University/Industry Collaborations:

Challenges and Strategies for Success

Eric M. Dashofy
The Aerospace Corporation

ASEE&T 2010
March 12, 2010

Outline

- Advantages of university / industry collaborations in software architecture
  - *As well as challenges*
- Finding and achieving common goals
  - *Examples of software architecture in industry*
    - Diversion: standards and artifacts most often encountered
- Strategies for success
- Pitfalls and caveats
  - *Personal reflections on projects: what worked, what didn't*
- Classroom strategies

A Historical Metaphor

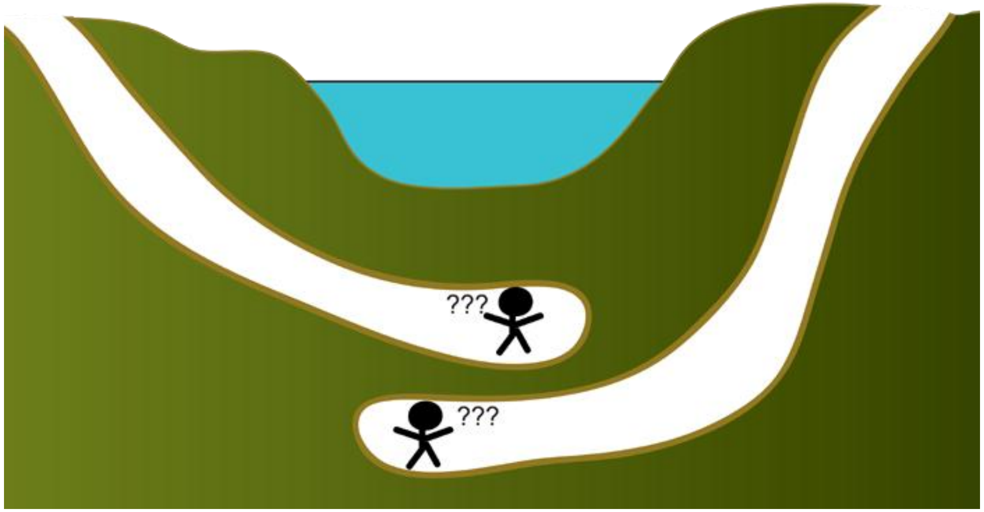
- The golden spike joined the two halves of the transcontinental railroad in Promontory, Utah in 1869

Why Collaborate?

- From Academic Perspective
  - *Validate and evaluate new approaches and techniques*
  - *Gain insight about industry problems and domains*
  - *Gain advocates and supporters for new approaches*
- From Industry Perspective
  - *Evaluate new methods and techniques for solving their own problems*
  - *(Occasionally) solve critical problems*
  - *Increased knowledge of the cutting edge*
    - Also inspiration
- From both perspectives
  - *Funding opportunities*
  - *Networking opportunities*

Another Historical Metaphor

- The “chunnel” (channel-tunnel) provides a way to travel under the English Channel by rail.



Artist’s Rendering of the Chunnel Effort

Key Challenges

Different Goals and Constraints

- University
  - *Create and disseminate (publish) research results*
  - *Do something new*
  - *Invent novel approaches*
  - *Create general solutions*
  - *Teach students*
  - *Softer deadlines*
  - *More tolerant of risk of failure*
  - *Disseminate results widely (e.g., At conferences, journals, on the Internet)*
- Industry
  - *Create products that satisfy customer requirements*
  - *Do something that works*
  - *Use best practices*
  - *Solve immediate problems*
  - *Increase competitive advantage*
  - *Harder deadlines*
  - *Less tolerant of risk of failure*
  - *Dissemination restricted (e.g., proprietary, classified)*

Finding Common Ground

- Start Small
  - *Look for small opportunities to make a connection*
  - *Researchers: Shouldn't expect practitioners to reorient their entire operation around the latest ideas*
  - *Practitioners: Shouldn't expect research ideas to be fully formed and productized*
- Prepare for growth
  - *Look for and develop approaches that facilitate incremental adoption*
  - *Both parties should prepare to capitalize on successful efforts and be thinking of longer-term collaborations*
  - *Be proactive in looking for win-win funding opportunities*

Finding Common Ground (cont.)

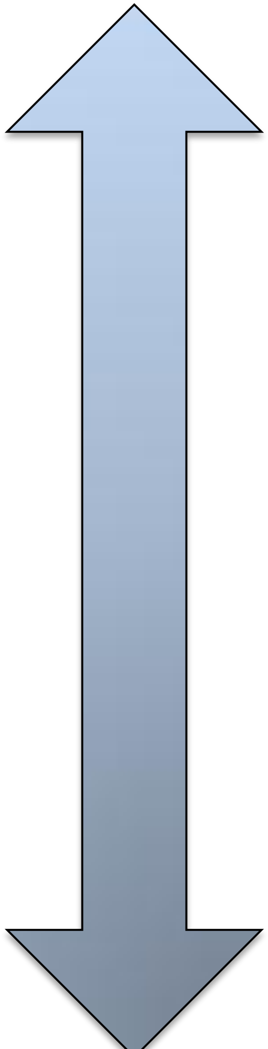
- Focus on integrating novel approaches with approaches and technologies used in industry
- Perpetually look for research *areas* that satisfy common and disparate goals
  - *Areas where there are important, open research questions*
  - *Areas where (incremental) improvements will make an immediate difference*
  - *Areas where there's a lot of future potential*
  - *Areas close to, <u>but not necessarily on</u>, the critical path*

What can Architecture Research Offer Industry?

- In industry, design above the level of classes and modules occurs
  - *May not be principled or adequately supported*
  - *Often, resulting artifacts are MS Office documents (MS Word, PowerPoint)*
- Lots of room for improvement
  - *Better ability to communicate (about) designs*
    - New visualizations, new models of systems
  - *Better ability to understand implications of designs and design decisions early*
  - *Better management of software evolution (both at development and run-time)*
  - *Better ability to understand and explore tradeoffs*
  - *Better ability to understand the relationships between products (e.g., product-lines)*

A Spectrum of "Architecture"

More Abstract,
More Social



More Concrete,
More Technical

People, processes

Quality Attributes and Trade-offs

Semi-rigorous models

Rigorous Models

Formal Analysis,
Mapping to Code,
Tie to Lifecycle

Social-Science Projects

Technology-centric Projects

Different kinds of projects at different points on the spectrum

A Spectrum of Projects

Projects with Pedagogical Aims ⟷ Projects with Research Aims

- Class projects (10-16 weeks?)
- Mostly undergrads (free!)
- Do not have to break new ground
- Many teams may do same assignment
- May have a desired outcome (i.e., for grading)

- Research projects (longer)
- Mostly grad students (paid)
- Generally do have to break new ground
- Duplication is unlikely
- Outcome unknown in advance

Different kinds of projects at different points on the spectrum

Examples of Potential Projects

Pedagogical and Abstract

- Observe and evaluate design meetings
- Evaluate quality trade-offs based on documentation
- Tie architecture to business goals

Research and Abstract

- New techniques for gathering stakeholder input
- New approach for evaluating quality trade-offs
- Ethnography of software architects and designers

Pedagogical and Technical

- (Re-)develop various views/models of a system
- Given a realistic problem, develop one or more designs
- Reverse-engineer architecture from code
- Evaluate architectural artifacts

Research and Technical

- Develop novel views/models of a system
- New approach for architectural recovery
- New approach for modeling, analysis, or simulation
- New approach for product-lines

Relationships horizontally: often related to scope and novelty

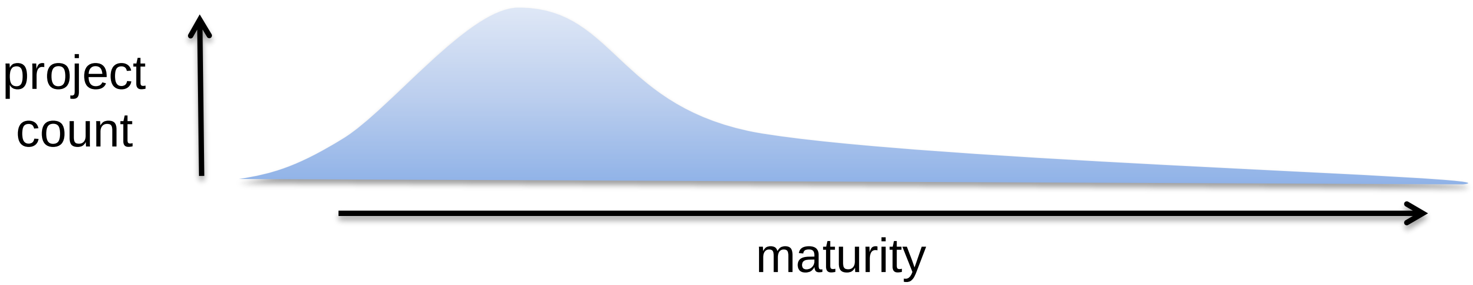
Why is Doing Architecture Projects with Industry Challenging?

- Proprietary Nature of Architectural Information
  - *Architectural information is high-value and practitioners want to protect it*
  - *May also be under other restrictions (e.g., ITAR)*
  - *May only be available inconveniently (e.g., at company site)*
    - With no network access for visitors
- Lack of Mature Architectural Practices in Industry
  - *Not everything may be documented*
  - *Documents may be incomplete or evolving*
  - *Documents may answer the wrong questions*
- Complexity of Architecture
  - *Documents may be hundreds or thousands of pages*
  - *"Big picture" may live in the heads of project team*

Why is Doing Architecture Projects with Industry Challenging? (cont.)

- Design is Hard to Access and Influence
  - *Practitioners generally skeptical of outsiders meddling in design*
  - *Most of project lifecycle is not spent designing, but dealing with ramifications of design*
  - *Best way to understand design is to talk to designers*
    - Designers and “big picture” people are often the most oversubscribed
    - Architects doubly so

The State of Architecture in Industry



- Architecture is a clear focus in most big projects
  - *However, the state of that architecture varies*
- Maturity tends to be "medium rare"
  - *Focus in lifecycle on architecture and design*
  - *Development of (often extensive) documentation*
  - *Prose and UML (often ambiguous) are dominant forms of models*
  - *Rationale capture can be an issue*
- Few projects that aren't doing anything about architecture
- Very few projects that are pioneering architecture approaches

What do you tend to see in industry architecture?

- Personnel
  - *Many implementers working on very specific portions of the project*
  - *A few designers and "big picture" people*
    - Find by looking for the cubicles/offices with lines outside them
- Organizations
  - *Customer organizations (acquiring products)*
  - *Developer organizations (developing products)*
  - *Oversight organizations (oversee acquisition and development*

What do you tend to see in industry architecture? (cont.)

