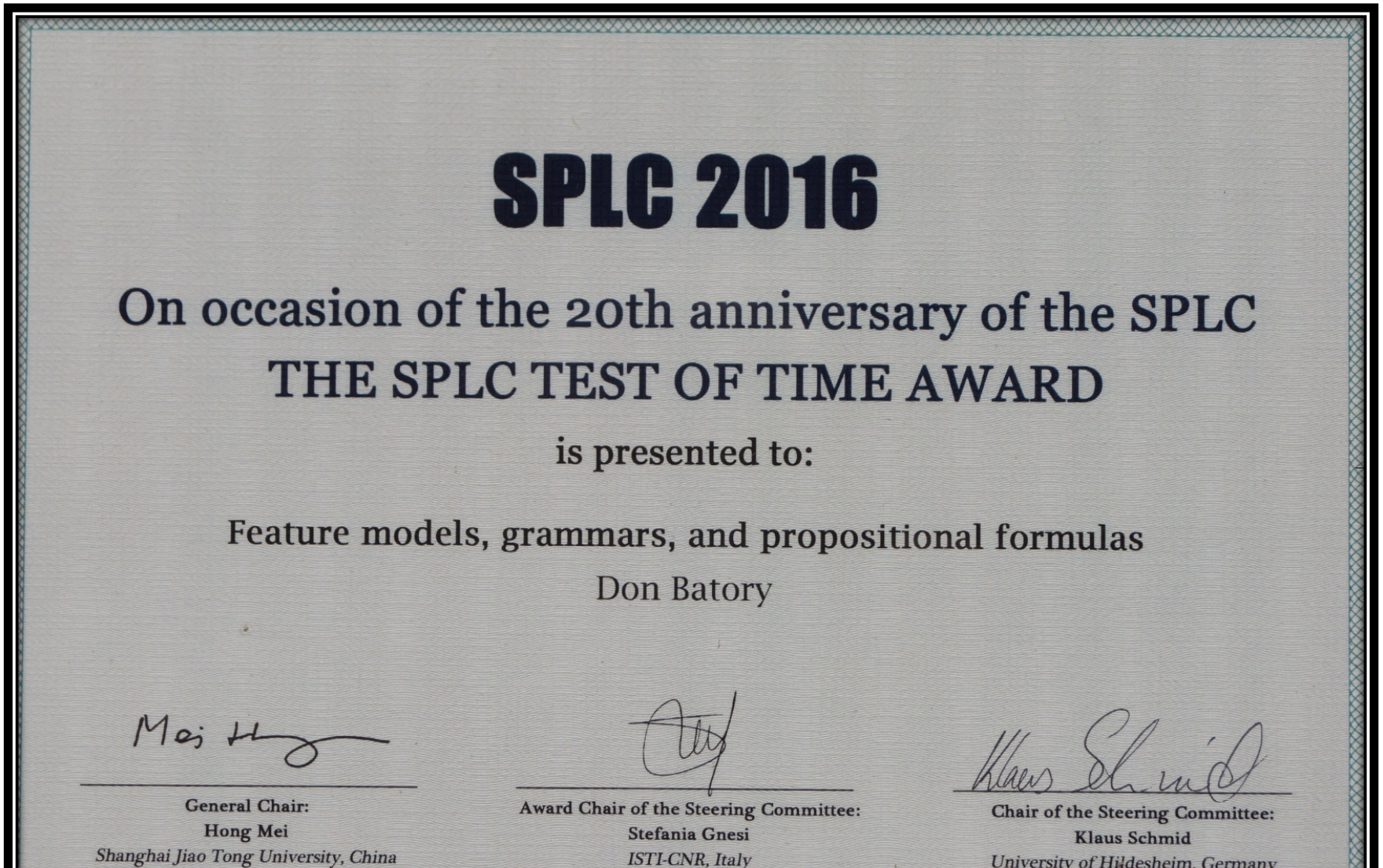
Test of Time Award and Conjectures on the Future of SPLs

Don Batory
Department of Computer Science
University of Texas at Austin

THANK SPLC STEERING COMMITTEE!

For the 1st SPLC Test of Time Award!



SPLC 2016

On occasion of the 20th anniversary of the SPLC

THE SPLC TEST OF TIME AWARD

is presented to:

