MBSE for SOCOM JATF-TALOS

Bjorn Cole, Richard Wise, Sean Higgins - Georgia Tech Research Institute

Nguyen La, Paul Kim – Johns Hopkins Applied Physics Lab

Vikram Mittal, Stephen Gillespie – US Military Academy

POC: Bjorn Cole – bjorn.cole@gtri.gatech.edu (404) 407-6453

JOHNS HOPKINS APPLIED PHYSICS LABORATORY

Georgia Tech Research Institute

UNITED STATES MILITARY ACADEMY WEST POINT

Outline for the Talk

- MBSE experience from organizations supporting TALOS
- Team Structure
- Specific modeling approaches
  - Electrical systems engineering and harness
  - Test coverage and functional description
  - Software/hardware integration
- Overall lessons

Quick Introduction to TALOS

- Tactical Assault Light Operator Suit
- Effort started in 2013 for building ingress
- Supported efforts in developing armor, vision, exoskeletons, and mobile power
- Current effort is the Mark 5 integrated suit
- Government is the integration lead (Joint Acquisition Task Force) with many supporting developers around the country

Team Structure

- Very distributed team
  - 2-3 members at each of the institutions below
  - Remote connection to JATF-TALOS in Tampa
  - Technical performance around the country
- Many practitioners have strengths outside of systems engineering
  - Formal backgrounds in aerospace, electrical, software, mechanical, and bio-inspired engineering
- Weekly sync telecons, best practices and work backlog kept on SOCOM Confluence, one-on-ones by phone and WebEx

