The Use of Influence Maps to Understand System-of-Systems Programmatics

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Jim Smith
April 21, 2009

Purpose

Introduce an elicitation and diagnostic technique—influence maps (IMs)— and a formal reasoning framework, and show how they provide useful insights into cross-program and inter-organizational programmatic conflicts in systems of systems.

Overview

Introduction

Background

- The problem--elaborated
- Brief introduction to modal logics

Discussion

- System-of-systems programmatics
  - “Laws” of programmatics
  - Reasoning Framework
- *Extremely* simple example

Conclusions

Future work

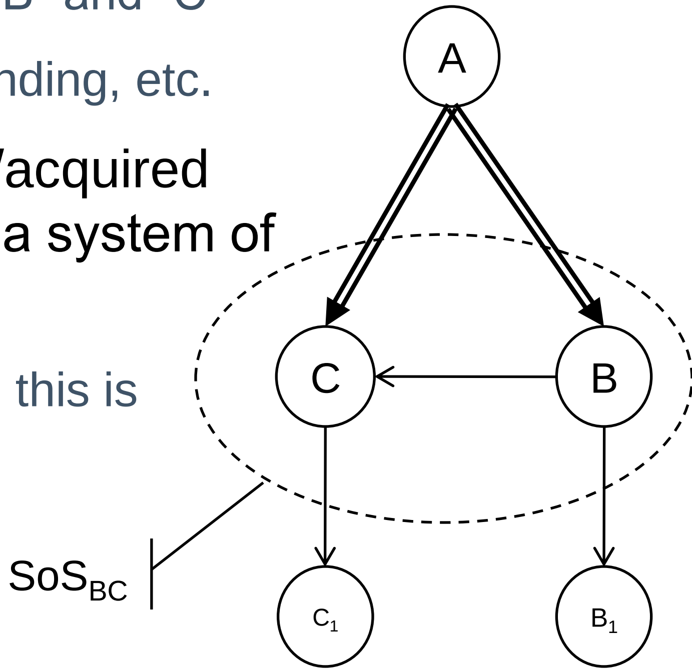
The Problem 1

Two PMs (“B” and “C”), reporting to a PEO (“A”)

- Each PMO has one development contractor (“B_1” and “C_1,” respectively)
- There is a giver-receiver relationship between “B” and “C”
- Each PMO has its own requirements, users, funding, etc.

The interactions of the systems being developed/acquired by PMOs “B” and “C” result in the emergence of a system of systems (“SoS_{BC}”)