Feature models, grammars, and propositional formulas

Don Batory

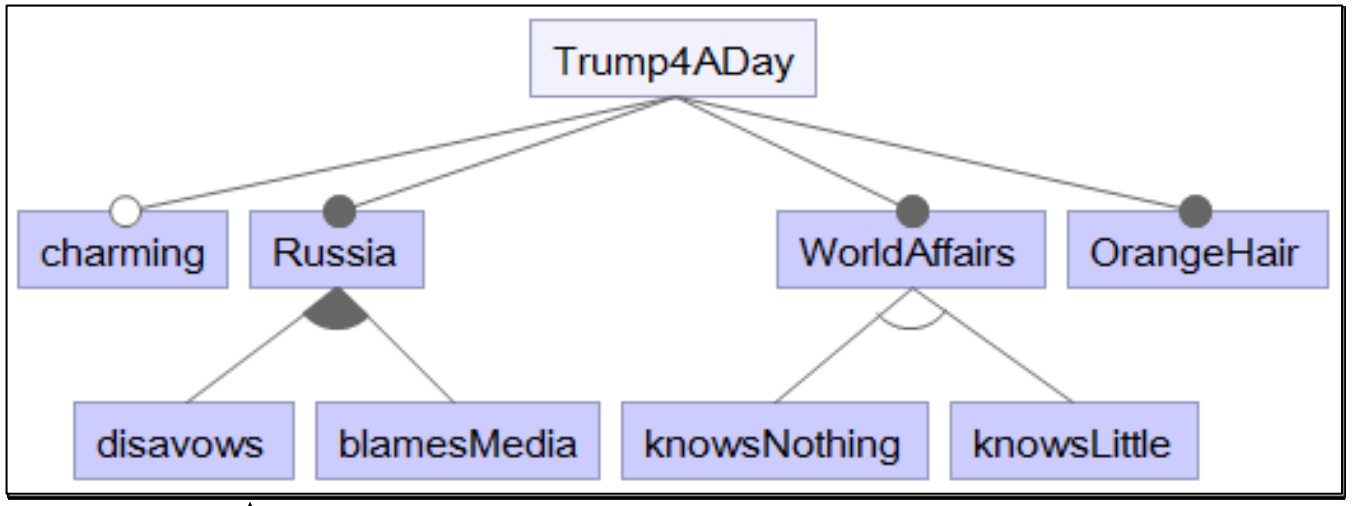
General Chair:
Hong Mei
Shanghai Jiao Tong University, China

Award Chair of the Steering Committee:
Stefania Gnesi
ISTI-CNR, Italy

Chair of the Steering Committee:
Klaus Schmid
University of Hildesheim, Germany

THE PAPER SHOWED

- Feature Diagrams (FDs)
  are context-free grammars
  fundamental concept in CS



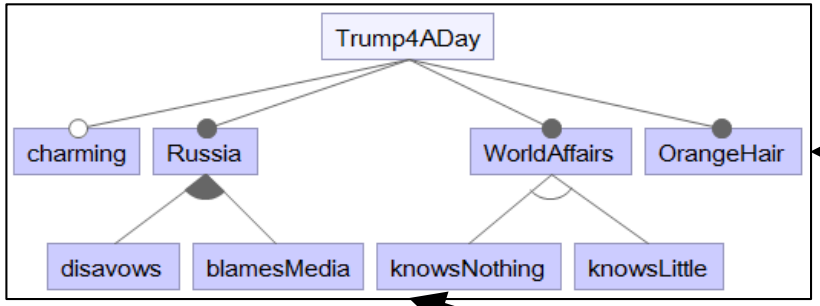
Productions:

```
Trump4ADay   : [charming] Russia+ WorldAffairs OrangeHair ;
Russia       : disavows | blamesMedia ;
WorldAffairs : knowsNothing | knowsLittle ;
```

- Not just any grammar – there are restrictions… *more later*

THE PAPER SHOWED

- FDs & grammars could also be mapped to propositional formulas – *yet another fundamental CS concept*




```
Trump4ADay  : [charming] Russia+ WorldAffairs OrangeHair ;
Russia      : disavows | blamesMedia ;
WorldAffairs  : knowsNothing | knowsLittle ;
```

$$
\begin{aligned}
&\text{Trump4ADay} \\
&\quad \wedge \\
&\text{Trump4ADay} \quad \Leftrightarrow (\text{Russia} \wedge \text{WorldAffairs} \wedge \text{OrangeHair}) \\
&\quad \wedge \qquad\qquad\quad \wedge \text{Charming} \Rightarrow \text{Trump4ADay}; \\
&\text{WorldAffairs} \quad \Leftrightarrow \text{choose1}(\text{knowsNothing}, \text{knowsLittle}); \\
&\quad \wedge \\
&\text{Russia} \qquad\quad \Leftrightarrow \text{disavows} \quad \vee \quad \text{blamesMedia};
\end{aligned}
$$

THE PAPER CONTINUED

- Cross Tree Constraints (CTC) – can be any propositional formula involving features

$$\texttt{charming} \vee \texttt{blamesMedia} \Rightarrow \texttt{knowsNothing}$$

STILL NOT STANDARD

- Basic identity:

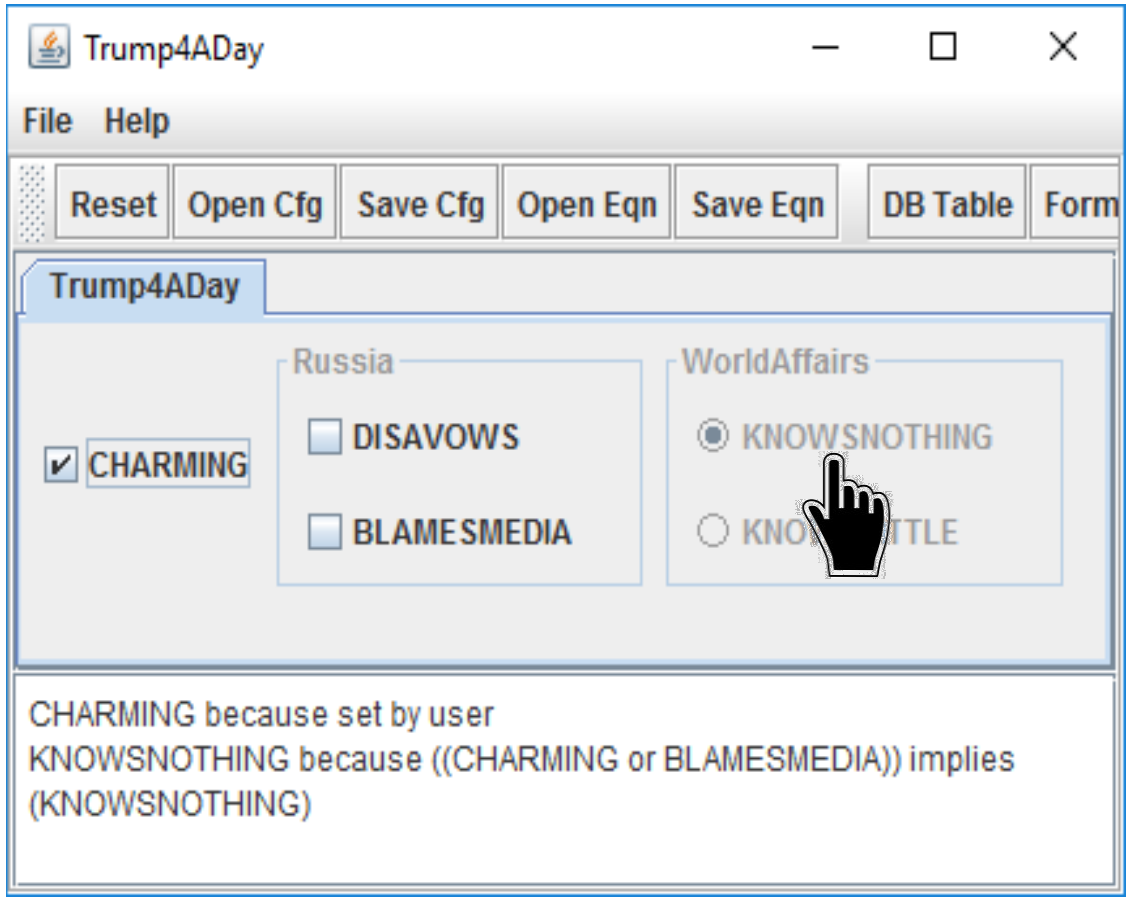
$$\rho(FM) = \rho(FD) \wedge \rho(CTC)$$