JOHNS HOPKINS APPLIED PHYSICS LABORATORY

Georgia Tech Research Institute

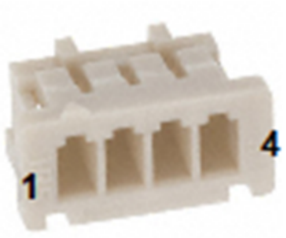
UNITED STATES MILITARY ACADEMY WEST POINT

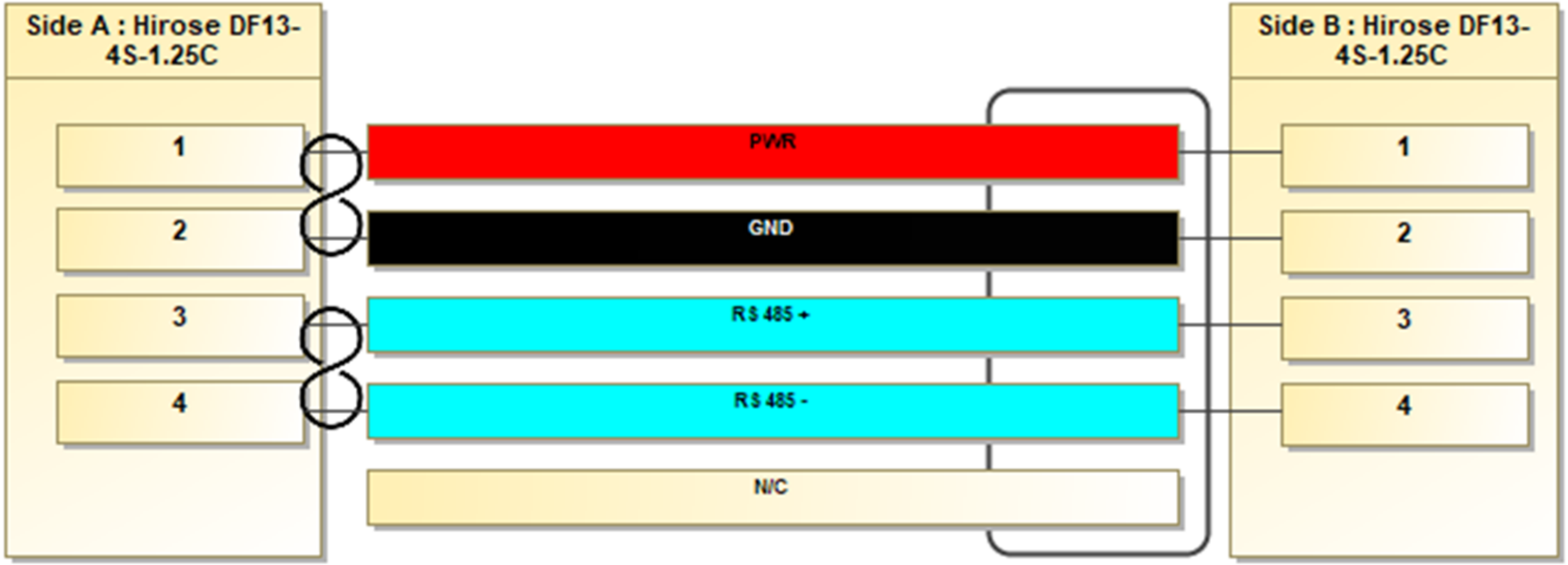
Electrical Systems Engineering Support

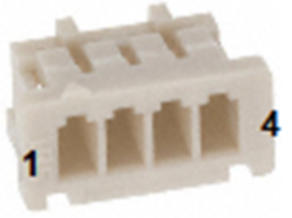
- Capture electrical functions between major components and their relevant standards
  - Physical – bolts, straps, mechanical hard points in structure
  - Logical – data or signals in various formats
  - Electrical – power supply

Electrical Systems Engineering Support

- Implementation of carriers for electrical functions now supported in the model and mapping to wire harness
- Harness model formatted to match harness engineer at APL’s visual expectations
  - Captures pair twisting, pinouts, connector terminating and bare wire





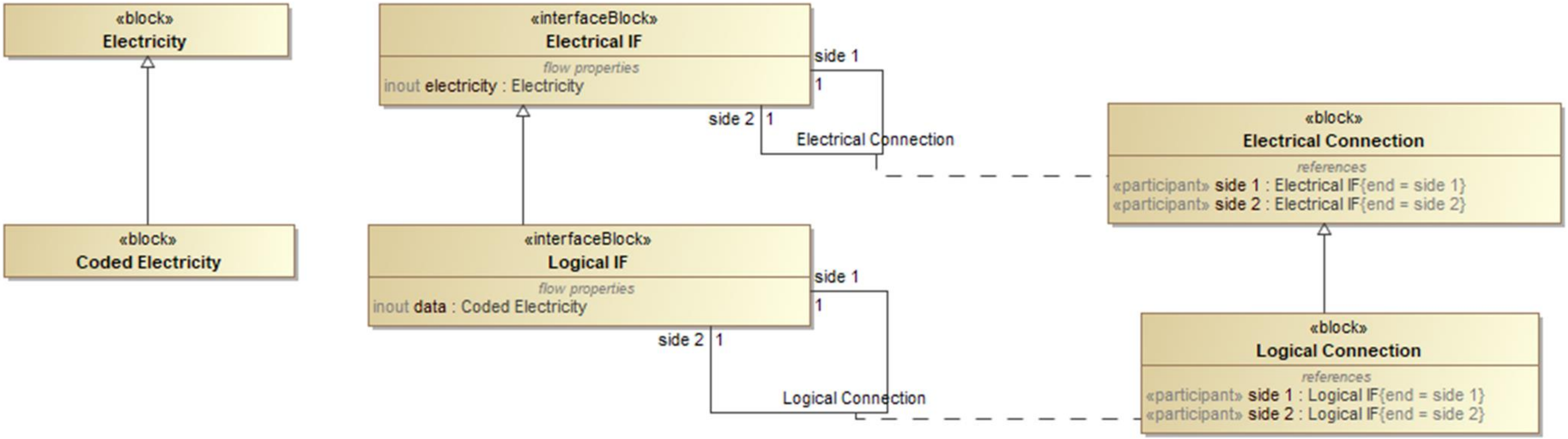


Electrical Systems Engineering Support

- Actual wire harness bound to electrical function representation in the model to support reporting and comprehensive capture of implementation
- Physical to functional connection also drove a revision to libraries to acknowledge that physical layer of data signals is still electricity

Electrical Systems Engineering Support

- Basis of function library took multiple revisions to arrive at simple unification of physical data layer and electricity
- All electrical flows can be connected; question is where a code reader is available to interpret signals



Test and Function Linking

- Very lightweight approach to connecting tests to functionality of integrated system
- Built for prototyping efforts where test coverage is important, but repeatability and auditing are not
- Criticality of test flows up to CONOPS and necessity
- Also a trace to performance requirements ("how well")

