THE POWER OF ABSTRACTION,

OR,

A CASE FOR DOMAIN MODELING

Pamela Zave

AT&T Laboratories—Research

Bedminster, New Jersey USA
RESEARCHERS IN COMPUTER SCIENCE . . .

. . . ALL HAVE THE SAME PROBLEM

How can we persuade those who build large software systems to use what we produce?

Most of us are asking people to change their own process, not just handing them a product.

probably not the right role for research
THE POWER OF ABSTRACTION: OUTLINE

1 ONE WAY TO IMPROVE RESEARCH AND FACILITATE ITS USE

2 OVERCOMING THE OBSTACLES TO DOMAIN MODELING AS UNIVERSITY RESEARCH

I won’t be telling you anything you don’t already know, . . .

. . . but maybe I can reinforce a healthy trend and give you a few new examples.
LARGE SOFTWARE SYSTEMS IN THE REAL WORLD

financial services
healthcare services
aerospace systems
air traffic control
automotive systems
factory automation
retail sales
environmental monitoring
energy grids
communication networks

. . . and every other aspect of modern life

THE INTERFACES TO COMPUTER SCIENCE

programming languages
specification languages
schema and query languages
rule-based languages
machine learning
operating systems
networks

THERE IS A VERY LARGE GAP BETWEEN THEM, FILLED BY:
application code

BUT THIS IS NOT ENOUGH!
WE NEED . . .
requirements
specifications
architectures

. . . WHICH ARE DOMAIN MODELS
CONTENTS OF A FULL DOMAIN MODEL:

**DOMAIN KNOWLEDGE:**
description of the system’s environment

**ENVIRONMENT**

**SYSTEM**

**SPECIFICATION:**
description of the behavior of the system

**ARCHITECTURE:**
functions, modules, platforms, frameworks, performance constraints, etc.

**REQUIREMENTS:**
description of how the environment should work when the system is installed

- *all are based on coordinated abstractions and terminology*
- *all are re-usable artifacts, intended for a family of systems*
- *all are structured and organized to serve several purposes*
- *some parts are formalized, but they need not be complete or completely formalized*
A SLEEKER DOMAIN MODEL

programs in domain-specific language can be analyzed for inconsistencies, can be verified to have desirable properties

ENVIRONMENT

SYSTEM

DOMAIN-KNOWLEDGE: description of the system’s environment

DOMAIN-SPECIFIC SPECIFICATION LANGUAGE

COMPILER OR INTERPRETER FOR SPECIFICATIONS

plus runtime platform

although the domain knowledge and requirements may be implicit, without them there are no operating assumptions, properties to verify, or even tests!

REQUIREMENTS: description of how the environment should work when the system is installed
GREATEST SUCCESS STORY: THE SEMICONDUCTOR INDUSTRY

Verilog and VHDL (circa 1984) become the standard domain-specific specification languages. continual research on the important problems improves design automation.

design automation (logic synthesis and verification) is a fundamental technology for the semiconductor industry.

by now the domain and its models are vastly more complex, . . .

. . . because the models and domain have grown up together.

fabricators do not need to worry about getting locked into one tool.

continual improvements in semiconductor fabrication demand more complex domain models.

an easy start: initial domain model only needs to describe the processor and memory architectures of the early 1980s.
WHY INDUSTRY HAS TROUBLE DEVELOPING
DOMAIN MODELS

DOMAIN MODELING IS A “HARD SELL” TO MANAGEMENT

- takes time and repetition to get it right
- domain modeling is an investment that does not pay off quickly

INDUSTRY DOES NOT HAVE THE RIGHT KIND OF PEOPLE

- practitioners are good at solving whatever problem is put in front of them, while domain modeling questions what the problem is
- practitioners are good at mastering complexity, while domain modeling requires abstraction (extracting simplicity)
- practitioners are good at optimizing, while domain modeling requires separating concerns

INDUSTRY DOES NOT HAVE PEOPLE WITH THE RIGHT TRAINING

- requires formal methods

the conclusion is that if industry cannot do domain modeling, . . .