- Now use off-the-shelf SAT technology to analyze $\rho(FM)$

KEY CONTRIBUTION

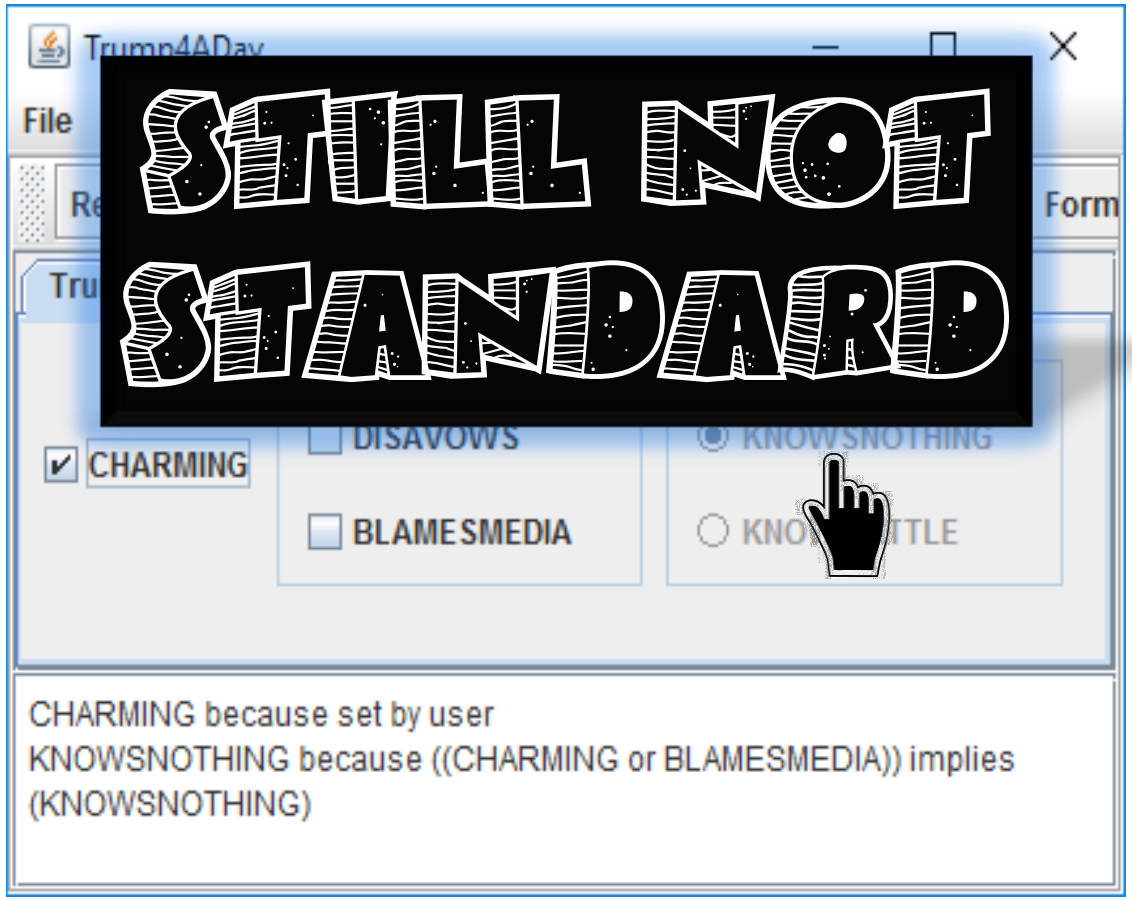
The Paper Continued

- Provide explanations for inferred features *a proof*



THE PAPER CONTINUED

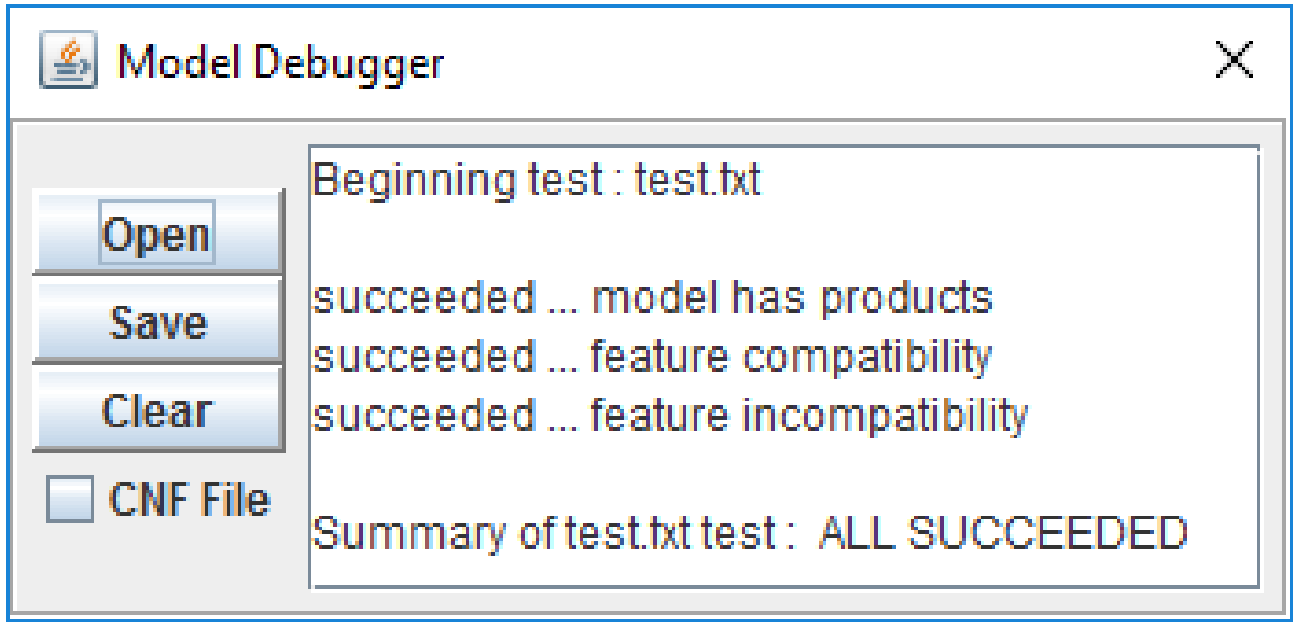
- Provide explanations for inferred features *a proof*

