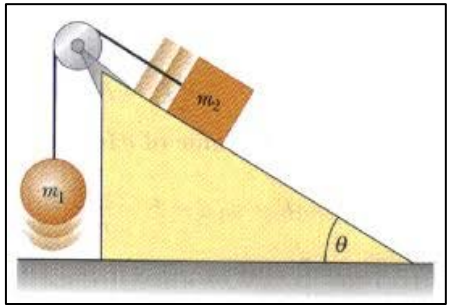
$$\nabla \cdot \boldsymbol{E} = \rho/\varepsilon_0$$
$$\nabla \cdot \boldsymbol{B} = 0$$
$$\nabla \times \boldsymbol{E} = -\frac{d\boldsymbol{B}}{dt} = -\left\{\frac{\partial \boldsymbol{B}}{\partial t} + \boldsymbol{v} \cdot \nabla(\boldsymbol{B})\right\}$$
$$\nabla \times \boldsymbol{B} = \mu_0 \boldsymbol{J} + \frac{1}{c^2}\frac{d\boldsymbol{E}}{dt} = \mu_0 \boldsymbol{J} + \frac{1}{c^2}\left\{\frac{\partial \boldsymbol{E}}{\partial t} + \boldsymbol{v} \cdot \nabla(\boldsymbol{E})\right\}$$
$$\nabla \cdot \boldsymbol{J} = -\frac{d\rho}{dt} = -\left\{\frac{\partial \rho}{\partial t} + \boldsymbol{v} \cdot \nabla(\rho)\right\}$$

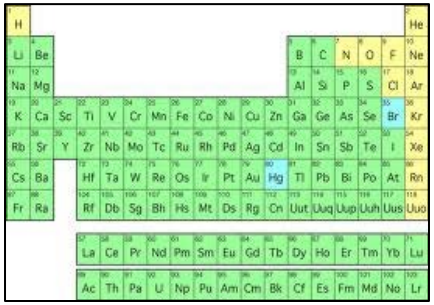
$$R_{ab} - \frac{1}{2}Rg_{ab} = \frac{8\pi G}{c^4}T_{ab}$$

ALBERT EINSTEIN'S GENERAL THEORY OF RELATIVITY, 1916

Why Meta-Theories of Automated Software Design Are Essential

Don Batory
Department of Computer Science
University of Texas at Austin
Austin, Texas 78712





My Introduction

- Worked 30+ years in software product lines and program generation
    - contribution: expose elementary mathematics that underlies activities of automated software development

We are geniuses at making the simplest things look complicated; finding the underlying simplicity is the challenge

- To generate a quality (correct) program – you have to know a LOT:
    - thoroughly understand the domain
    - thoroughly understand the trade-offs that experts make in program design
    - encode “best practices” of engineering and design
    - goal to mechanize this knowledge/process

- With this in mind, here is what I want to say…

Definition of Science

- From `dictionary.com`

sci·ence [sahy-*uh* ns] Show IPA

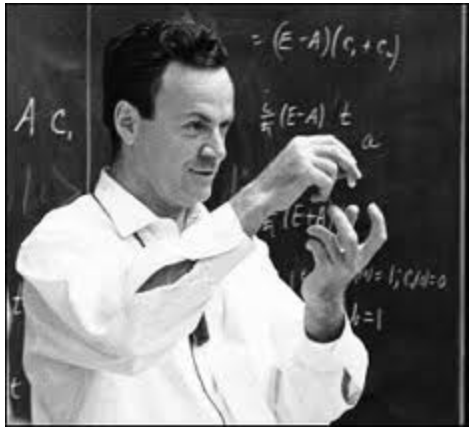
noun

1. a branch of knowledge or study dealing with a body of facts or truths systematically arranged and showing the operation of general laws: *the mathematical sciences.*
2. systematic knowledge of the physical or material world gained through observation and experimentation.
3. ~~any of the branches of natural or physical science.~~
4. systematized knowledge in general.

- Dominant paradigm in SE insists on a hypothesis evaluation. A set of tests are conducted by an author and a careful analysis of one or more hypotheses must be presented. This is the "Scientific Method"
- This matches Definition 2 and the intended use of empirical methods in SE
- We are missing the most important part of science

And the Important Part?