Test	Covered CI Function	CI Supports System Function	CI Function Performed By	Date of Test	Characterization
Check out low power distribution to LV ports	Produce Power from Storage	Provide Regulated Power at Voltage for Electronics	Advanced Battery-Only Solution Batteries	7/12/18	Power is supplied to high-voltage and all low-voltage ports

Test and Function Linking

- Top-down flow
  - CONOPS down to system functions down to CI functions
- Bottom-down flow
  - CI Functions up to system functions seeking CONOPS use

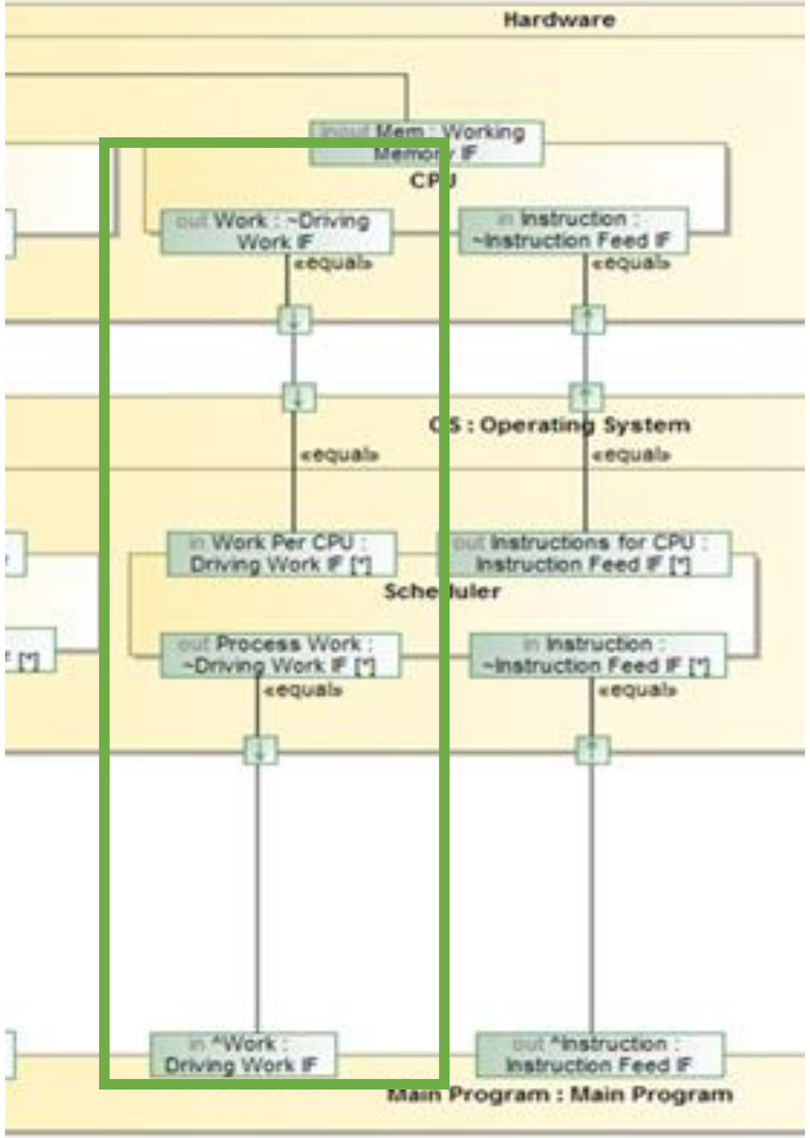
Test and Function Linking

- Experience of effort showed nicely the two end points of formality and framework weight
  - Heavyweight Test and Evaluation Framework built off of the UML Test Profile – rigorous approach for programs of record and integrated schedules
  - Lightweight linking – provides visibility into coverage and criticality but doesn’t go to logistics or auditing
- Heavyweight framework captures all information necessary to plan a test; question is who comes into the loop

Hardware/Software Integration

- Modeling pattern based on reality of software
  - Abstract model of software flow from UML provides a description of major blocks of algorithm, data flow, and order of execution
  - Real-time software needs to know about available resources (computing time and memory) to assure deadlines are met
  - Real software is interpreted or compiled into machine code for execution on processors or controllers