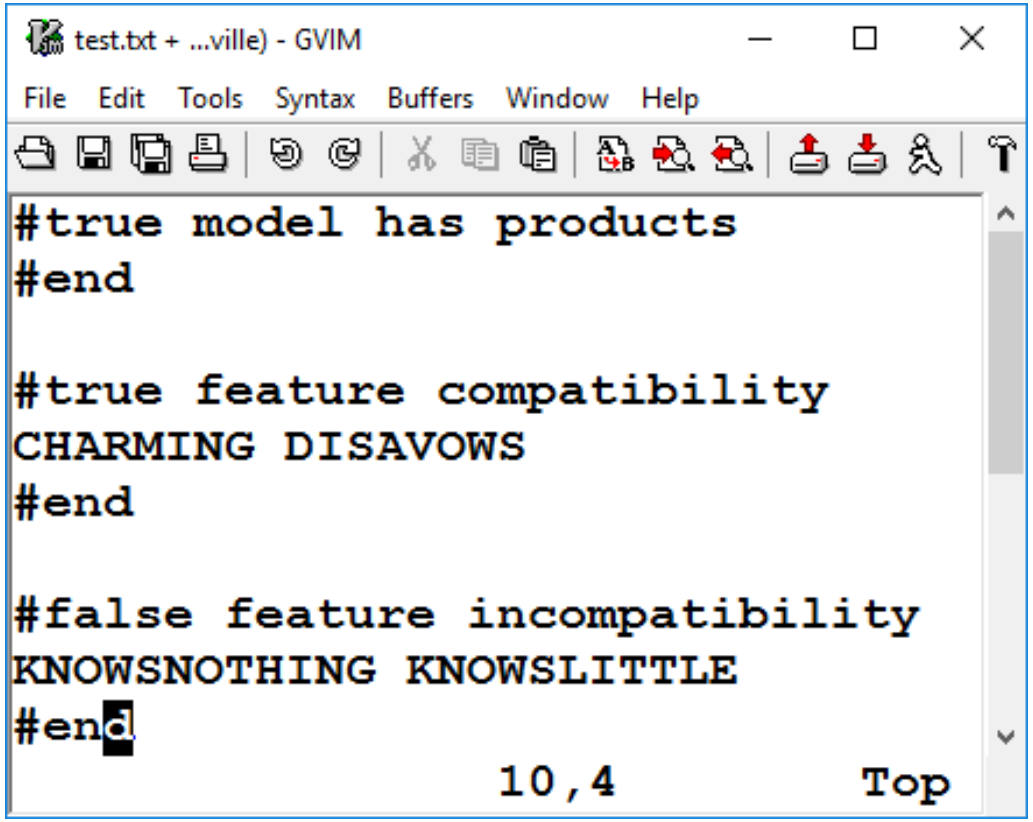


STILL NOT STANDARD

THE PAPER CONTINUED

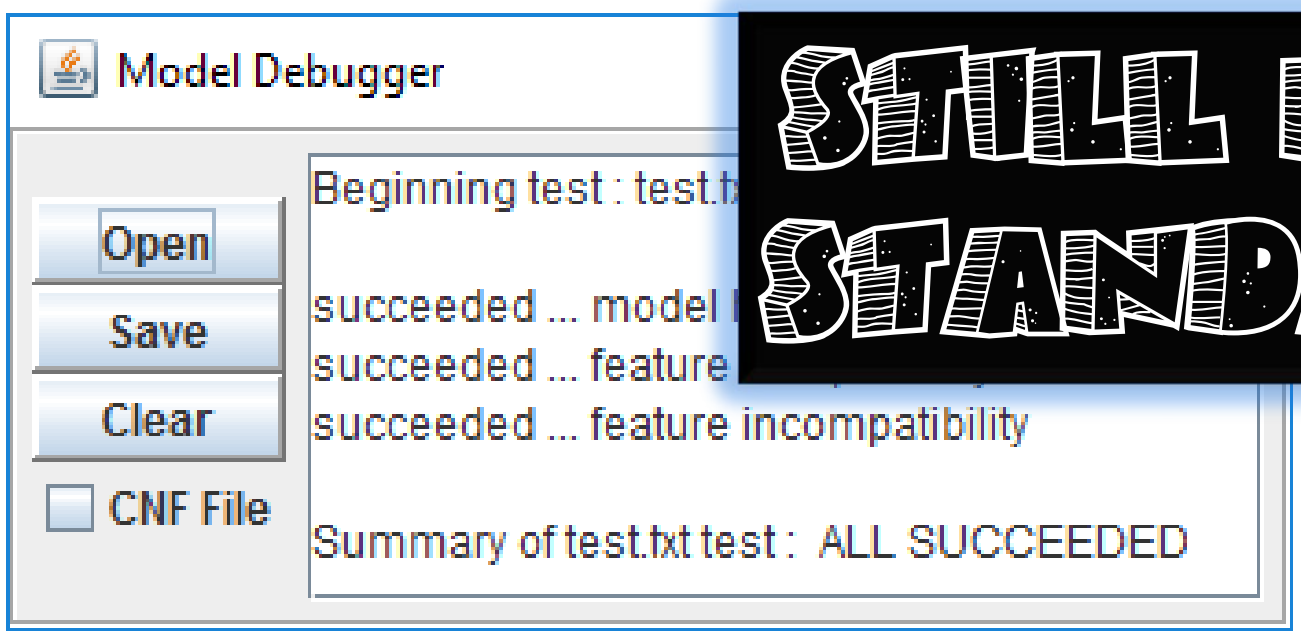
- Debugging FMs – Unit tests for FMs



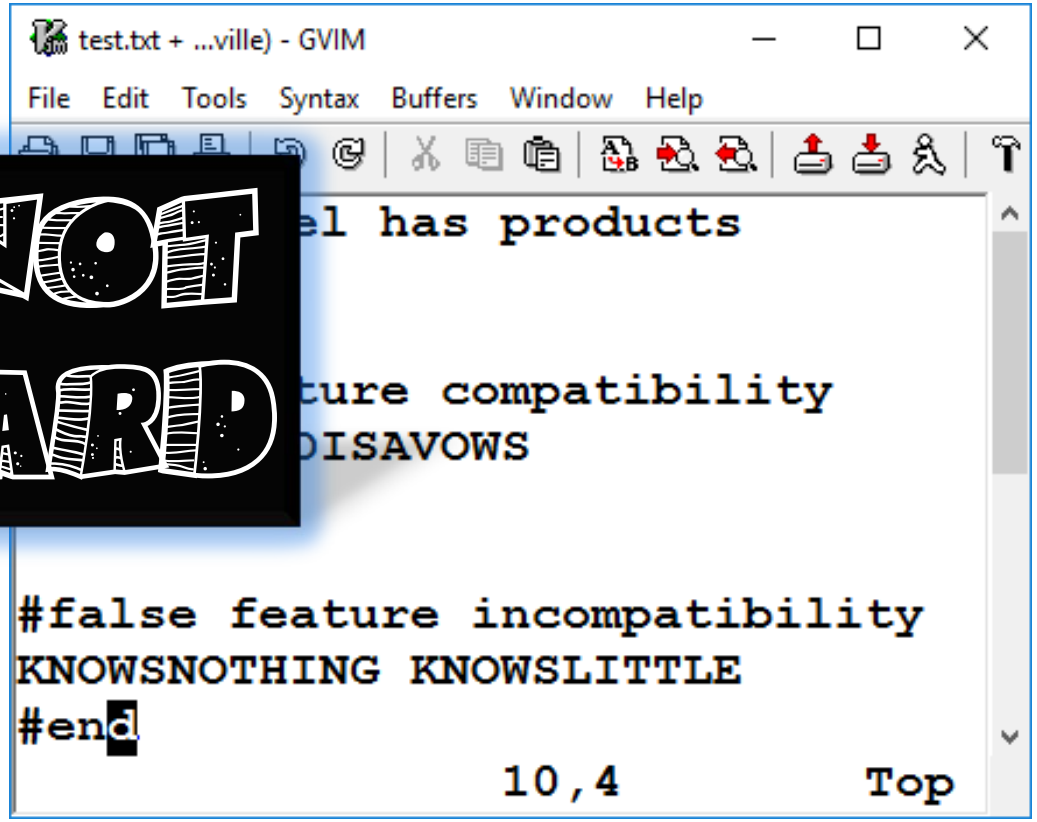


THE PAPER CONTINUED

- Debugging FMs – Unit tests for FMs



STILL NOT STANDARD



PERSONAL PERSPECTIVE

- Influential technical papers tend to be 1 of 3 kinds:

IDEA PAPERS	– keep researchers busy for years
BREAKTHROUGH PAPERS	– stars must align as breakthroughs are rare
WHO GETS THERE FIRST	– known ideas waiting to be put together

- This paper was of the kind:

WHO GETS THERE FIRST

Salute to Contemporaries

- Benavides CAISE 2005 “Automated Reasoning on Feature Models”
  - FMs with attributes ⇒ 0-1 optimization problems
  - gateway to using CSP solvers *today SMT solvers*

- Czarnecki’s GPCE 2006 “Verifying Feature-Based Model Templates”
  - gateway to verify type-safety of all products in SPL

- Schoebben’s ReqE 2006 “Feature Diagrams: A Survey and A Formal Semantics” MIP ReqE 2016
  - defined a formal semantics that unified existing FMs

Forgive me if I am missing you

MY WORK INFLUENCED BY…

- Perry's IS
  - fur
    co
  - ba
- Kang et al

THE GENESIS DATABASE SYSTEM

Feature-Oriented Domain Analysis (FODA)
Feasibility Study

Kyo C. Kang
Sholom G. Cohen
James A. Hess
William E. Novak
A. Spencer Peterson

November 1990

WHAT KIND OF GRAMMAR ?