- Artifacts
  - *Often*
    - Design documents mostly separate from the system implementation
    - Word documents containing many box-and-arrow diagrams
      - *A few diagrams may be very semantically-rich and well-thought-out but these can be hard to find*
    - UML
      - *Some use of profiles and consistency rules across diagram types*
  - *Sometimes*
    - Framework-compliant diagrams/documents (e.g., DoDAF, SysML, RM-ODP)
    - Decent simulators
  - *Rarely*
    - Artifacts with strong traceability / that are part of the implementation
    - Rigorous or formal models

A Diversion: “Industry Standard”

- Standards and frameworks are huge drivers of:
  - *What kinds of artifacts get generated*
  - *How architecture is conceived by various stakeholders*
  - *Contract deliverables (sometimes)*

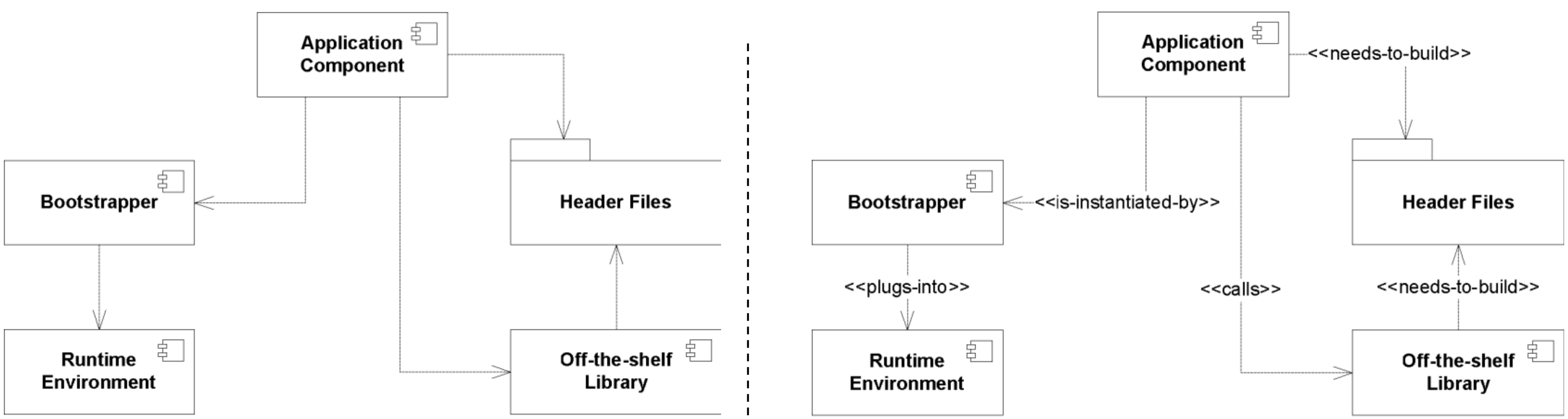
- A quick tour through standards you’re most likely to encounter
  - *IEEE 1471*
  - *UML*
  - *SysML*
  - *DoDAF*
  - *RM-ODP*

IEEE 1471

- Recommended practice for architecture description
  - *Often mandated for use in government projects*
- Scope is limited to architecture descriptions (as opposed to processes, etc.)
- Useful as a starting point for thinking about architecture description
  - *Defines key terms*
  - *Identifies the importance of stakeholders and advocates models that are tailored to stakeholder needs*
- Does not prescribe a particular notation for models
  - *Does prescribe a minimal amount of content that should be contained in models*
- Reasonable definitions of views and viewpoints
- Being compliant does not say much about quality

The Unified Modeling Language (UML)

- Fourteen (14) types of diagrams
  - *Addressing structural and behavioral aspects*
- As a standard, primarily prescribes a syntax
- Some semantics with purposeful ambiguity
  - *Also some implicit and explicit consistency rules across diagrams*
- Encourages specialization of the standard through the use of profiles
  - *Profiles are “mini-standards”*
  - *Profiles can customize existing diagrams but cannot define new ones*



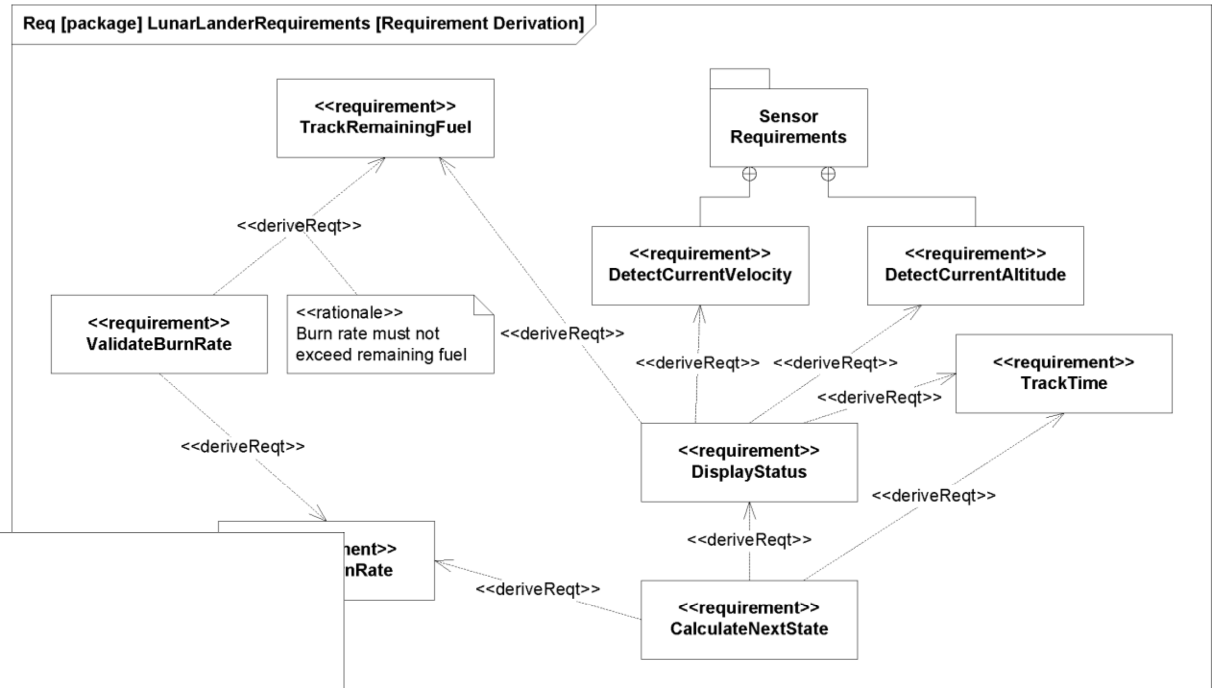
SysML

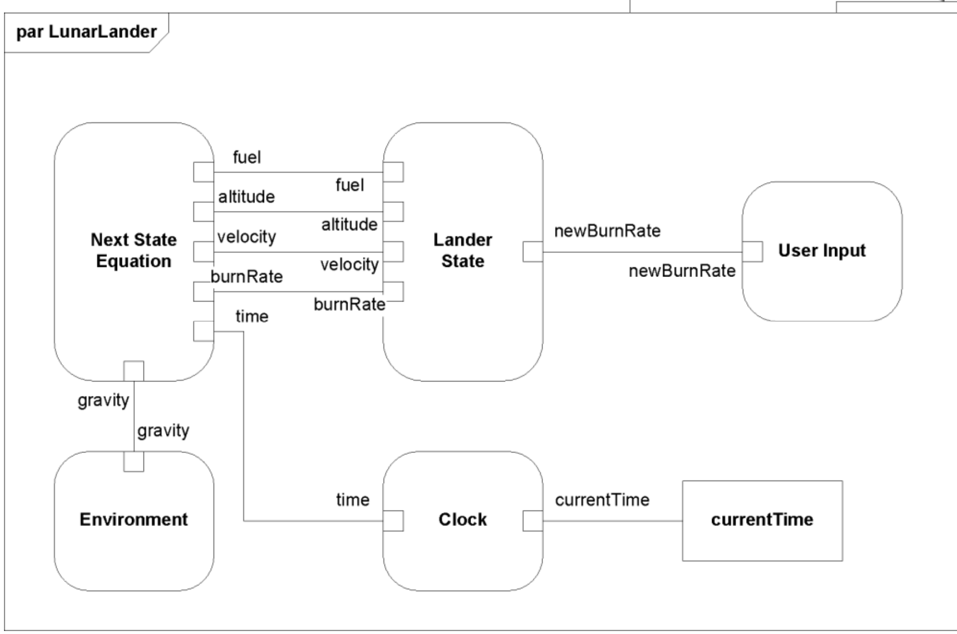
- An extended version of UML
- Developed by a large consortium of organizations (mainly large system integrators and developers)
- Intended to mitigate UML's "software bias"

- SysML developers found UML insufficient
  - *Initially decided profiles were not enough to resolve this*
  - *Developed new diagram types to capture system-engineering specific views*
  - *Limited momentum among tool vendors; focus shifting to more heavily use UML profiles*

SysML

SysML Requirement Diagram





SysML Parametric Diagram

Department of Defense Architecture Framework

- DoDAF, evolved from C4ISR
  - *Has some other international analogs (MoDAF)*
  - *'Framework' here refers to a process or set of viewpoints that should be used in capturing an architecture*
    - Not necessarily an architecture *implementation* framework
- Identifies specific viewpoints that should be captured
  - *Includes what kinds of information should be captured*
  - *Does not prescribe a particular notation for doing the capture*
- Some vocabulary inconsistency with, e.g., IEEE 1471

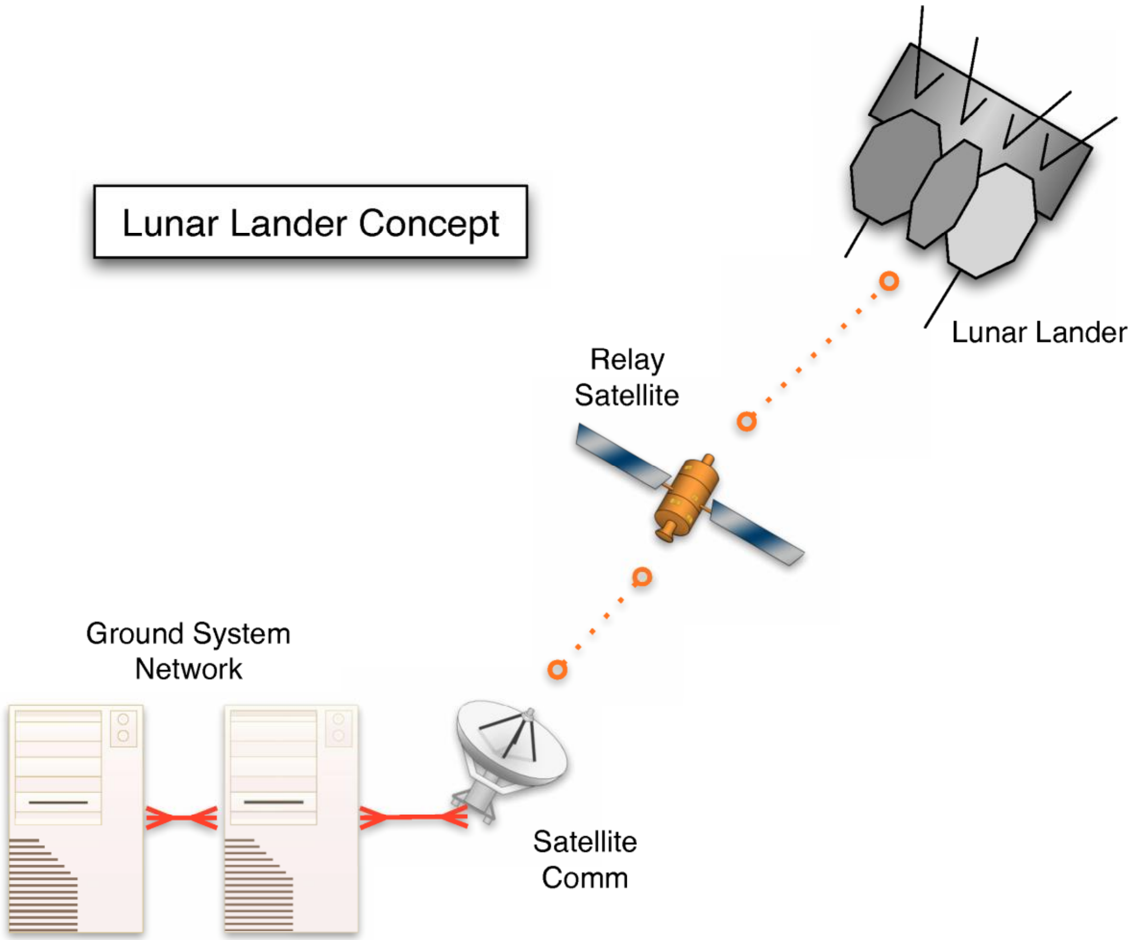
Concept	Common Term	DoDAF Term
A set of perspectives from which descriptions are developed	Viewpoint set	View
Perspective from which descriptions are developed	Viewpoint	(Kind of) Product
Artifact describing a system from a particular perspective	View	Product

Department of Defense Architecture Framework (cont.)

