for design automation/optimization. (See Boeing paper “The Case for Scripted Process Design and Engineering” [Grandine, 2014 – I have a copy].) The small size and elegance of axiomatic language would make it a good standard for long-term design data preservation [http://axiomaticlanguage.org/LOTAR_Thoughts.html]. Note that an open source CAD system may cost less than what the DoD indirectly spends on commercial CAD licenses. And having government control of the CAD system may help us avoid costly CAD migration.

3. The Project

The implementation of axiomatic language requires automatically transforming specifications to efficient programs – a grand challenge of computer science. In earlier work I created a crude system that could transform some tiny examples [http://axiomaticlanguage.org/BabySteps.pdf]. Since then I have done some proofs and worked on an enhanced transformation system that will handle some larger examples like sorting and arithmetic. This project will continue work on proof and will complete this enhanced transformation system prototype. Milestones will include (1) implementation of a proof checker, (2) working out efficient transformation algorithms that will allow the system to scale to real-world problem sizes, and (3) incorporating proof into some example transformations for a guarantee of correctness. I believe 1000 hours may be a reasonable estimate for these tasks. Note that the definition of a CAD system in axiomatic language would not be part of this project.