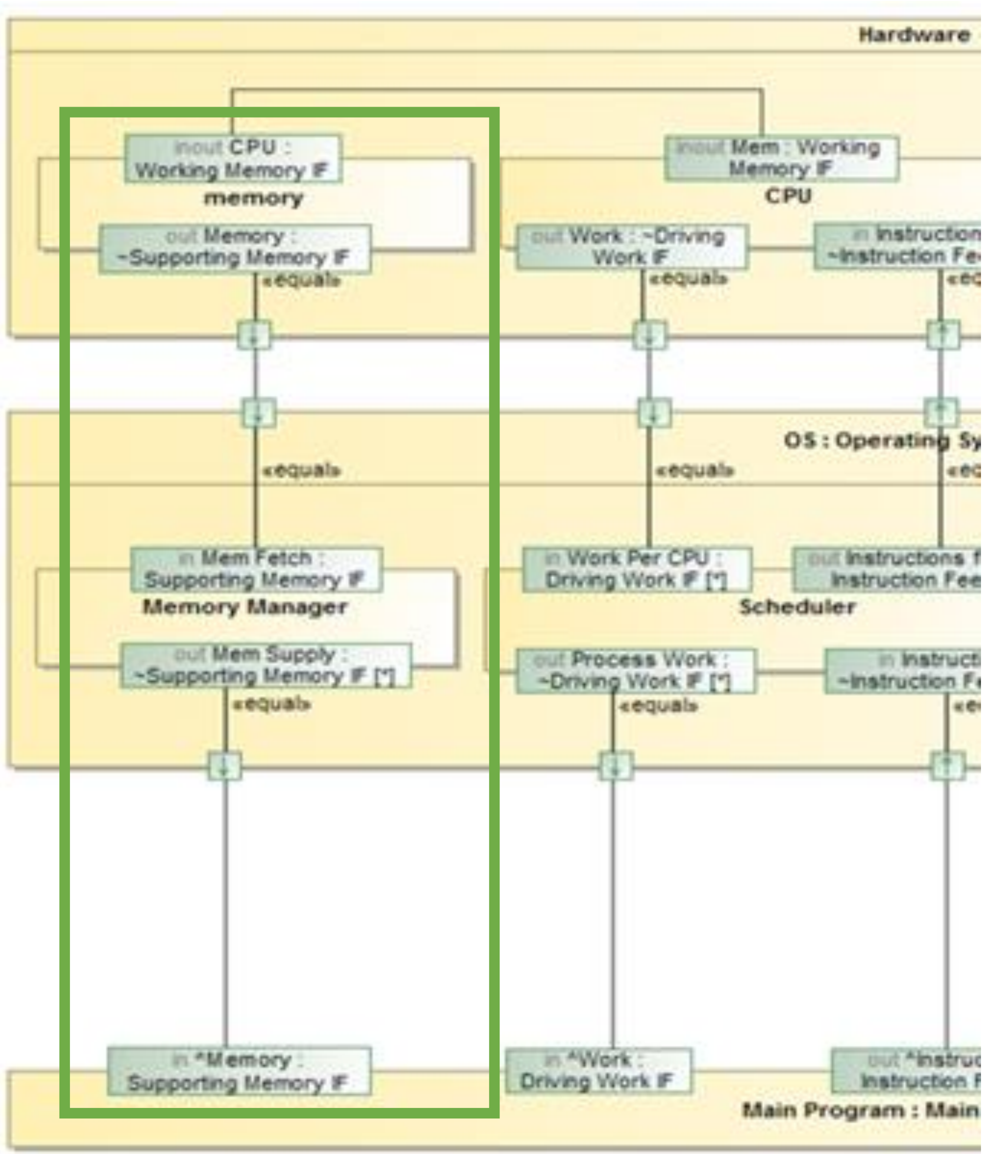
Hardware/Software Integration



“Driving Work” interface talks about how compute cycles are made available to move the program forward

This shows the full stack of a main program accessing compute through the Operating System, which schedules compute availability to different programs