- Aliakbar Safilian, PhD Thesis McMaster University, 2016
    - FDs are grammars but even simpler than that!

REGULAR EXPRESSIONS

```
Trump4ADay : [Charming](disavows|blamesMedia)+
      (knowsNothing|knowsLittle) OrangeHair ;
```

- Still context sensitive constraints. Formalism admits replicated features, multi product lines

Significant follow-on work

- *Not in any order*

- family based verification of type safety for all products Czarnecki et al.
- model checking of product lines Claussen, Legay, Schoebbens, et al.
- using SAT and beyond to analyze Linux Apel, Kästner, et al.

Nadi et al.

Forgive me if I'm missing you

Significant follow-on work

- *Not i*

ebbens, et al.

So again

Thank you !

CONJECTURES ON THE FUTURE OF SPLS

3 OBSERVATIONS

HAVE YOU OBSERVED...

1. **QUALITY** of software seems to be getting worse?
   - maybe 10× more complex than it was 10 years ago
   - I'm seeing more problems than ever before

Adam
Kilvans
AI, Theory, ML

2. **KNOWLEDGE** about SPLs is more widespread and more shallow
   - why is refereeing become "awful" or "unpredictable"?
   - reviews are off-base, punitive – how can we fix this?

3. **IMPORTANCE** of SPLs is established, but where might SPLs be in next 20 years?

HAVE YOU OBSERVED...

1. QUALITY of software seems to be getting worse?
   - maybe 10× more complex than it was 10 years ago
   - I'm seeing more problems than ever before – ex: Skype