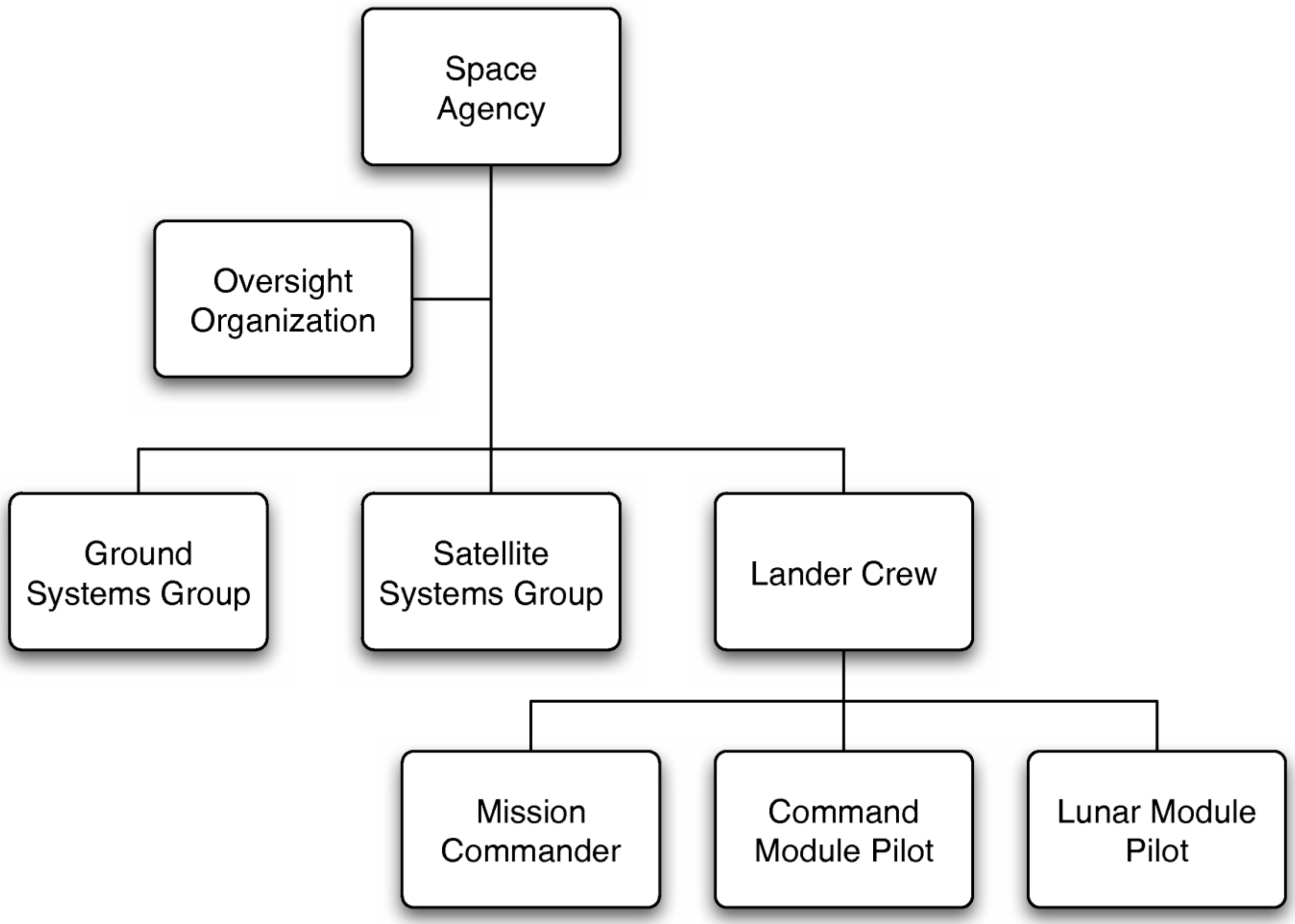
- Three views (common: viewpoint sets)
  - *Operational View (OV)*
    - "Identifies what needs to be accomplished, and who does it"
    - Defines processes and activities, the operational elements that participate in those activities, and the information exchanges that occur between the elements
  - *Systems View (SV)*
    - Describe the systems that provide or support operational functions and the interconnections between them
    - Systems in SV associated with elements in OV
  - *Technical Standards View (TV)*
    - Identify standards, (engineering) guidelines, rules, conventions, and other documents
  - *Cross-cutting products – 'All Views' (AV)*
    - E.g., dictionary/glossary of terms

DoDAF OV-1 Product

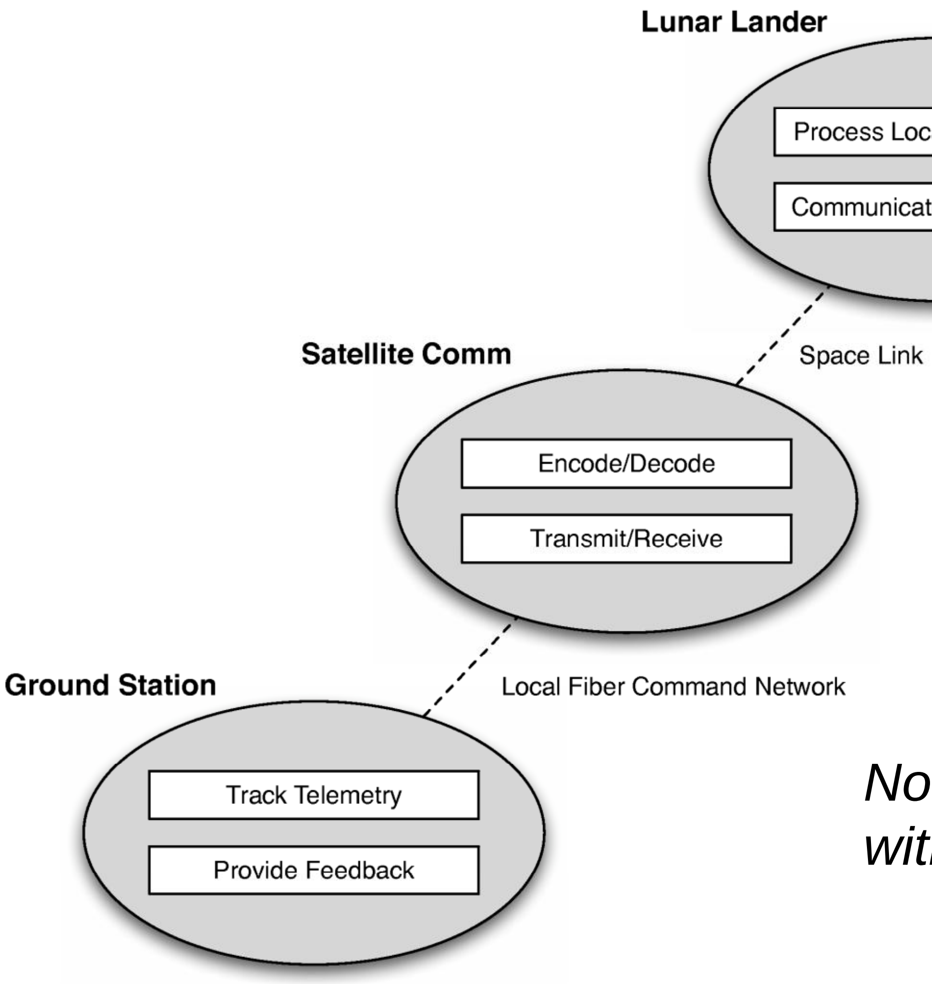
"High-Level Operational Concept Graphic"



DoDAF OV-4 Product
"Organizational Relationships"



DoDAF SV-1 Product
"Systems Interface Description"



Note implied correspondence with OV-1 entities

DoDAF SV-3 Product

"Systems-Systems Matrix"

from \ to	Ground Station	Satellite Comm	Lunar Lander
Ground Station		Ground Feedback (TCP/IP)	
Satellite Comm	Lander Transmissions (TCP/IP)		Ground Feedback (Space Protocol)
Lunar Lander		Lander Transmissions (Space Protocol)	

- *Note correspondence with SV-1*
- *One of several "N^2"/matrix diagrams in DoDAF*

DoDAF TV-1 Product

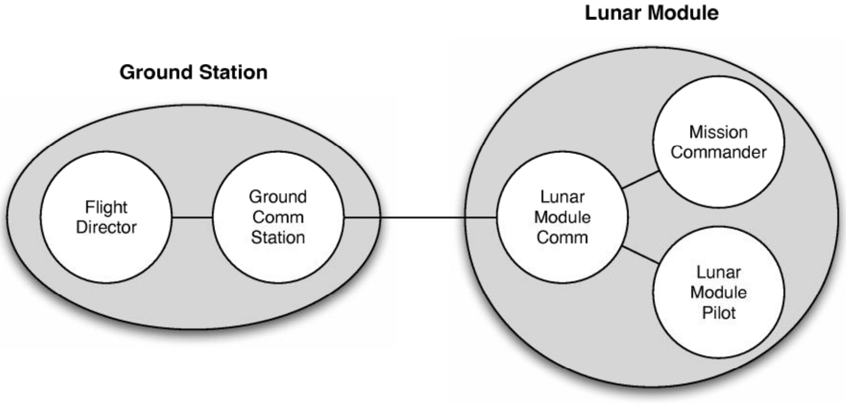
"Technical Standards Profile"

Standards for SV-1 Systems			**Ground Station**	**Satellite Comm**	**Lunar Lander**
Service	*Service Area*	*Standard*			
Information Technology Standards	Operating System Standard	ISO/IEC 9945-1:1996, Information Technology - Portable Operating System Interface (POSIX) - Part 1: System Application Program Interface (API)	Baseline: 1 January 2007	Baseline	Baseline + 3 mos
Information Transfer Standards	Data Flow Network	Extensible Markup Language (XML) 1.0 (Fourth Edition) W3C Recommendation 16 August 2006	Baseline + 6 mos	Baseline + 6 mos	
	Physical Layer	FDDI / ANSI X3.148-1988, Physical Layer Protocol (PHY) -- also ISO 9314-1	Baseline + 3 mos	Baseline + 3 mos	