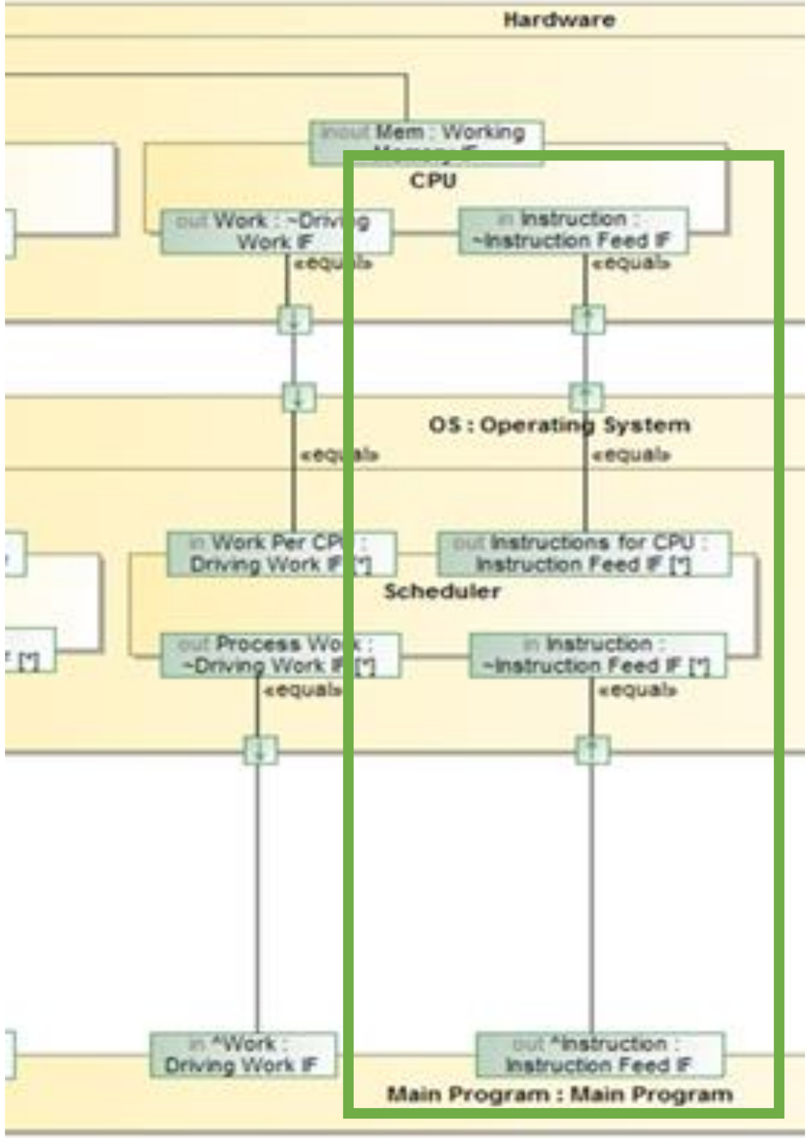
Hardware/Software Integration



"Working Memory" interface talks about how much memory a program can access to store variables and working values

This shows the full stack of a main program accessing memory through the Operating System, which has a memory manager to supply programs