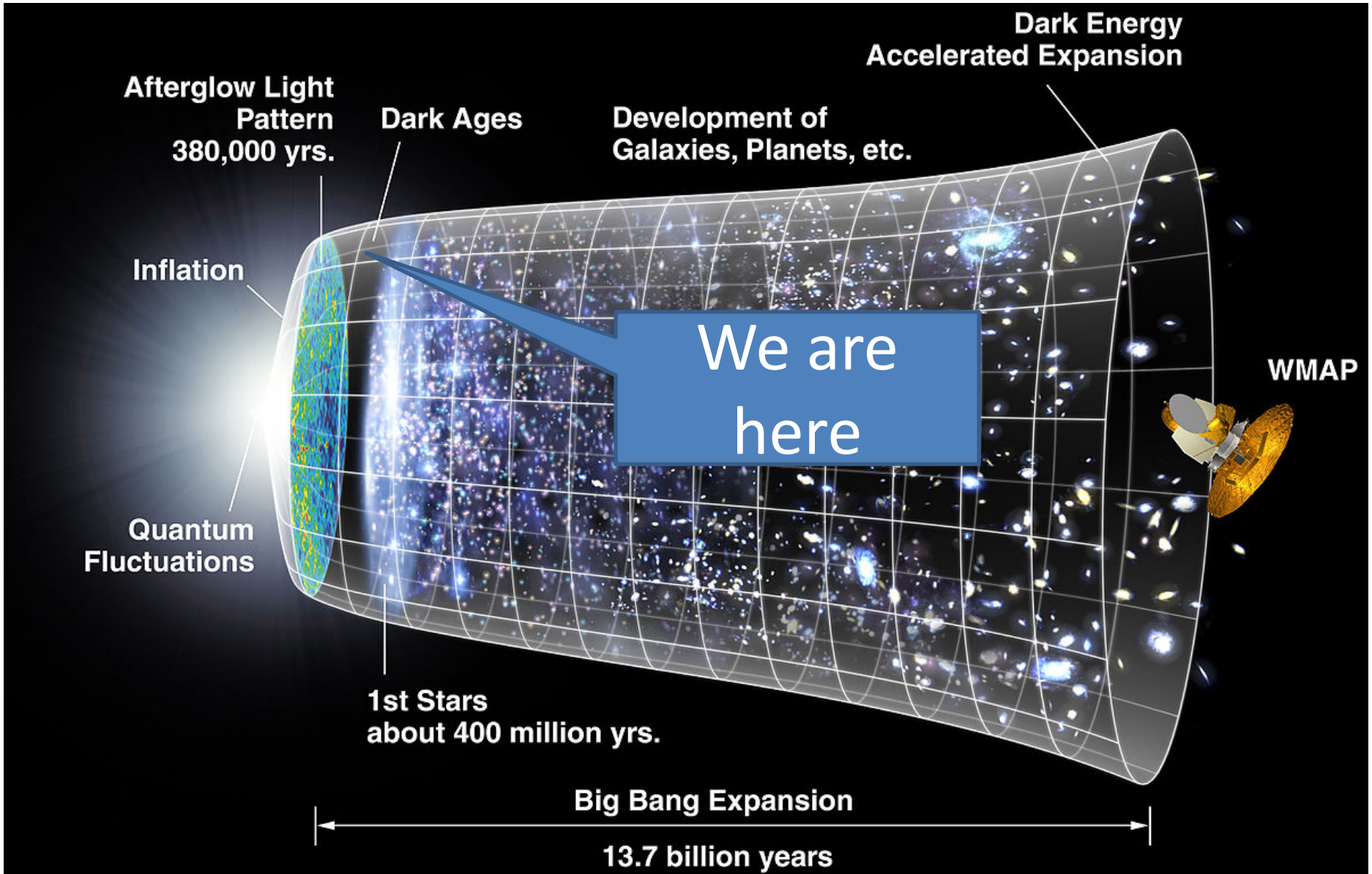
Adam
Kilvans
AI, Theory, ML

I ADMIT I DON'T KNOW THE ANSWERS
BUT...
I KNOW THEIR ANSWERS ARE RELATED

2. IMPORTANCE of SPLs is established, but where might SPLs be in next 20 years?

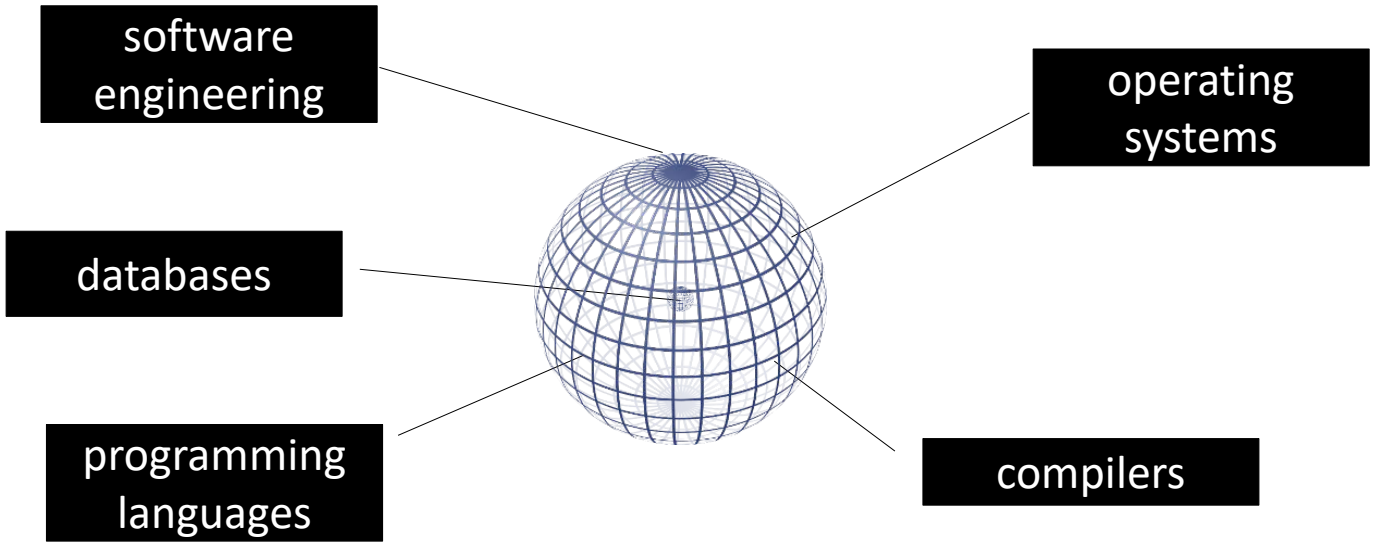
BACKGROUND

NOW PAST HYPER-INFLATION



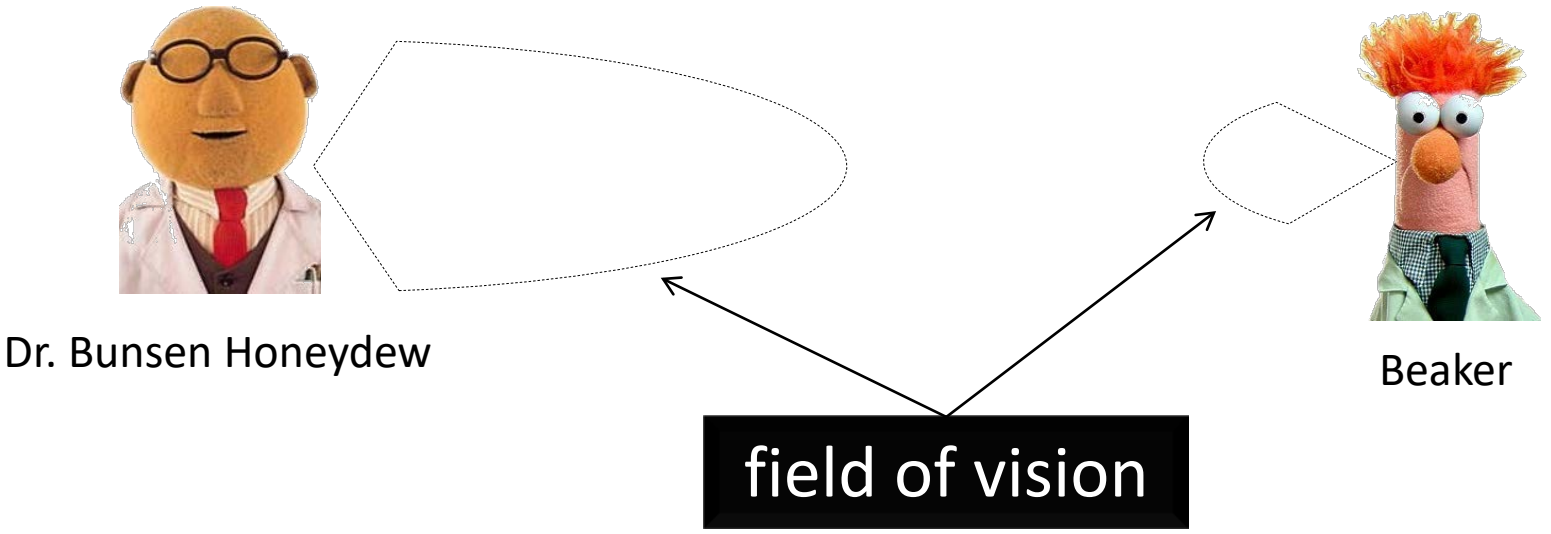
In the beginning 40s-50s

- Universe of Software Science was small
- It grew fast from virtually nothing



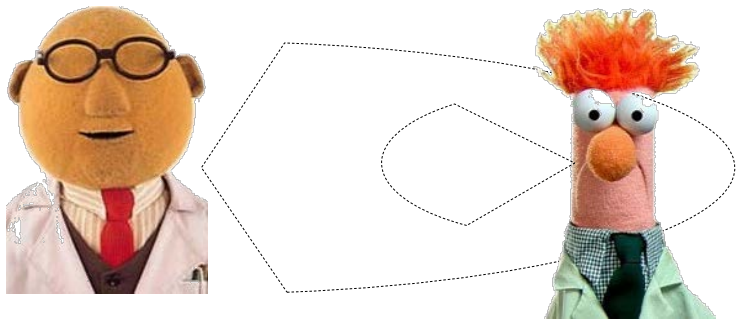
LAWS OF THIS ARTIFICIAL UNIVERSE