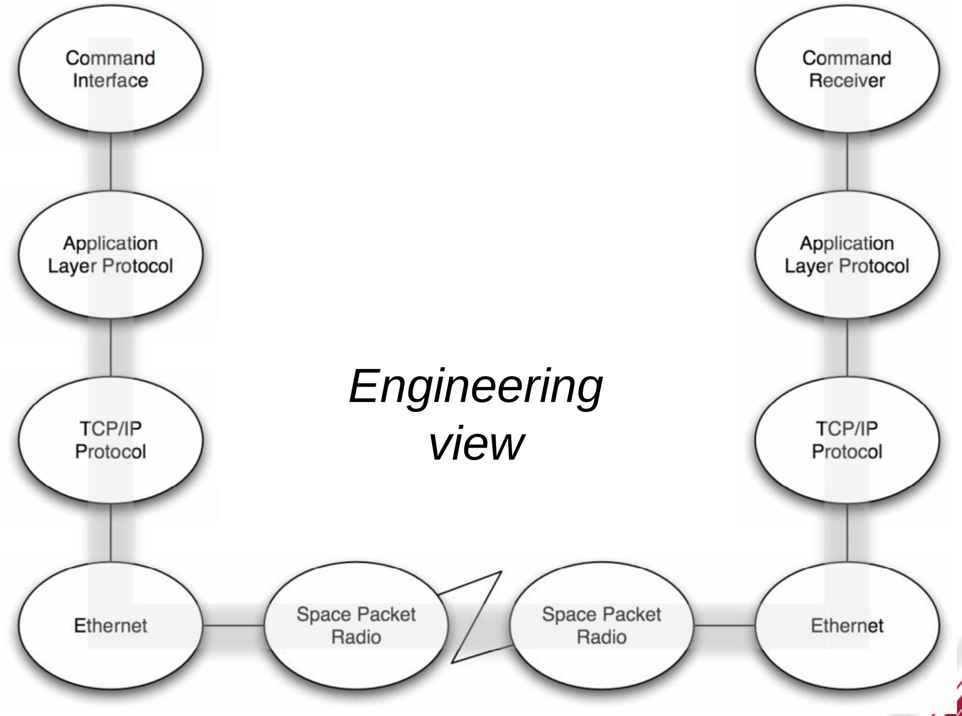
- *Somewhat non-traditional for an architecture view*

RM-ODP

- Another standard for viewpoints, similar to DoDAF but more limited in scope; resemble DoDAF SV
  - *Prescribes 5 viewpoints for distributed systems*



Enterprise View



Engineering view

Diversion: Takeaways

- These standards tend to be abstract and do not provide specific design guidance
  - *Provide specific guidance about **what** to capture, but not **how** to capture it*
  - *Diagrammatic notations improve communications*
    - But generally need additional prose documentation for semantics
  - *Consistency across views is generally an exercise for the user*

- Additional standards
  - *Standard tools (Rose, Rhapsody, System Architect)*
  - *Analysis and Evaluation Approaches (ATAM)*
  - *Process and documentation standards (ISO 9000; CMM/CMMI)*

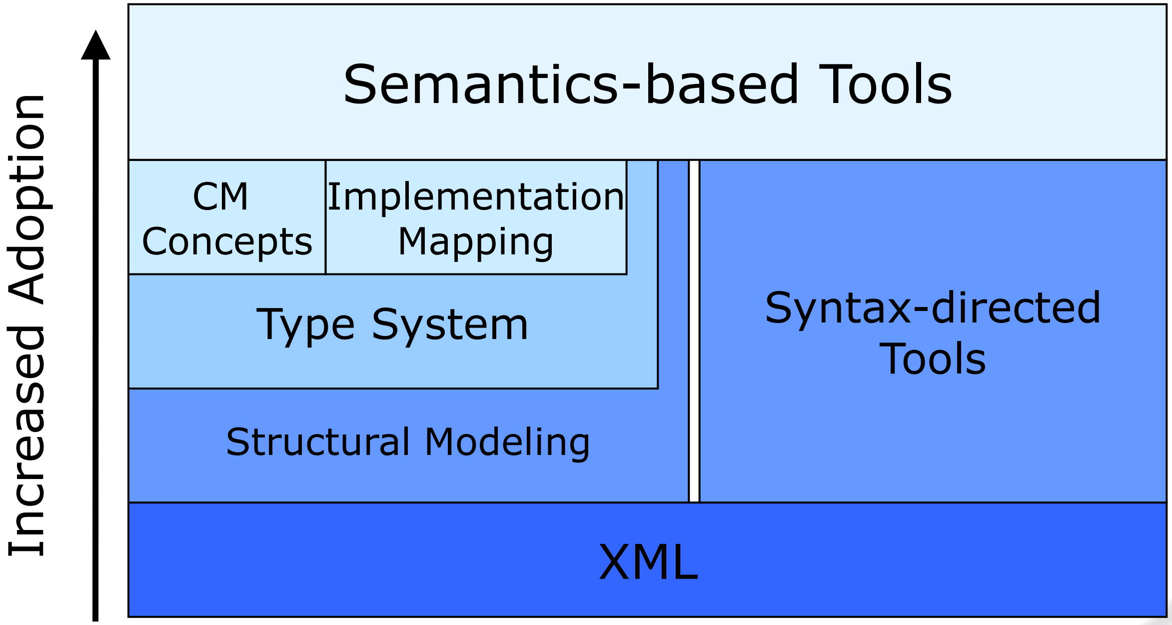
Strategies, Caveats, and Personal Reflections

- Given all these constraints, the interests of architecture research, and the state of architecture in industry, what can we do to make collaborative projects successful?
  - *And what are some things to watch out for?*

- Four key strategies
- A few example projects
  - *What worked, what didn’t, and why?*

Key #1: Start Small
Academic/Research Perspective

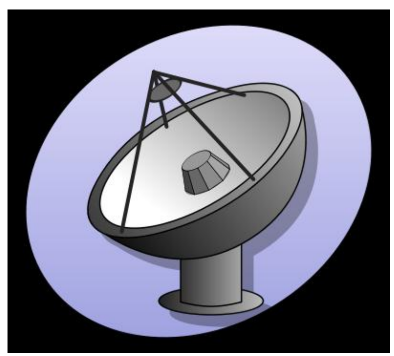
- All-or-nothing solutions are hard to adopt
  - *Industrial partners will work in an environment where they are highly constrained by numerous factors*
    - Contractual obligations, momentum, skill mix, corporate culture
- For any new approach, a step-by-step **adoption plan** is critical to success



Key #1: Start Small

Industry Perspective

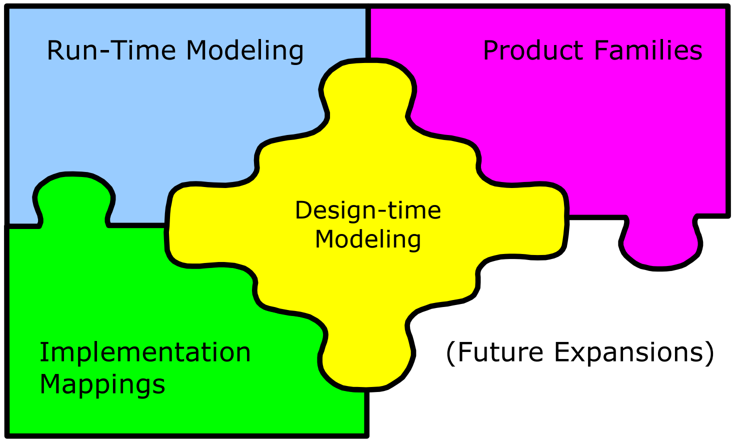
- All-or-nothing problem domains are hard to learn and get involved with
- Applying new technologies or techniques to complete, huge systems over multiple dimensions is infeasible
- The full details of huge systems are often proprietary

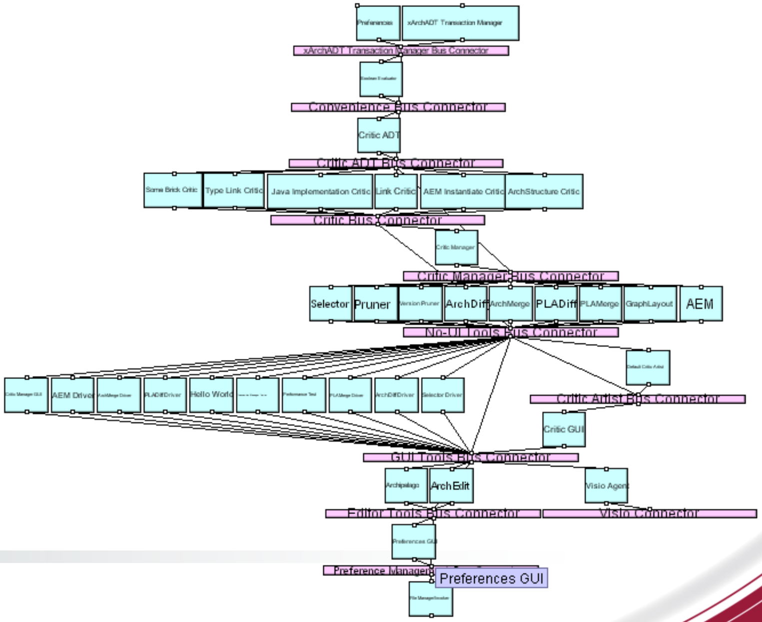


Key #2: Grow Big with Extensibility

Academic/Research Perspective

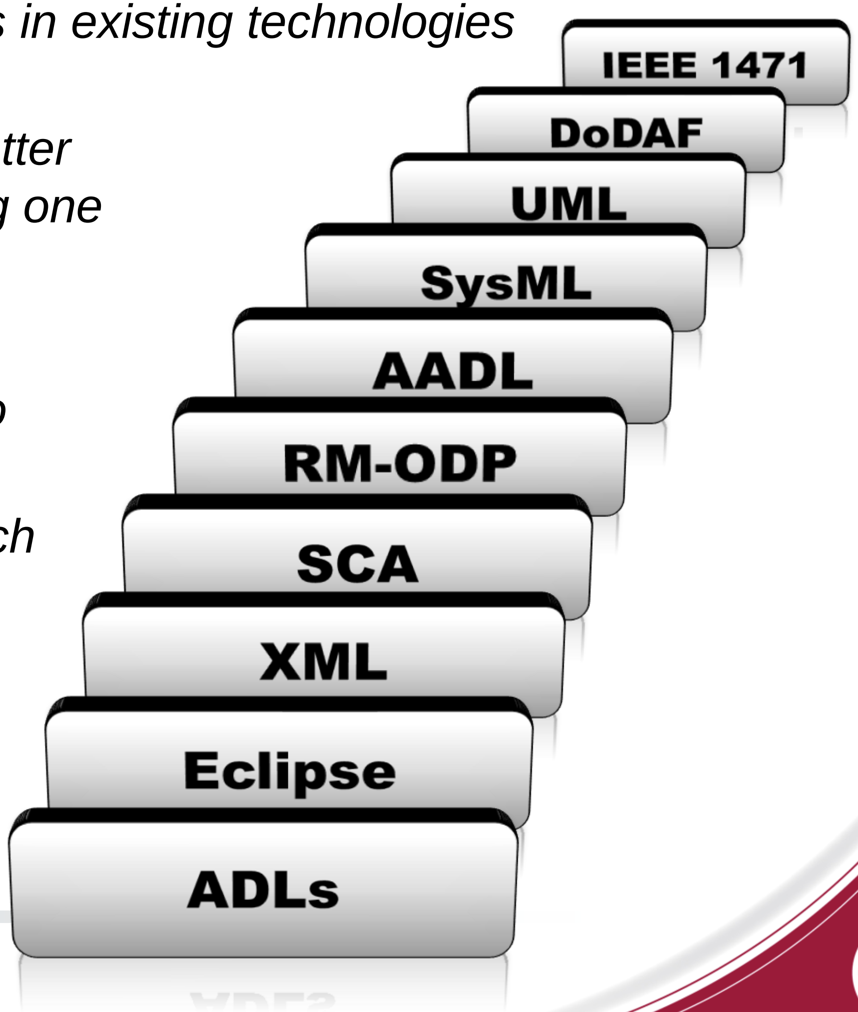
- Many projects are not built with a plan for extensibility in mind
- Extensibility mechanisms are difficult to “bolt on”—they must be “built in”
- Extensibility is especially helpful in industry collaborations to adapt to project-specific needs or incorporate project-specific components
- Architects should focus on extensibility naturally