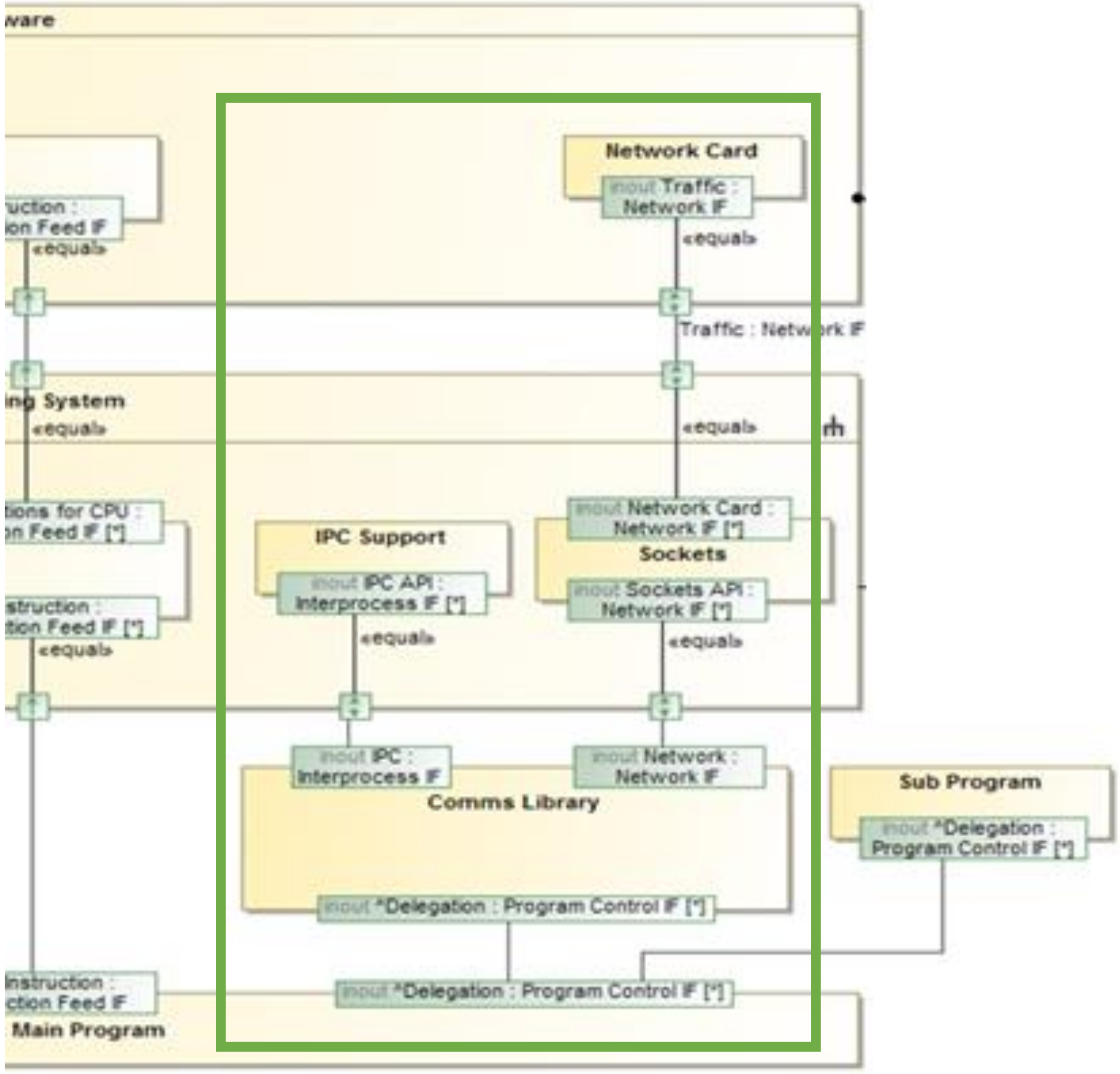
Hardware/Software Integration



“Instruction Feed” interface talks about how program is rendered into a stream of instructions over time that flows at the rate of available resources

This shows the full stack of a main program loaded onto the CPU as mediated by the Operating System