- My answer is an analogy from physics…
- In physics, there are lots of poorly related phenomena – they vary some how
- A theoretical physicist would select a set and seek a mathematical theory that unifies them as manifestations of the same underlying concepts
    - broader the initial set
    - fewer the concepts
    - more general and significant the theory might be

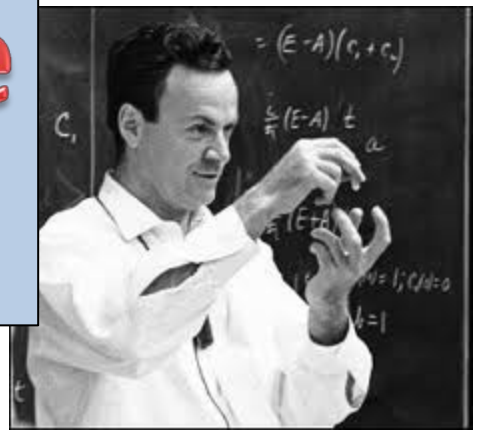


- Initial test of a theory is a check that it does precisely what it claims
    - reproduce, explain phenomena of the initial set
    - explain, predict other phenomena as well

And the Important Part?

- My answer is an analogy from physics…
- In physics, there are lots of poorly related phenomena – they vary some how
- A theoretical phy
  mathematical the
  of the same unde
  - broader
  - fewer th
  - more general and significant the theory might be

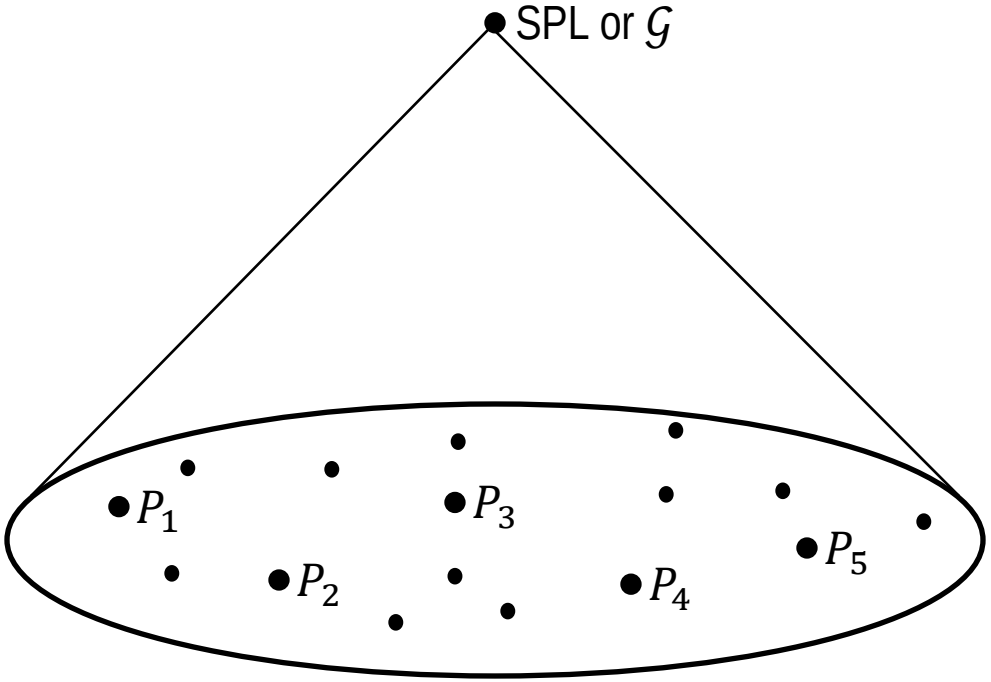
This is Science Definition #1



- Initial test of a theory is a check that it does precisely what it claims
  - reproduce, explain phenomena of the initial set
  - explain, predict other phenomena as well