Key #3: Play Nice with Other Kids

- Academic/Research Perspective
  - *Generally a goal to find deficiencies in existing technologies and techniques and remedy them*
  - *Compromise between building a better mousetrap and adapting an existing one*
- Industry Perspective
  - *Will continue using off-the-shelf technologies, but should be open to augmenting them with research*
  - *With the understanding that research techniques and tools (especially) will not have evolved to the point of some of these well-seated standards*



Key #4: Incremental Value

- The ideal area is one that
  - *Contains lots of interesting, “big” research questions*
  - *Is valuable to practitioners in the short-term*
  - *Is important to practitioners in the long-term*
  - *Provides incremental opportunities for success*

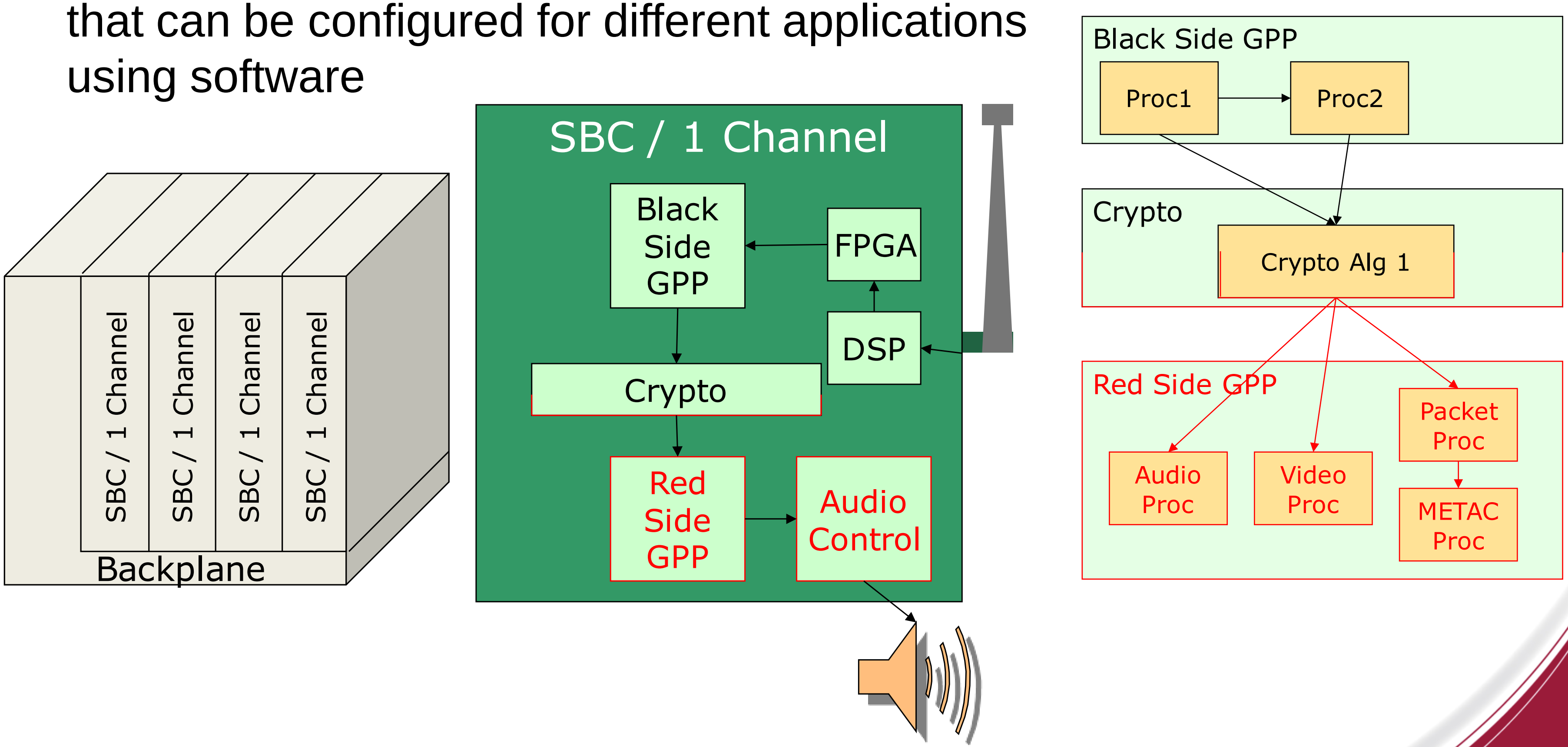
- Look for problems that
  - *Can be understood by students/researchers with a modicum of effort*
  - *Align with research interests of students and have identifiable research questions*
    - Are suitable for inclusion in the “evaluation” section of a paper/thesis
  - *Are nearby, but not necessarily on, the critical path*
  - *Provide an obvious “value-add” story for the practitioners*
    - Establish periodic milestones with incremental value at each

Key #C: The Cross-Cutting Key

- Collaborations are ultimately about people
- Ultimately there needs to be a point of contact/primary person responsible for the collaboration on both sides
  - *Academic side*
    - Primary person responsible is likely a professor
    - POC may be a professor or student project lead
  - *Industry side*
    - Primary person responsible *must be* an evangelist for the collaboration
      - *Often a ‘tech fellow’ or similar*
    - POC may be a member of the project on the industry side

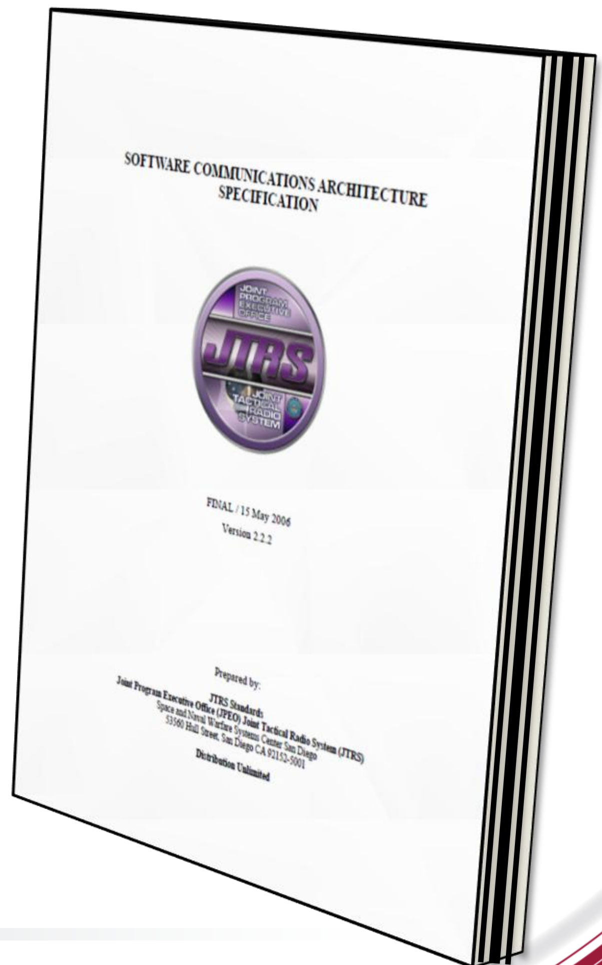
A Typical Project: Software-Defined Radio

- Large industrial project to develop a software-defined radio
- A general-purpose platform for digital communications that can be configured for different applications using software



Software-Defined Radio Project

- Governing specification was the Software Communications Architecture (SCA)
  - *http://sca.jpeojtrs.mil/*
- How architectural is the SCA?
  - *Based on CORBA and POSIX standards*
  - *Primarily defines APIs (CORBA IDL interfaces) that are found in common components*
    - Filesystem, resource manager, device driver, application main component…
  - *Does not provide specific guidance for how to define or connect components to make a software-defined radio*


SOFTWARE COMMUNICATIONS ARCHITECTURE SPECIFICATION

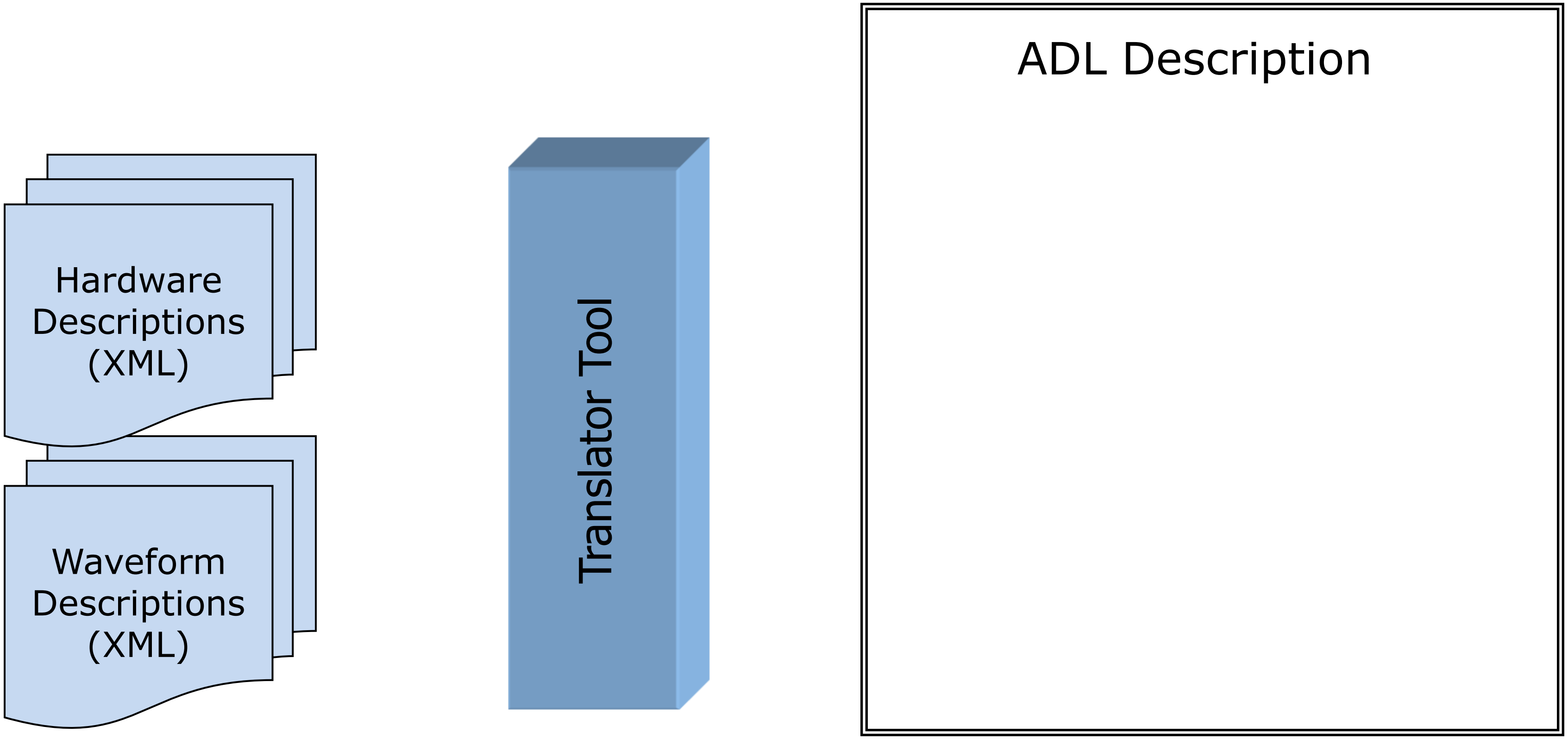
FINAL / 15 May 2006
Version 2.2.2

Prepared by:
JTRS Standards
Joint Program Executive Office (JPEO) Joint Tactical Radio System (JTRS)
Space and Naval Warfare Systems Center San Diego
53560 Hull Street, San Diego CA 92152-5001
Distribution Unlimited