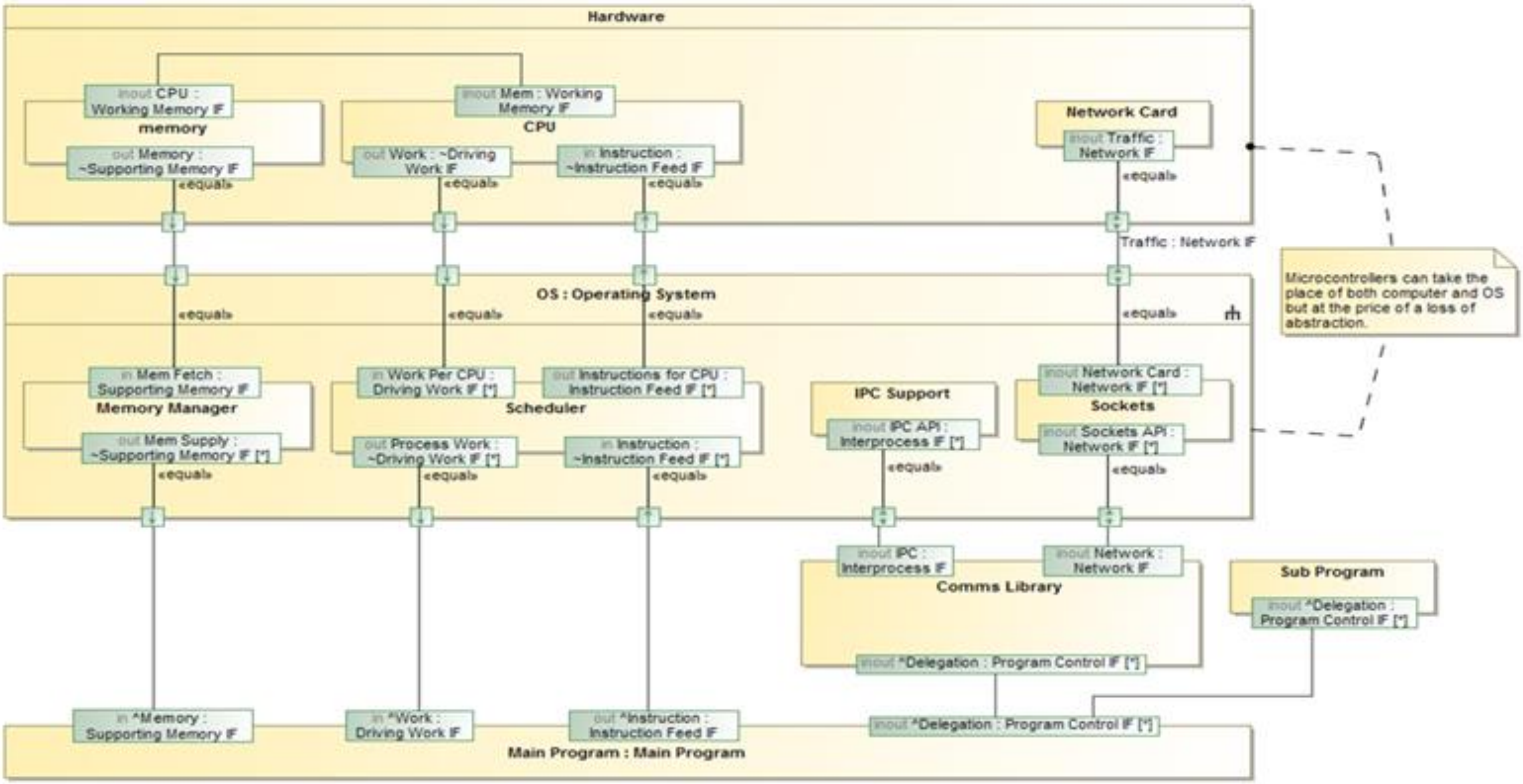
Hardware/Software Integration



This area shows how a main program can delegate resources to and forward instructions from sub programs

One type of sub program is a networking and communications library that can talk to relevant parts of the OS and supporting hardware to analyze connectivity of services to each other over networks

Hardware/Software Integration



Lessons: Keeping the Model Clean

- Any long-running system model eventually needs a mechanism to support cleaning and removal of unused elements
- Developed a heuristic to help
  - Table of Contents points to diagrams that are of interest
  - Only elements on a diagram or supporting what is on diagram (e.g., more general Blocks of portrayed Blocks) are of interest
  - Everything else is marked for potential cleaning through model queries

Lessons: Co-location vs Remote Support

- Systems teams require some degree of co-location or other means of getting immersed in technical design and approach
  - Hallway conversations still matter
  - Remote immersion is possible (and enhanced through a shared systems model) but requires significant effort

Lessons: Finding the Right Weight

- All systems engineering and project management have a “consent of the governed” aspect – if work is not well-justified or tracked it will be de-prioritized
- Finding right weight on test tracking required a back-to-basics thought on purposes of test products
  - Assuring coverage versus supporting audits
  - Looking over planners’ shoulder or providing freedom
- Keep in mind that this effort is not free – it consumes time and schedule!

Lessons: Directions for MBSE Tooling

- MBSE tools are currently oriented for architects and systems engineers to develop a high-level description of a system within the tool and pass on to other engineers
- When direction of data is reversed (other engineers to MBSE's), the tools are far too slow for good response
  - Non-responsiveness is a major threat to SE credibility on a project and a major opening for the development of "shadow models"
- Current importers are helpful, but too trivial for connection to custom spreadsheets

Summary

- Organizations below have supported a virtual, distributed model-based systems engineering team for TALOS
- Developed patterns driven by engineering needs near the hardware and software
- Lessons learned based on team dynamics and challenges of finding right amount of SE to apply to integrated prototype and "flight demo" of a system