. . . then researchers must do it!
WHY RESEARCHERS SHOULD BE HAPPY TO DEVELOP DOMAIN MODELS

(BESIDES THE OBVIOUS INTELLECTUAL CHALLENGES)

because this is how to find the best research problems

because domain models are the key to making agile development methods work well

because domain models solve the “plumbing problem”—computer science contributes something valuable and tangible to the domain

“plumbing problem”: when computer scientists collaborate with researchers in other domains, they are perceived as providing no more than the plumbing that allows data to flow
SDN IS BEST KNOWN FROM THE OpenFlow STANDARD

a controller for a subnetwork maintains a centralized abstraction of the network and writes to the forwarding tables in the routers ("the control plane")

OpenFlow standardizes the interface

routers forward packets ("the data plane")

WHY SDN IS POPULAR

• industry sees it as the key to virtualization of routers—and big savings because routers are so expensive

• researchers see it as a place to apply knowledge of programming languages and formal methods as well as networking
THE “CLASSIC” INTERNET ARCHITECTURE

this architecture has succeeded (beyond most peoples’ wildest dreams) in fostering innovation and shaping the world we live in.

however, it is now widely agreed that it does not meet society’s present and future requirements.

security
dependability
mobility
scalability
quality of service
resource management

the trend is toward a more pluralistic architecture . . .

. . . with multiple, customized protocol stacks
WHAT IS REALLY GOING ON

headers in a typical AT&T packet, one header per layer: 12 instead of 4

<table>
<thead>
<tr>
<th>Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
</tr>
<tr>
<td>HTTP</td>
</tr>
<tr>
<td>TCP</td>
</tr>
<tr>
<td>IP</td>
</tr>
<tr>
<td>IPsec</td>
</tr>
<tr>
<td>IP</td>
</tr>
<tr>
<td>GTP</td>
</tr>
<tr>
<td>UDP</td>
</tr>
<tr>
<td>IP</td>
</tr>
<tr>
<td>MPLS</td>
</tr>
<tr>
<td>MPLS</td>
</tr>
<tr>
<td>Ethernet</td>
</tr>
</tbody>
</table>

HTTP being used as a transport protocol because it is the only way to traverse NAT boxes and firewalls

security

cellular service (mobility, QoS, billing)

multiple layers of resource management
there is a fixed number of levels

the scope of each layer is global, so layer = level

each layer/level has a specialized function

there can be any number of levels

some layers have small or local scopes

each layer is a microcosm of networking, containing all the basic functions (state components and mechanisms)
WE CALL THIS THE "GEOMORPHIC VIEW" OF NETWORKING . . .

. . . BECAUSE THE COMPLEX ARRANGEMENT OF LAYERS RESEMBLES THE EARTH’S CRUST
TODAY’S INTERNET, CLASSIC AND GEOMORPHIC VIEWS

CLASSIC VIEW:
Stuff all the new complexity into the network layer, which is the only place for it.

SO FAR, this is the approach that SDN research is taking.

GEOMORPHIC VIEW:
- accurately describes the structure of today’s Internet
- relatively simple layers modularize the complexity

Even if the implementation looks like this, the geomorphic view is a better abstraction for structuring the software and analyzing its properties.
THOUGHTS ON SDN

WHAT NICK McKEOWN SAID

One of the three major benefits of SDN is a well-defined control abstraction that can be implemented separately from the forwarding plane . . .

. . . so that software engineering can be applied to this implementation.

WHAT I OBSERVE

● repertoire of “properties to prove” is a bit boring

● many conflicting requirements (from different stakeholders), with little help in resolving the conflicts

● serious complexity problems in all aspects: modeling networks, expressing desired properties, deciding properties

“tunneling makes the state explode”

WHAT IS SOFTWARE ENGINEERING?

Above all, software engineering is about . . .

. . . modularity

. . . separation of concerns, which is what you get from layers in the geomorphic view.

It can help you . . .

. . . develop re-usable theories that apply at many levels for many different purposes

. . . understand where the requirements come from and how conflicts should be resolved

. . . manage complexity

. . . extend SDN beyond the most basic aspects of networking.
I won’t be telling you anything you don’t already know, . . .