- In the DoD systems engineering guide for SoS, this is an “acknowledged” SoS



The Problem $_2$

System “B” has experienced a funding cut

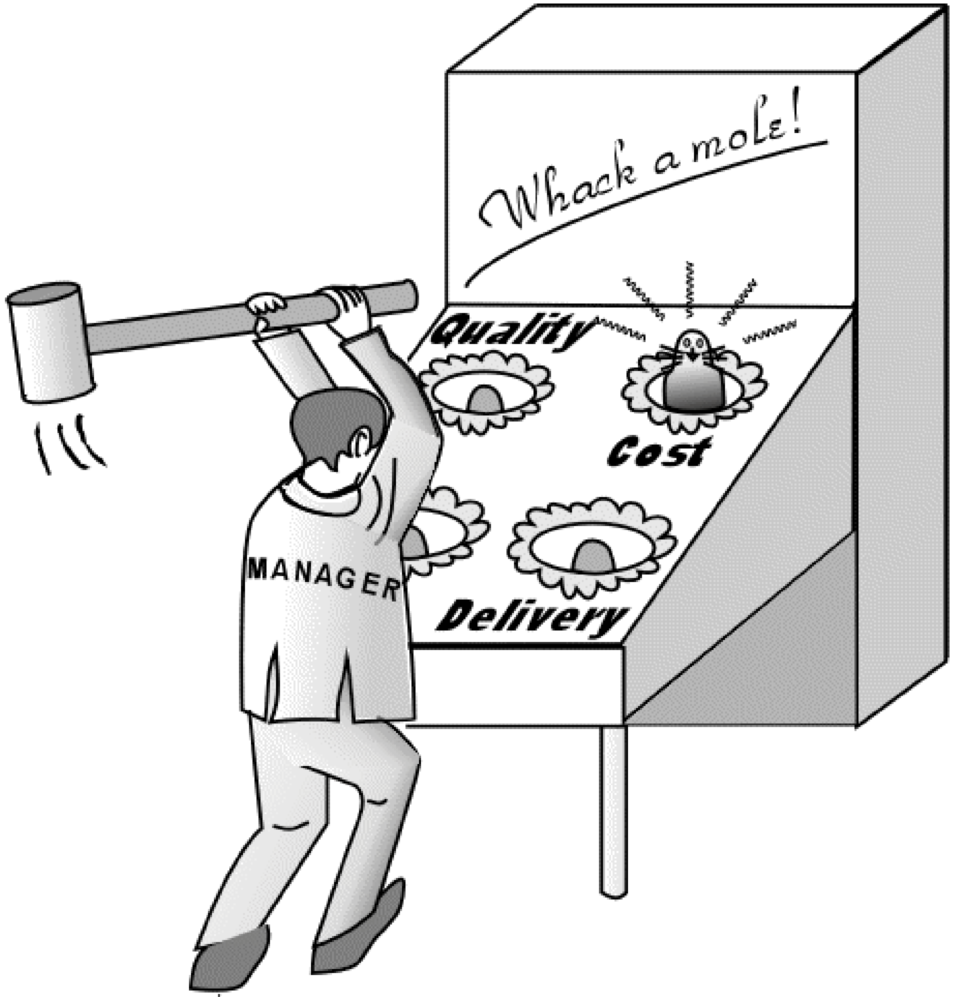
- PMO “B” is slightly behind schedule
- There are some requirements for system “B” that are non-critical for PMO B (and its associated users)
- One obvious approach for PMO “B” would be to delete some non-critical requirements (to reduce total development costs, and possibly gain some schedule relief)

However … Those “non-critical” requirements are essential for PMO “C” and SoS_{BC}

- Deferring or deleting these requirements would cause PMO “C” and/or SoS_{BC} to fail to satisfy their schedule and performance goals

So… Whaddya do if you’re PMO “B”?

One Possible Approach



A Better Approach

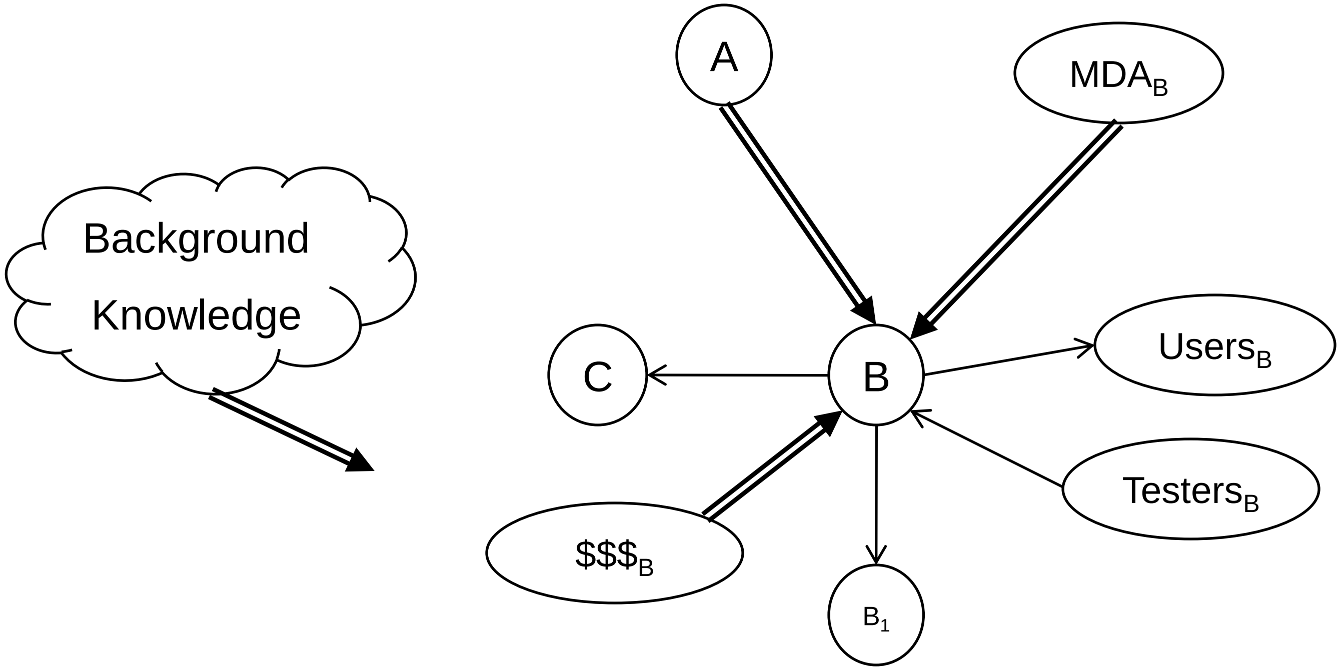
Instead of “Program Manager Whack A Mole,” base decisions on