Software Defined Radio Project: Artifacts

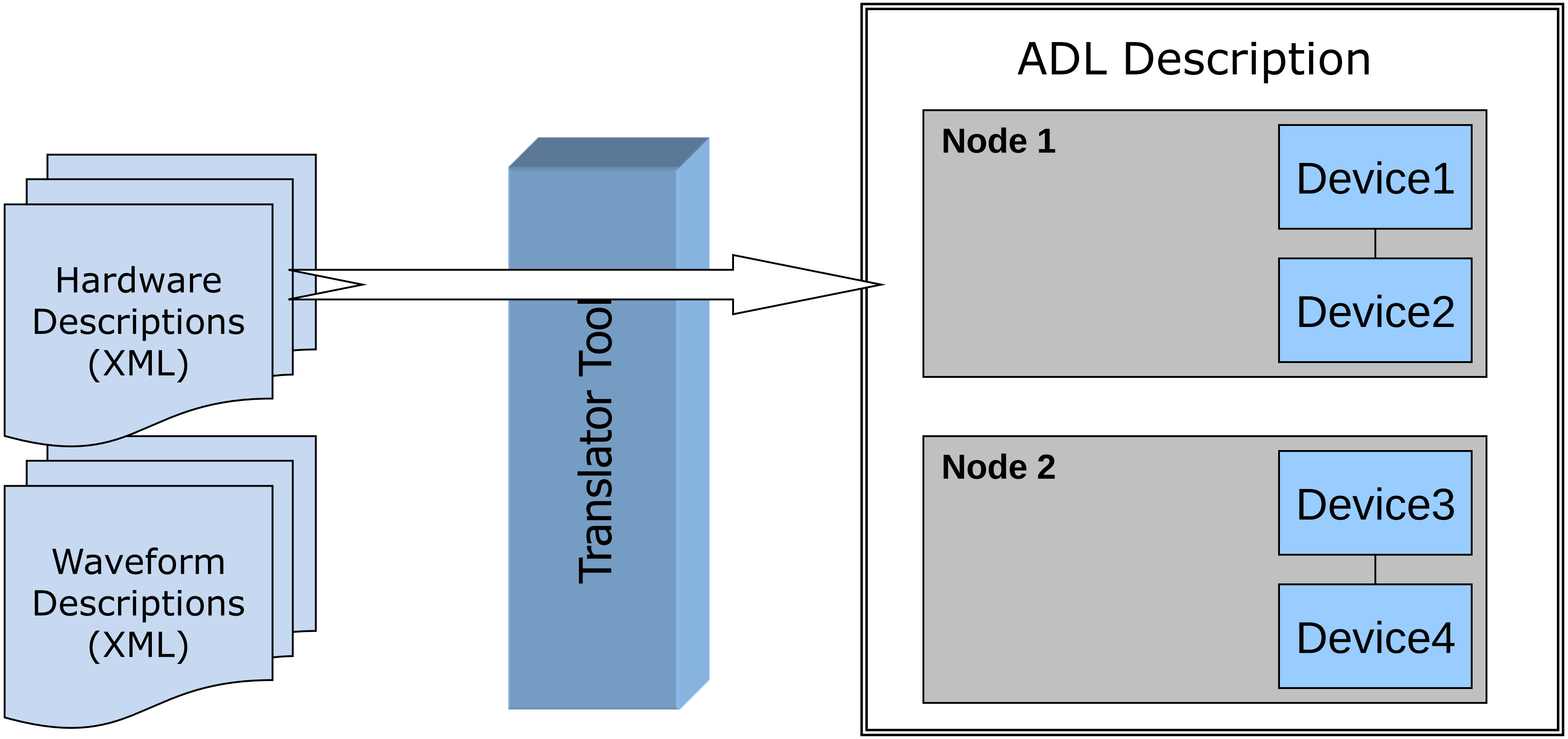
- Many Word and PowerPoint documents with diagrams
- Some UML
- SCA “Appendix D”-compliant XML files
  - *These define*
    - Components in operating environment or waveform
    - How they are implemented, and
    - How they are connected
  - *But*
    - Deployment of components on hosts is often ill-specified
    - Built by hand
    - Lack of visualizations
    - Lack of ability to analyze for even simple correctness
- They were doing architecture but saw it as implementation

First Collaborative Project



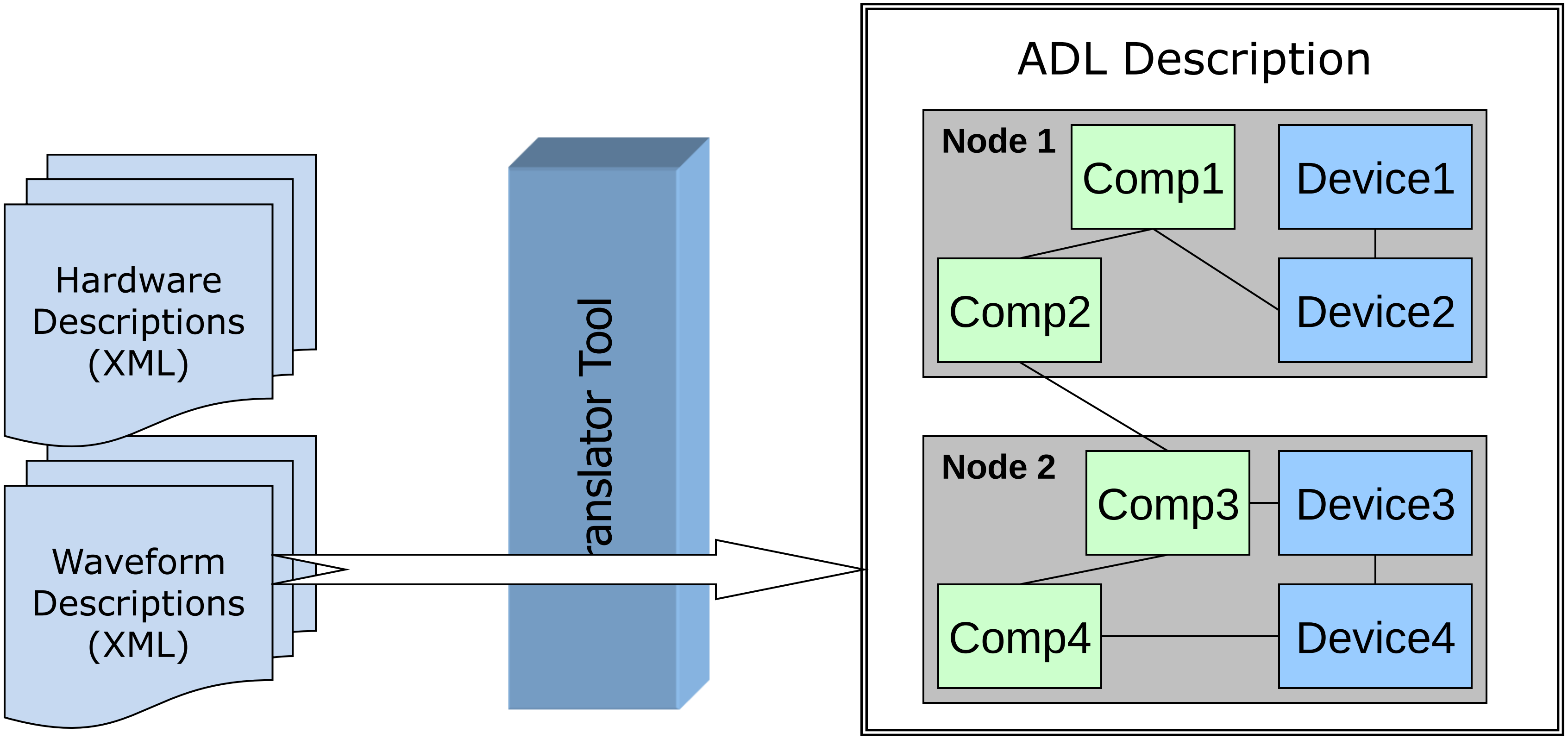
- Translate "SCA Appendix D" Domain Profiles to an ADL
- Use ADL-translated descriptions for visualization and analysis