. . . but maybe I can reinforce a healthy trend and give you a few new examples.
OBSTACLES TO DOMAIN MODELING AS RESEARCH IN UNIVERSITIES

LEARNING ABOUT THE DOMAIN

- no access to domain experts, or . . .
- . . . domain experts do not have time for you
- need long-term stable funding to commit to learning a domain

PUBLICATION

- work in cooperation with industry may not be released for publication
- domain-specific results are interdisciplinary—there is no place to publish them
- a descriptive model is not a new result
- comparing models of a domain is not science, it is religion
- the pressure to publish in quantity is too great to take any risks
- what matters is citation by fellow researchers, not real-world impact
IT’S A BIG, OPEN WORLD OUT THERE

CHALLENGING THESE OBSTACLES:

- no access to domain experts
- domain experts do not have time for you
- need long-term stable funding to commit to learning a domain
- work in cooperation with industry may not be released for publication

IMPORTANT DOMAINS HAVE MANY PLAYERS

- established companies
- start-ups
- standards bodies
- government regulators

INFORMATION IS WIDELY AVAILABLE

- open-source software
- standards documents
- (almost) everything is on the Web!
- collaboration between university departments
- people working in and with industry do get their papers released
a new technology is arousing commercial interest, but the customers (e.g., Internet service providers) are holding back

CUSTOMERS WANT:

- to be sure the technology will succeed before adopting it
- to avoid interoperability problems
- to avoid being the captive of one vendor

VENDORS WANT:

- to bring their products to market first
- to differentiate their products from those of other vendors
- to capture customers so that they cannot change vendors

obviously standards benefit the customers more than the vendors . . .

. . . and vendors accept them because the customers demand them

TALES FROM THE INTERNET ENGINEERING TASK FORCE
once the standards process has begun, the vendors try to control it

VENDORS WANT:

- to make the process as fast as possible, by finishing a few basic use cases first
- to standardize as little as possible

WHICH HAS THESE UNFORTUNATE SIDE-EFFECTS:

- with no early thought about generality, each new increment of capability requires a similar or greater increment of complexity
- the standard has many recommendations and optional extensions

a protocol with N optional extensions has, in effect, $2^N$ versions

THE ABSENCE OF FORMAL METHODS MAKES THESE PROBLEMS MUCH WORSE
**THE SIP STANDARD**

- **THE MEDIUM**
  - IETF philosophy is to standardize based on "rough consensus and working code"
  - finite-state machines are rarely used
  - specifications are written in English, augmented only by message sequence charts that usually look like this (IETF macros):

```
  process1                        process2
```

- note how this forces you to forget race conditions!

- **THE MESSAGE**
  - the base document (IETF RFC 3261) is 268 pages
  - it is continually being extended, bottom-up, in response to an endless series of new use cases
  - "A Hitchhiker's Guide to SIP" is a snapshot of SIP RFCs and drafts as of 2009 . . .
  - . . . which lists 142 documents, totaling many thousands of pages
it sometimes takes hours to get an answer to a simple question about SIP (and even then you are not sure)

Test cases are insufficient to insure interoperation of products (which is the main purpose of a standard)

many people don’t want to use SIP because it is too complex, are looking for simpler alternatives

the overall inefficiency and and waste are staggering

FOR COMPUTER SCIENCE, THIS IS LOW-HANGING FRUIT

working with SIP, straightforward modeling and model-checking . . .

. . . provided unambiguous, searchable documentation

. . . revealed many inconsistencies and unknown race conditions

. . . suggested simplifications

. . . automatically generated thousands of test cases

at the same time, the diverse aspects and scale of real standards means that there are many interesting research questions to work on
HOW TO INFILTRATE THE STANDARDS PROCESS

1. get involved with new standards, where the mess is not yet hopeless

2. achieve credibility (without attending endless meetings) with the results of automated analysis

3. provide up-to-date, searchable, unambiguous documentation

4. generate test cases automatically

5. tell your granting agency that you want to improve commercial standards

6. go to work for a vendor and convince your colleagues that formal methods are a secret weapon

7. go to work for a customer and convince your colleagues that formal methods are a protective shield
AN INDUSTRY PERSPECTIVE ON PUBLICATION

CHALLENGING THESE OBSTACLES:

- domain-specific results are interdisciplinary—there is no place to publish them
- a descriptive model is not a new result
- comparing models of a domain is not science, it is religion
- the pressure to publish in quantity is too great to make long-term investments or take any risks
- what matters is citation by fellow researchers, not real-world impact

THESE ATTITUDES SEEM SOMEWHAT OUT-OF-BALANCE

- the world of computing already has far too many mechanisms, too little ability to compose them into something of lasting value
- most published models are toys, which is why there are few interesting differences between them—there are many important differences between industrially useful models
- if the system discourages work on the most important problems, then maybe the system should be changed