- Knowledge of the relevant interrelationships between constituents within a system-of-systems context
- An understanding of the origins and nature of the underlying legal and moral imperatives that act as constraints on a program manager
- Use of a formal reasoning framework that allows one to weigh competing constraints to arrive at a satisficing solution—if one is possible

Interrelationships 1

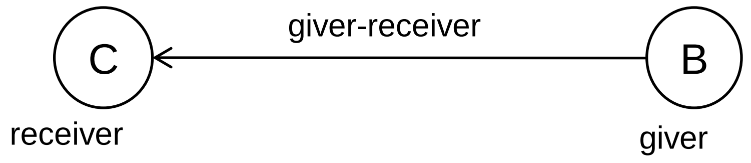
To understand the origins of the conflicts between the individual program (PMO "B") and the SoS ("SOS$_{BC}$"), as well as reason a way through to a satisficing solution, additional information is needed

PMO "B" sees the following:



Interrelationships $_2$

At the level of the giver-receiver agreement between PMOs “B” and “C,” we have



PMO “B” obligations to PMO “C”

- Provide agreed-on capabilities in system “B” on agreed-upon schedule
  - Satisfying quality attributes, with adequate confidence, etc.
- Not act in a way to violate agreement

PMO “C” obligations to PMO “B”

- Accept system “B” if it meets agreed-upon schedule and capabilities
- Not act in a way to violate agreement (e.g., desired schedule, capabilities)

Interrelationships $_3$

There are similar pictures for PMO “C” and PEO “A”

However, there can be some important differences between these views

- MDA_B might be different than MDA_C
  - Different priorities, different perceptions of problems and solutions
- Different users, testers, funding, etc. for PMO “C”

There is no “owner” for the acknowledged SoS (“SoS_{BC}”)

- *Maybe* some explicit funding, testing, etc. for SoS_{BC} … but probably not
- No explicit authority to make cross-program tradeoffs

Philosophy, Modal Logics, Legal Theory, etc. 1

Problems of programmatics in systems of systems (or in individual systems) do not lend themselves to solutions based on propositional logic and predicate calculus, or conventional quantitative methods (e.g., linear programming, multi-criteria decision-making) because of:

- Conflicting normative obligations (i.e., do $A \cap \neg A$, $B \cap \neg B$, etc.)
- Socio-economic influences
- Other legal and moral imperatives

Modal logics provide a way to reason about issues that are not simply “black and white”:

- Belief (doxastic logic)
- Time (temporal logic)
- Knowledge (epistemic logic)
- Necessity (alethic logic)
- Obligation (deontic logic)

Philosophy, Modal Logics, Legal Theory, etc. 2

Modal logics widely used in:

- Modern philosophy/metaphysics
  - Kripke semantics and “possible worlds” models
  - Speech acts
- Legal theory
  - Side effects/unintended consequences of legislation
  - Reasoning about precedent
- System science/computer science
  - Describe normative behaviors of systems

And … proposed for use in system-of-systems programmatics

- “All of the above”

“Laws of Programmatics” 1

One possible set of normative obligations for a program manager includes:

- Maximize satisfaction of cost, schedule, and performance requirements
- Obey all applicable laws, regulations, directives, etc.
- Take appropriate mitigation steps when problems occur
- Be creative – don’t rely on “textbook” solutions

Taking the first imperative (*maximize satisfaction of cost, schedule, and performance requirements),* we can express this as:

$$\Phi : the\ set\ of\ all\ possible\ assertions, where\ \varphi_i \in \Phi$$

$$\mathbb{A} : the\ set\ of\ all\ possible\ actions, where\ \alpha \in \mathbb{A}$$

$$\varphi_0 \vdash max\ \ cost, schedule, performance$$

$$\bigcirc_0 \left[\alpha \rightarrow \varphi_0 \ \& \neg\ \ \alpha \rightarrow \neg \varphi_0 \ \ \right]$$

“Laws of Programmatics” $_2$

From the initial minimal set of normative obligations, we can define a set of “laws,” or meta-behaviors, for any program manager:

> **Zeroth law**: An actor or constituent (hereinafter referred-to as actor) shall act in such a way so as to maximize satisfaction of applicable cost, schedule, and performance goals for the system.
>
> **First law**: An actor shall comply with all applicable laws, regulations, directives, policies, etc. issued by “competent authority” except when doing so would conflict with the zeroth or first laws.
>
> **Second law**: An actor shall take corrective actions to remedy any risks, issues, problems etc. in their program except when such actions conflict with the zeroth, first, or second laws.
>
> **Third law**: An actor may “be creative” as long as their actions do not conflict with any higher law.

As was previously shown for the zeroth law, these can be expressed in deontic form

"Laws of Programmatics" 3

By themselves, the laws are insufficient. Other general knowledge and background information is necessary to perform any meaningful analysis. For example:

$$\varphi_{ADA} \vdash obey\_anti-deficiency\_act$$

$$\bigcirc_{ADA}\left[\alpha \rightarrow \varphi_{ADA} \,\&\, \neg \;\; \alpha \rightarrow \neg\varphi_{ADA} \;\;\right]$$

$$!(\bigcirc_{ADA} / appropriated\_\ funds)$$

$$\varphi_{MA} \vdash obey\_misappropriation\_act$$

$$\bigcirc_{MA}\left[\alpha \rightarrow \varphi_{MA} \,\&\, \neg \;\; \alpha \rightarrow \neg\varphi_{MA} \;\;\right]$$

$$!(\bigcirc_{MA} / appropriated\_\ funds)$$

The truth of the assertion of the Anti-Deficiency Act requires that a program obey the ADA

The ADA levies an obligation on a program to act so as to preserve the assertion, and not—by failing to act—to allow the assertion to be falsified

The precondition—that a program uses appropriated funds—entails the consequent that the program must comply with the obligation to preserve the assertion

“Laws of Programmatics” 4

Let’s use these to examine a typical problem from a system-centric perspective:

- Factors from the laws and general, background knowledge include:

$$f_5 : take\_appropriate\_mitigation$$

$$f_6 : be\_creative$$

$$f_7 : obey\_applicable\_laws$$

$$f_8 : has\_non-critical\_requirements$$

$$f_9 : has\_schedule\_slack$$

$$f_{10} : delete\_requirements\_to\_reduce\_costs$$

$$f_{11} : delay\_schedule\_to\_reduce\_costs$$

$$f_{12} : funds\_cut$$

$$f_{13} : must\_reduce\_costs$$

"Laws of Programmatics" 5

- Derived constituent conditional imperatives include:

$c_4 = \langle f_{12}, f_{13} \rangle$: *must reduce costs if funds cut*

$c_5 = \langle f_{13}, f_8, f_{10} \rangle$: *if required to reduce costs, and has non-critical requirements, then delete non-critical requirements*