First Collaborative Project



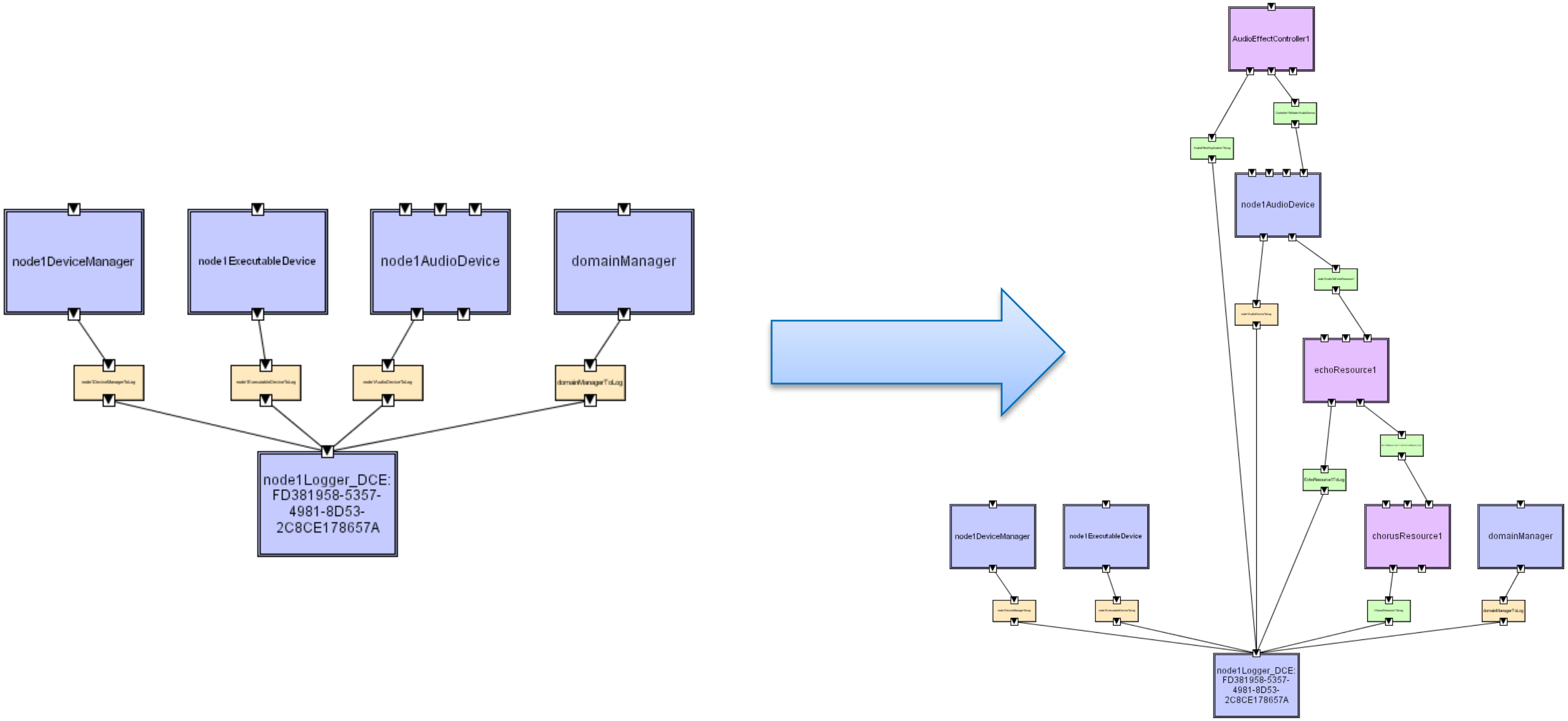
- Translate "SCA Appendix D" Domain Profiles to an ADL
- Use ADL-translated descriptions for visualization and analysis

First Collaborative Project



- Translate “SCA Appendix D” Domain Profiles to an ADL
- Use ADL-translated descriptions for visualization and analysis

First Collaborative Project



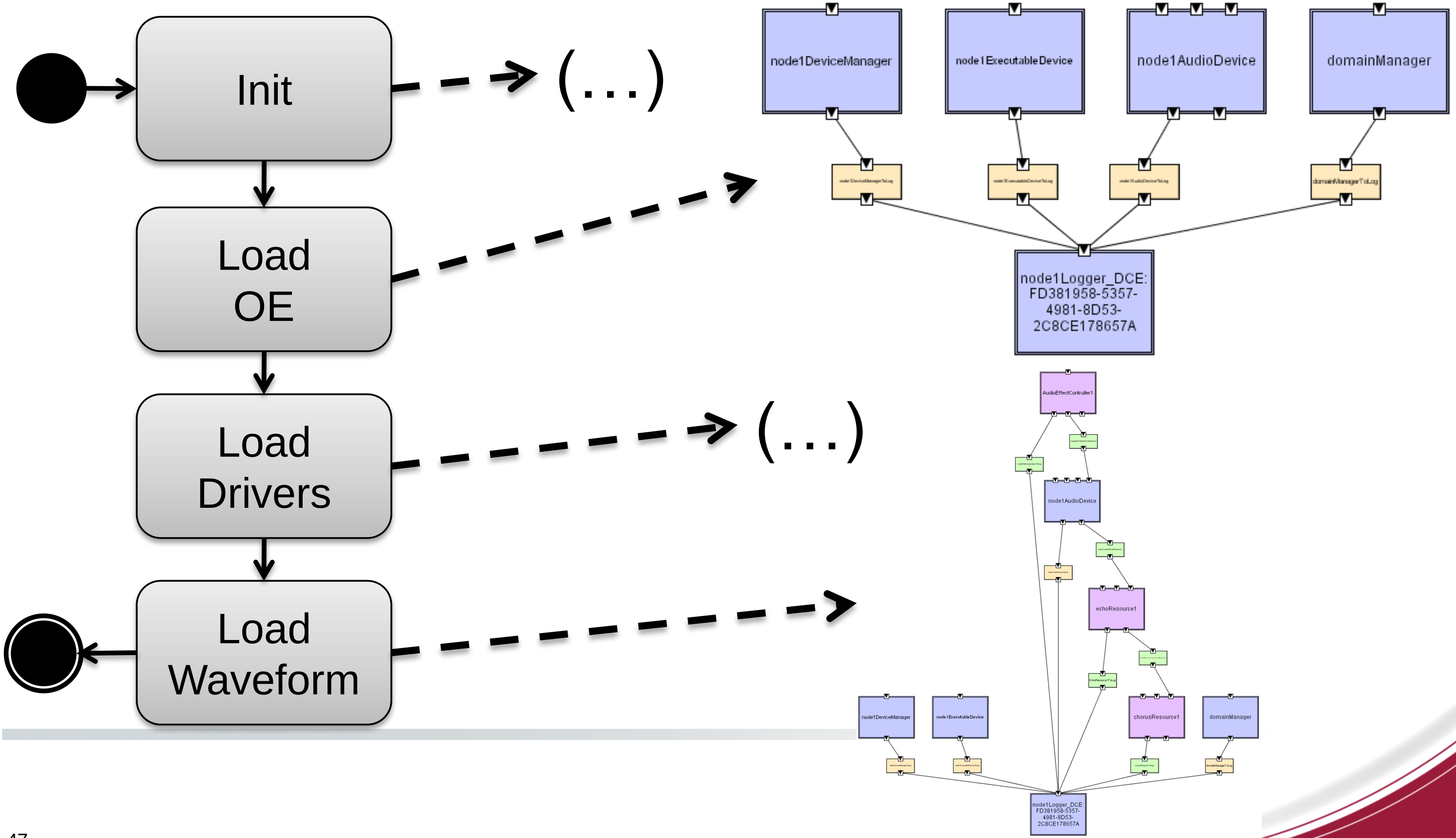
- Translate “SCA Appendix D” Domain Profiles to an ADL
- Use ADL-translated descriptions for visualization and analysis

Evaluation of First Collaboration

- What Worked
  - ***Start Small:*** *start with basic structural modeling; ignore temporal aspects and behavior for now*
  - ***Grow Big:*** *needed to add support for "Appendix D" files to ADL*
  - ***Play Nice:*** *Translator tool took advantage of ADL libraries, but also integrated ADL with existing project standard*
  - ***Incrementality:*** *Provided additional views that added value but were off the critical path*
  - *Found reference model to work on open data offsite*
  - *Fostered use on another project by practitioners exclusively*
- What Didn't
  - *Lots of uncertainty at the beginning before finding the problem*
  - *Strict constraint: can't "fix" practitioner files, must process with errors if possible*
  - *Practitioner-only project had much more limited data exchange possible*

Second Collaborative Project

- Practitioner had little insight into temporal aspects of system
  - *Not all components are present at all times*
  - *Map statecharts to configurations using product-line tools*



Evaluation of Second Collaboration

- What Worked
  - ***Start Small:*** *Use basis from first collaboration and add on*
  - ***Grow Big:*** *Incorporate statecharts and product-line capabilities*
  - ***Play Nice:*** *Still use "Appendix D" files; incorporate UML statecharts*
  - ***Incrementality:*** *Provided more views that added value but were off the critical path*
- What Didn't
  - *Basic statecharts got reimplemented in the ADL's environment rather than incorporating off-the-shelf models*
  - *Results (in both this and first collaboration) were primarily used as additional documentation artifacts*

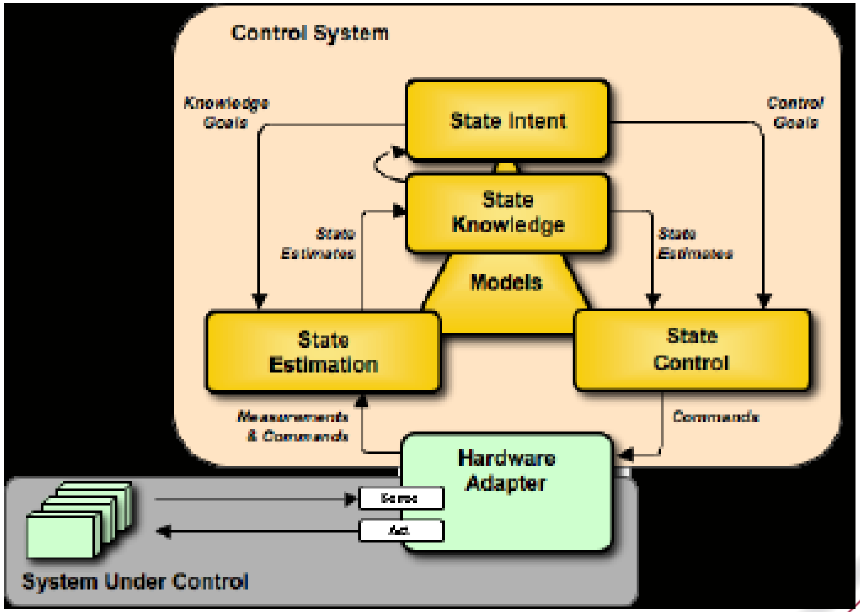
Evaluation of SDR Project

- Value for Academia
  - *Got to evaluate ADL and tool set in real-world domain*
  - *Increased knowledge about a new domain*

- Value for Industry
  - *Generated first graphical visualizations of configurations*
    - Provide “visceral insight” into configurations
  - *Generated novel temporal views that inspired some reconsideration on designers’ part*
  - *Enumerated errors and inconsistencies in configurations*
  - *Learn about new techniques*

Another Project: Mission Data System

- Practitioners defining an architectural style and model-driven approach that tied architectural models to implementation
- Wanted to use research ADL as the basis for models
- Generated lots of interesting discussion but no full collaboration
  - *No funding match ever made*
  - *Proprietary info constraints made informal collaboration difficult*
  - *Practitioners incorporated concepts/ideas but did not adopt technology*

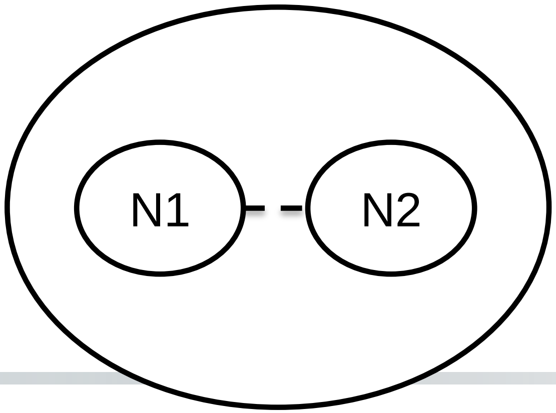


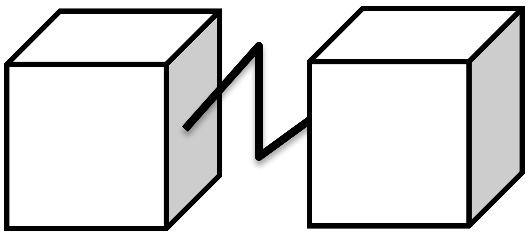
Evaluation of Mission Data System Collaboration