Automated Software Design

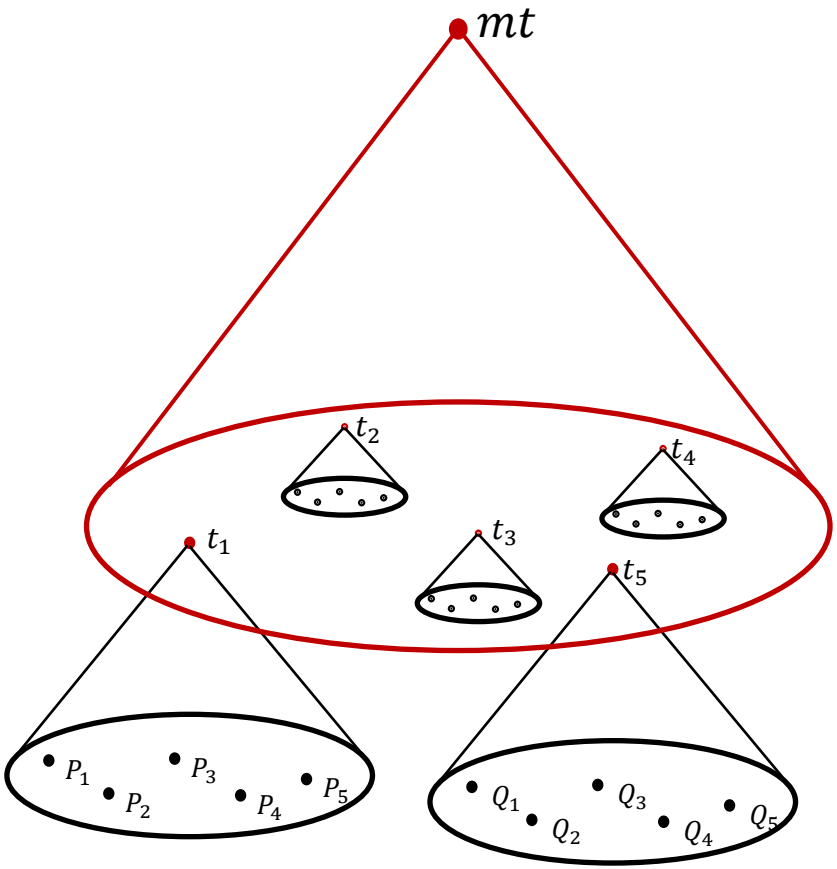
- Manufacture programs with certain properties
- A software product line (SPL) or generator ($\mathcal{G}$), is a concrete embodiment of an "implicit" theory of how to automatically build programs in this domain with lower cost and higher quality



SPL or $\mathcal{G}$ not only explains and reproduces initial programs, but predicts and explains the existence of other programs as well

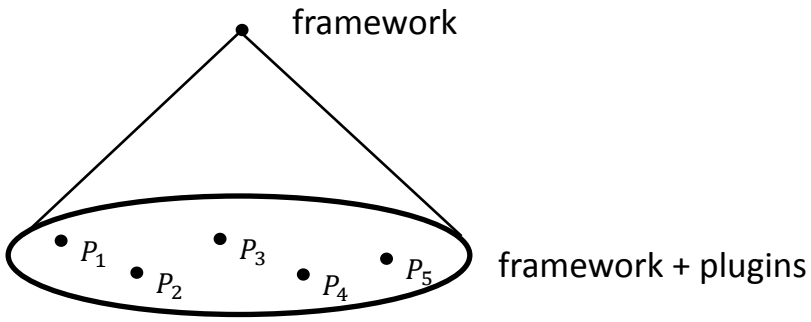
History and Experience Tells Us

- Such "theories" must be domain-specific (DS) to have any chance of success
- DS knowledge is rich and deep, with few specifics transferable to other domains
  - irony: DS theories (t_1 ... t_n) are not very interesting to the general SE community
- Meta-theories (mt) are more valued
  - domain-independent concepts
  - instances are DS theories
  - teach ideas to students; they will produce instances of their own



Familiar SE Meta-Theories

- Just not very “automatic” or mathematical
- OO frameworks are common in today’s libraries
  - framework designers understand that a set of similar programs will be built
  - their OO framework codes the common objects and activities in this domain to minimize what others have to write
  - ideas of frameworks, abstract classes, plugins are meta-theory
  - we teach (meta-theory) to our students
  - our students instantiate ideas to create frameworks, plugins of their own



Another Example

- UML asserts than an OO design can be documented in the languages of class diagrams, state machines, etc. (the meta-theory part)
- We teach UML (meta-theory) to our students; they instantiate to design their own OO programs

sci·ence [sahy-*uh* ns] Show IPA

noun

1. a branch of knowledge or study dealing with a body of facts or truths systematically arranged and showing the operation of general laws: *the mathematical sciences.*
2. systematic knowledge of the physical or material world gained through observation and experimentation.
3. any of the branches of natural or physical science.
4. systematized knowledge in general.

- Not Definition 1, likely Definition 4

Why a Meta-Theory is Important

- How tools should work – gives a precise definition of what “composition” means
    - are you aware of the volume of technical papers in SE where “composition” makes no sense mathematically?