$c_6 = \langle f_{13}, f_9, f_{11} \rangle$: *if required to reduce costs, and has schedule slack, then delay schedule*

$c_7 = \langle f_1, f_5 \rangle$: *programs must take appropriate mitigation steps*

$c_8 = \langle f_1, f_6 \rangle$: *programs must be creative*

$c_9 = \langle f_1, f_7 \rangle$: *programs must obey all applicable laws*

Reasoning Framework $_1$

Conditional imperatives define actions that must be taken (consequent) if the necessary precondition (antecedent) is true:

$$!(\bigcirc_0 / \mathit{progam\_manager})$$

which is read “if you are a program manager, then you must fulfill obligation $\bigcirc_0$”

Precedents are a subset of conditional imperatives that are applicable to the instant case. A conditional imperative is applicable if it has one or more antecedent factors in common with the instant case.

$$Ant(i) = \text{ program, program\_manager}$$

Precedents can be more-or-less “on point,” and can be partially-ordered for a particular system-of-systems context

- Precedents that are more specific, and have more factors in common with the instant case are, in general, more on point than more general precedents

Reasoning Framework $_2$

Precedents can trump other precedents (even if they are more general)

- Obligation to obey 31 U.S.C. § 1517 (Anti-Deficiency Act) trumps obligation to maximize cost, schedule, and performance goals

$$\bigcirc_0 <_X \; obey\_ada$$

To recap:

- The “laws” (expressed as a series of imperatives) can be represented (using deontic logic) as statements of obligation
- Conditional imperatives give rise to precedents
- Precedents can be ordered or trumped

Reasoning Framework $_3$

Application of the laws requires a reasoning framework

- Provides rules to reason about a particular situation in context through successive refinement, or elaboration to either include, or exclude, particular factors
- Permits determination that there is no constructible reasoning chain (i.e., no solution possible)

Reasoning chain begins with the deconstruction of conditional imperatives into factors (i.e., their consequents and antecedents)

$$f_1 : is\_a\_program$$

$$f_2 : uses\_appropriated\_funds$$

$$f_3 : must\_satisfy\_csp\_objectives$$

$$f_4 : must\_obey\_ada$$

Reasoning Framework $_4$

These factors are combined to form constituents of the set of conditional imperatives

$$c_1 = \left\langle \; f_1 \;\; , f_2 \right\rangle : \; a \; progam \, is \; funded \; by \; appropriated \;\; funds$$

$$c_2 = \left\langle \; f_1 \;\; , f_3 \right\rangle : \; a \; program \, must \; satisfy \; cost, schedule,$$

$$and \;\; performance \, objectives$$

$$c_3 = \left\langle \; f_2 \;\; , f_4 \right\rangle : \; appropriated \;\; funds \; are \; subject \; to \, the \; ADA$$

The relative rankings of the conditional imperatives are defined:

$$c_1 <_X \; c_2 <_X \; c_3$$

Reasoning Framework 5

To illustrate, take a (really, *really*) simple example

Suppose we have the current situation as described by:

$$X_0 = f_1 \quad : this\ is\ a\ program$$

and we want to ascertain if we are subject to the Anti Deficiency Act (i.e., can we use precedent to construct a reasoning chain to include factor f_4 in a refinement of X_0?)

Starting with the initial situation, X_0, we have $f_1 \xrightarrow[c_1]{} f_1, f_2$ which shows that conditional imperative c_1 (a program is funded by appropriated funds) supports the inclusion of factor f_2 (uses appropriated funds) as an elaboration of the initial situation.