- What Worked
  - ***Start Small:*** *Focus on structural modeling for one particular domain*
  - ***Grow Big:*** *Extensibility mechanisms attracted practitioners*
  - *Willingness of both sides to informally collaborate without a specific mandate*
    - Required some extra flexibility on practitioner side
- What Didn’t
  - ***Play Nice:*** *Possibly too much novel technology here*
  - ***Incrementality:*** *Project would have been on the critical path; likely even with funding would have been difficult for it to keep pace with practitioner needs*
  - *Artifacts being generated by practitioner were considered “proprietary;” without funded mandate it was difficult for researchers to get access or publish about them*

Third Project: Tools for Reference Architecture for Space Data System (RASDS)

- Practitioners defining a domain-specific modeling notation for space data systems based on RM-ODP
- Wanted a toolset for modeling, visualization, and analysis
- Capabilities matched research environment well
  - *Various types of components and connectors, links, hierarchical composition*
- Sticking point: what should the semantics be?
  - *Researchers thought practitioners should define and vice-versa*



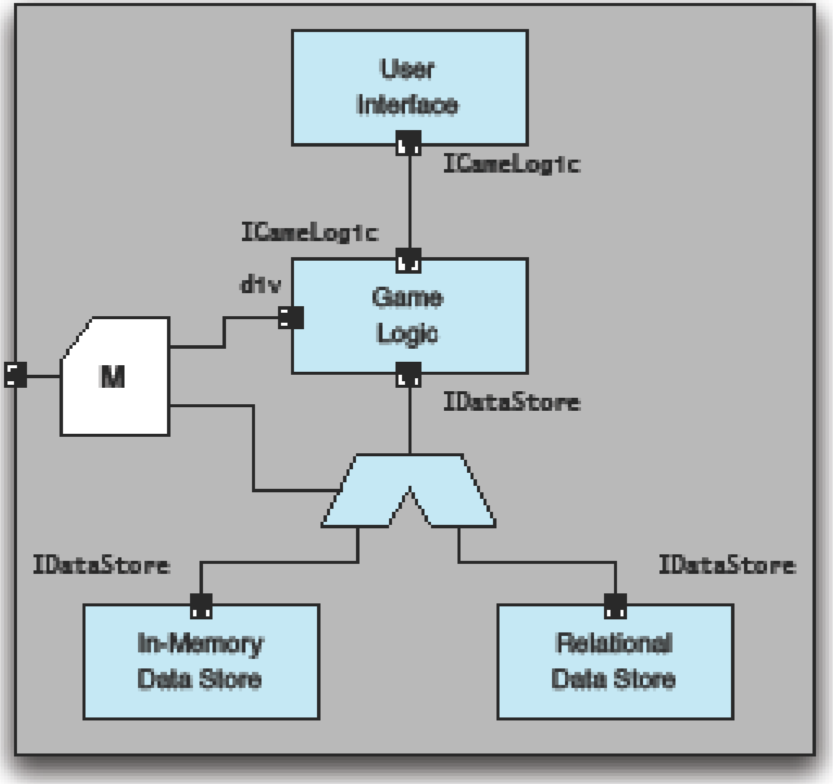


Evaluation of Mission Data System Collaboration

- What Worked
  - ***Grow Big:*** *Extensibility mechanisms attracted practitioners*
  - ***Play Nice:*** *Reuse of RM-ODP concepts*
  - *All relevant artifacts were available in the open; no restrictions*

- What Didn't
  - ***Start Small:*** *Notational mapping and tool development worked OK, but was not enough to "prime the pump" without semantics*
  - ***Incrementality:*** *Level of funding and collaboration was not enough for ambitious environment development; practitioners eventually fell back to UML profile*
  - *Not enough access to domain experts; not enough communication*

An Atypical Project: Koala



- Complete marriage of architecture research and industrial practice
- Spearheaded by Rob van Ommering at Philips Electronics in Eindhoven, Netherlands
- A form of model-driven architecture
  - *Models are both documentation and implementation artifacts*
  - *Each element of the model maps to some implementation construct*
  - *Use of product-line techniques (explicit options, variants) in the architecture to generate product architectures for a family of consumer electronics devices*

Why Does Koala Work so Well?

- Not a ‘collaboration,’ per se:
  - *Researcher(s) ARE the practitioner(s)*
  - *Rob works at Philips but also attended academic conferences and was getting his Ph.D. at the same time (advisor: Jan Bosch, who is now both a professor and an architect at Intuit)*
  - *Major cultural shift at the company toward horizontal thinking*
  - *Straightforward to understand how the approach adds technical value*
    - Direct mapping to specific implementation techniques used to build software for embedded devices
    - Clear mapping between business goals and technical goals
      - *The business product line of consumer electronics is also a technological product line*

Another Atypical Project: AADL

The Architecture Analysis and Design Language

- Society of Automotive Engineers (SAE) standard
  - *Key personnel are strong research/practice mix*
  - *Extensive industry participation in the standardization process*
- A complex, domain-specific architecture description language tailored for:
  - *Specifying both software and hardware/system elements*
  - *Specifying systems with embedded or real-time constraints*
- Coordinate with UML through UML 2.0 profile
- High-complexity but high-value
- Supported by an open-source environment (OSATE) and several novel analysis tools
- Extensibility through a mechanism called “annexes” which add sub-notations to the core
- Diverse array of projects underway in different areas

Why is AADL so successful?

- Big “human surface” between researchers/academia and practitioners/industry
  - *Cross-cutting Key #C writ large*
  - *Lots of communication bandwidth and buy-in*
  - *The key personnel are also domain experts*
- Focus on a specific domain and set of problems
  - *Very much not a ‘least common denominator’ standard*
- “Enough” extensibility through annex mechanism
  - *Allows external participants, explorations into other domains slightly off the critical path*
- Has hit “critical mass” so users are evangelizing in many places
  - *Multiple major funding streams now established*

Strategies in the Classroom

How can we prepare students for industry collaborations?

- Teach students about common standards
  - *Make scope clear: what are they good for, what aren't they?*
- 'Capstone' project courses that involve industry teams are very useful
  - *Keys to success: keep practitioner expectations realistic; keep both sides engaged with each other*
- Increase focus on "reading," not "writing"
  - *Few industry projects are 'green field' development*
  - *Students/academics will inevitably be coming in 'in medias res'*
  - *Assignments that focus on architectural recovery, reading and understanding standards or industrial specifications, etc.*
- Use the classroom as an opportunity to network and 'socialize'
  - *Invite practitioners to present (but give them time to get material cleared!)*
  - *Host one-day workshops and invite industry*

Conclusions

- Differing goals will always cause tension between academia and industry when collaborating
- Successful collaborations can be achieved if goals are kept in mind and expectations are realistic
- Keys to success
  - *Start small*
  - *Grow big*
  - *Play nice*
  - *Add value incrementally*
- Person-to-person communication, mutual understanding, and evangelism is the backbone of collaboration
- Successful collaboration is an art – train students early!

References

- Material on standards adapted from Chapter 16 of *Software Architecture: Foundations, Theory, and Practice*; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; John Wiley & Sons, Inc., 2008.
- Software Communications Architecture
  - *http://sca.jpeojtrs.mil/*
- Mission Data System
  - *http://mds.jpl.nasa.gov/public/index.shtml*
  - *See also http://trs-new.jpl.nasa.gov/dspace/bitstream/2014/8580/1/02-1118.pdf*
- Consultative Committee for Space Data Systems (CCSDS) Reference Architecture for Space Data Systems (RASDS)
  - *http://trs-new.jpl.nasa.gov/dspace/bitstream/2014/7485/1/03-1372.pdf*
- The Koala Component Model for Consumer Electronics Software
  - *http://portal.acm.org/citation.cfm?id=621436*
- The SAE Architecture Analysis and Design Language (AADL) Standard
  - *http://www.aadl.info/*

References (cont.)

- IEEE Std 1471-2000 IEEE Recommended Practice for Architectural Description of Software-Intensive Systems -Description
  - *http://standards.ieee.org/reading/ieee/std_public/description/se/1471-2000_desc.html*
- The Unified Modeling Language (UML)
  - *http://www.uml.org/*
- The Department of Defense Architecture Framework (DoDAF)
  - *http://cio-nii.defense.gov/docs/DoDAF_volume_I.pdf*
  - *http://cio-nii.defense.gov/docs/DoDAF_Volume_II.pdf*
- SysML
  - *http://www.sysml.org/*
- Reference Model of Open Distributed Processing (RM-ODP)
  - *http://www.rm-odp.net/*

Backup Charts and Credits

Questions for Discussion

- What collaboration stories do you have?
  - *What worked, and what didn't for you?*
  - *What were the "make it or break it" factors?*
  - *What parts of the collaborations were the most frustrating?*
- How do researchers perceive industrial software architecture?
- How do researchers perceive popular standards?
- What are the biggest gaps in understanding between students, researchers, and practitioners?
- How can you find or cultivate industrial evangelists in projects?
- Are there any good strategies for making the "proprietary info" constraint easier to deal with?
- How can designated academic/industry affiliates be used most effectively?

Photo Credits

- By DanMS, based on a public-domain map; released under GFDL

- Image in the public domain; copyright has expired

- Released under CC-BY-SA-2.5 license by user Americasroof; it is incorporated unmodified here and as such does not create a new derivative work.

Photo Credits (continued)

- Released under CC-BY-2.5 license
by user Xtrememachineuk

- The SCA is a U.S. Government document approved for unlimited distribution.

- All UML, DoDAF, RM-ODP, SysML & Koala diagrams from *Software Architecture: Foundations, Theory, and Practice*; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Used with permission.

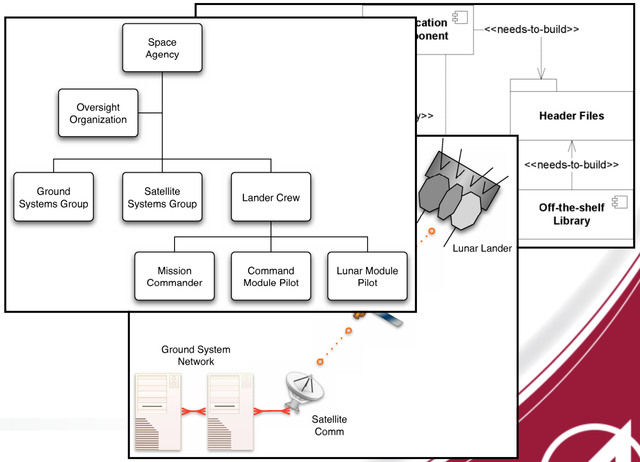
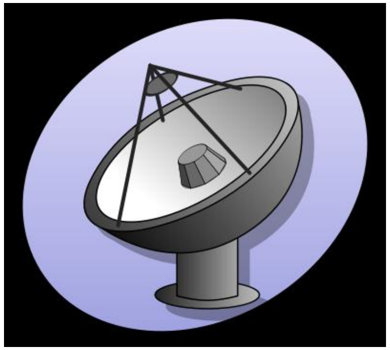


Photo Credits (continued)

- Released under CC-Attribution license by Thomas Duesing

- Released into the public domain by user Booyabazooka



- Released into the public domain because it is a NASA image

Photo Credits (continued)

- Used in accordance with JPL image use policy. Image courtesy NASA/JPL-Caltech.