- In mature communities, MT provides a standard way to describe problems and how to formulate solutions
    - type systems for programming languages
    - relational algebra and sets for classical databases
    - conceptual & technical glue that holds communities together

- MTs bring organization to what would otherwise be intellectual chaos

Form of a Theory

- Paraphrasing Dijsktra:

  Today's programs are among the most complex structures ever built by man

- Todays' tools manipulate and map structures:
  - compilers map source structures to byte-code structures
  - refactoring tools map source structures to source structures
  - model driven engineering is all about transforming models

- Mathematics is the science of structure and structure manipulation

There has to be a connection…

My Research
(which I illustrate in my paper)

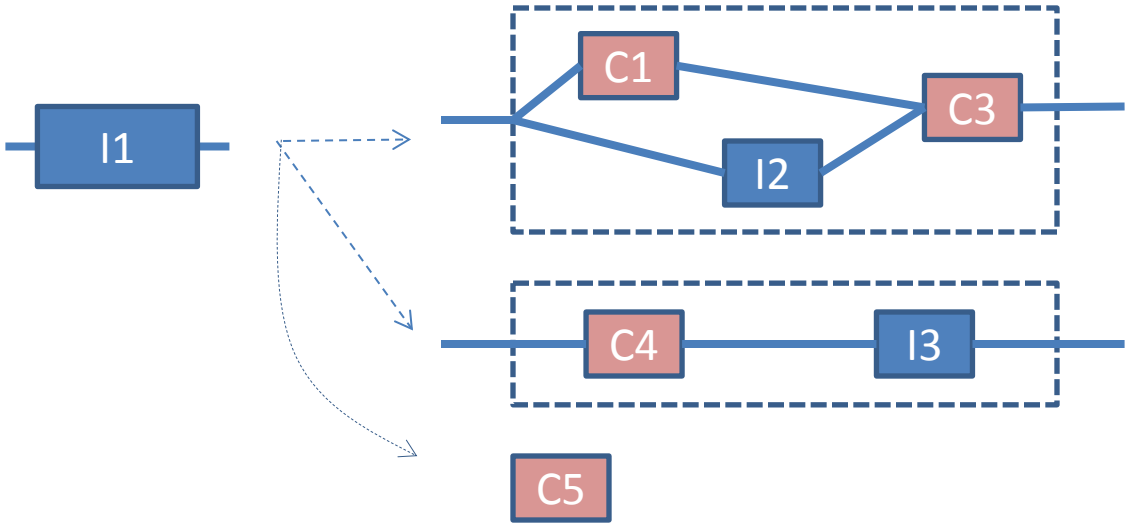
- Traveled from practice to theory
- Exposed fundamental:
    - mathematical structures that underlie program construction in software product lines
    - transformations underlie automated program construction
    - how correct by construction can scale to large systems
- Some thoughts for the GTSE attendees…

My Perspective

- Chorus: Software design is an art form – it should be treated and taught as such
    - Dick Gabriel – software is 'poetry'
- If you are build/design a 1-of-a-kind product – design is an art form
- Also heard: "We've done this so many times, we've got it down to a science"
- If you have ever built variants of a program, with different functionalities
    - think on a bigger plane, designs are not just for 1-of-a-kind, but for a family of programs
    - "physics" of design is **very** different

Science of Design

- Doesn't come from 1-of-a-kind designs
- Comes from studying a family of designs of similar systems
    1. know the domain
    2. know how to engineer software in a domain (which is different from 1)
    3. know how to codify design knowledge in abstract structures, rules
    4. codify knowledge as grammars (equations)



encode meta-rules for decision-making

"Core Engineering" to me is

- Mechanizing, standardizing what experts know to avoid reinvention
  - stand on the shoulders of others, not stand on their feet

- In the domains in which I have worked, "SE" has done an exceptionally poor job at this form of Engineering

en·gi·neer·ing [en-*juh*-**neer**-ing] Show IPA

noun

1. the art or science of making practical application of the knowledge of pure sciences, as physics or chemistry, as in the construction of engines, bridges, buildings, mines, ships, and chemical plants.
2. the action, work, or profession of an engineer.
3. skillful or artful contrivance; maneuvering.

- Most software engineering today fits Definition 3, not 1.

Applies to LOTS of Domains

- 10 years ago, there were many domains which could be standardized in this manner
- Today there are more….
- 10 years from now, many more still
- Yet, we as a community don't teach automated design or how to get there…
    - and people wonder where is the science…

Questions?