Similarly, conditional imperative c_3 supports the inclusion of factor f_4, represented as $f_1, f_2 \xrightarrow[c_3]{} f_1, f_2, f_4$, establishing that the program is subject to the Anti Deficiency Act

Application of the Reasoning Framework 1

Initial problem statement (from original problem scenario):

> Program "*X*" is currently slightly behind schedule and under budget, but has just sustained a funding cut. Can the program delete non-critical (in DoD-ese, objective) requirements to reduce costs? Can it delay schedule to achieve the same results?

$$X_0 = \; f_1, f_8, \neg f_9, f_{12}$$

Can we construct a reasoning chain to include f_{10} *or* f_{11}*?*

Reasoning chain:

$$f_1, f_8, f_{12} \xrightarrow[c_1]{} f_1, f_2, f_8, f_{12} \xrightarrow[c_3]{} f_1, f_2, f_4, f_8, f_{12} \xrightarrow[c_4]{}$$

$$f_1, f_2, f_4, f_8, f_{12}, f_{13} \xrightarrow[c_5]{} f_1, f_2, f_4, f_8, \mathbf{f_{10}}, f_{12}, f_{13}$$

Conclusion: Yes, objective requirements can be deleted to reduce costs, but you cannot delay schedule

Application of the Reasoning Framework $_2$

What happens in a systems of systems (e.g., SoS_{BC})?

Two additional “laws” come into force:

> **Fourth law**: An actor shall not take unilateral action within their program to the detriment of another program, or through inaction, knowingly allow detriment to come to another program except when doing so would conflict with the fifth law.
>
> **Fifth law**: An actor shall not take unilateral action within their program to the detriment a system of systems, or through inaction, knowingly allow detriment to come to a system of systems.

We need to elaborate the initial situation to include factors germane to the system of systems:

$$f_{14} : in\_a\_sos$$

$$f_{15} : has\_requirement\_critical\_for\_sos$$

Application of the Reasoning Framework $_3$

The fourth and fifth laws give rise to two additional conditional imperative constituents

$$c_{10} \vdash \left\langle \; f_{14}, f_{15} \quad , \neg f_{10} \right\rangle : \; if \; in\, an\, SoS, may\, not$$
$$\text{"}injure\text{"}\; the\; SoS\; by\; deleting\; requirements\; that$$
$$are\; critical\; to\; the\; SoS$$
$$c_{11} \vdash \left\langle \; f_{14}, f_{16} \quad , \neg f_{11} \right\rangle : \; if \; in\, an\, SoS, may\, not$$
$$\text{"}injure\text{"}\; the\; SoS\; by\; delaying\; schedule$$

The revised initial situation is now described by:

$$X_0' = \quad f_1, f_8, \neg f_9, f_{12}, f_{14}, f_{15}$$
$$Can\; we\; construct\; a\; reasoning\; chain\; to\; include\; f_{10}\; or\; f_{11}?$$

Application of the Reasoning Framework 4

Revised reasoning chain:

$$f_1, f_8, f_{12}, f_{14}, f_{15} \xrightarrow[c_1]{} f_1, f_2, f_8, f_{12}\, f_{14}, f_{15} \xrightarrow[c_3]{}$$

$$f_1, f_2, f_4, f_8, f_{12}, f_{14}, f_{15} \xrightarrow[c_4]{}$$

$$f_1, f_2, f_4, f_8, f_{12}, f_{13}, f_{14}, f_{15} \xrightarrow[c_{10}]{} \varnothing$$

No constructible reasoning chain exists:

$$c_5 \to f_{10}$$

$$c_{10} \to \neg f_{10}$$

$$c_5 <_x c_{10}$$

Bottom line: The inclusion of program "X" in a system of systems context results in an inability to take local action to accommodate the funding cut

Conclusions

The programmatics of systems acquisition are complex … systems of systems even more so

To get beyond current “program management dart board,” or other heuristics-based approaches to decision-making in this complex environment requires:

1. A method for identifying conflicts between system-centric and system-of-systems contexts
2. Some formal methods to enable program managers to recognize potential solutions to seemingly intractable problems

Modal logics—and an accompanying reasoning framework—appear well-suited to this task

Future Work

This work is still fairly immature … several avenues are being explored, including:

- Incorporation of additional modalities (i.e., epistemic and doxastic) and temporal logic to deal with the “knowingly” aspects of the fourth and fifth laws, as well as changes over time
- Elaboration of “injure” in the context of a “theory of agreements”
- Incorporation of Ashley’s factor weighting and magnitude considerations to refine determination of precedent applicability

Currently applying this approach to a case study

- Synthesized from several customer engagements
- Underlying phenomena well understood; provides a good surrogate for “ground truth”

Some Recent Technical Reports

Meyers, C.; & Smith, J. "*Programmatic Interoperability*" (CMU/SEI-2008-TN-012)

Smith, J. "*Topics in Interoperability: Structural Programmatics in a System of Systems*" (CMU/SEI-2006-TN-037)

Smith, J.; & Phillips, D. "*Interoperable Acquisition for Systems of Systems: The Challenges*" (CMU/SEI-2006-TN-034)

Brownsword, L.; *et al.* "*System-of-Systems Navigator: An Approach for Managing System-of-Systems Interoperability*" (CMU/SEI-2006-TN-019)

Fisher, D. "*An Emergent Perspective on Interoperation in Systems of Systems*" (CMU/SEI-2006-TR-003)

Meyers, C.; Smith, J.; *et al.* "*Requirements Management in a System of Systems Context: A Workshop*" (CMU/SEI-2006-TN-015)

Smith, J.; & Meyers, C. "*Exploring Programmatic Interoperability: Army Future Force Workshop*" (CMU/SEI-2005-TN-042)

Including Interoperability in the
Requirements Management in a
Interoperable Acquisition for Systems of Systems: The Challenges

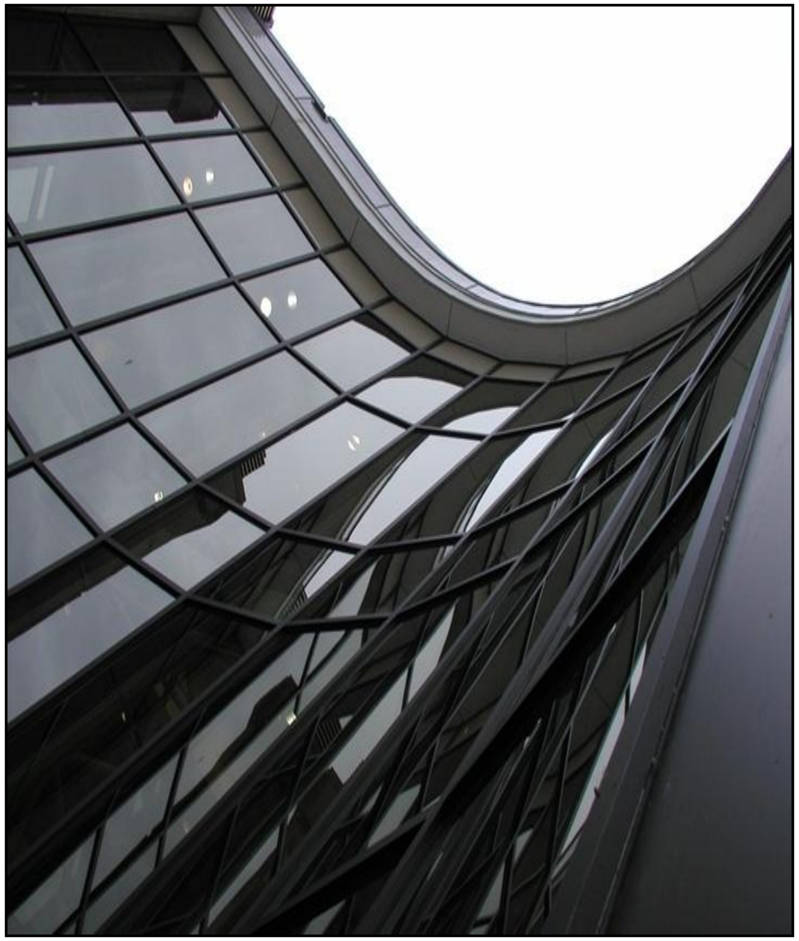
Contact Information

Jim Smith

(703) 908-8221

jds@sei.cmu.edu

http://www.sei.cmu.edu/staff/jds/



Software Engineering Institute | Carnegie Mellon