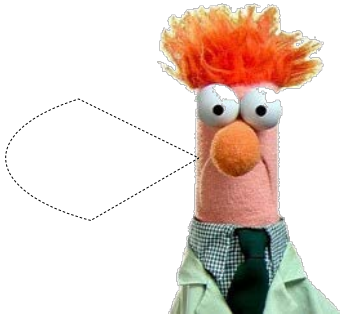
- Limitations of humans ex: We are naturally near-sighted *limited capacity to absorb & understand*
- Field of vision is volume of knowledge that person understands
- Different people have different fields of vision, different depths of knowledge



IMPLICATIONS

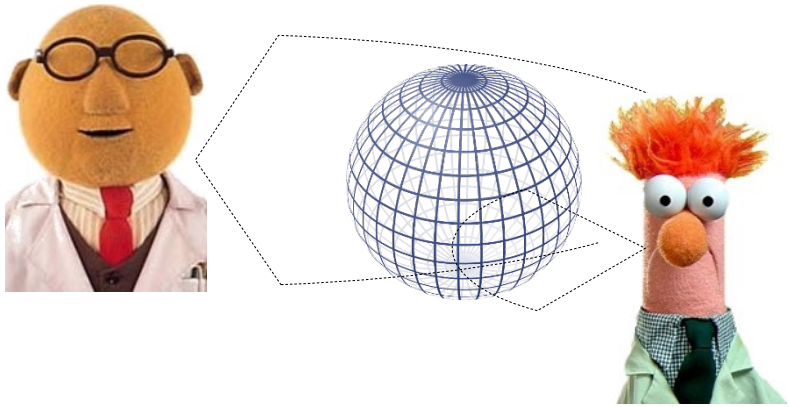
- In close proximity, both know what the other is doing
- When they go beyond each other's "horizon", no longer in touch,
  can't see what the other is doing and know little or nothing about each other's research





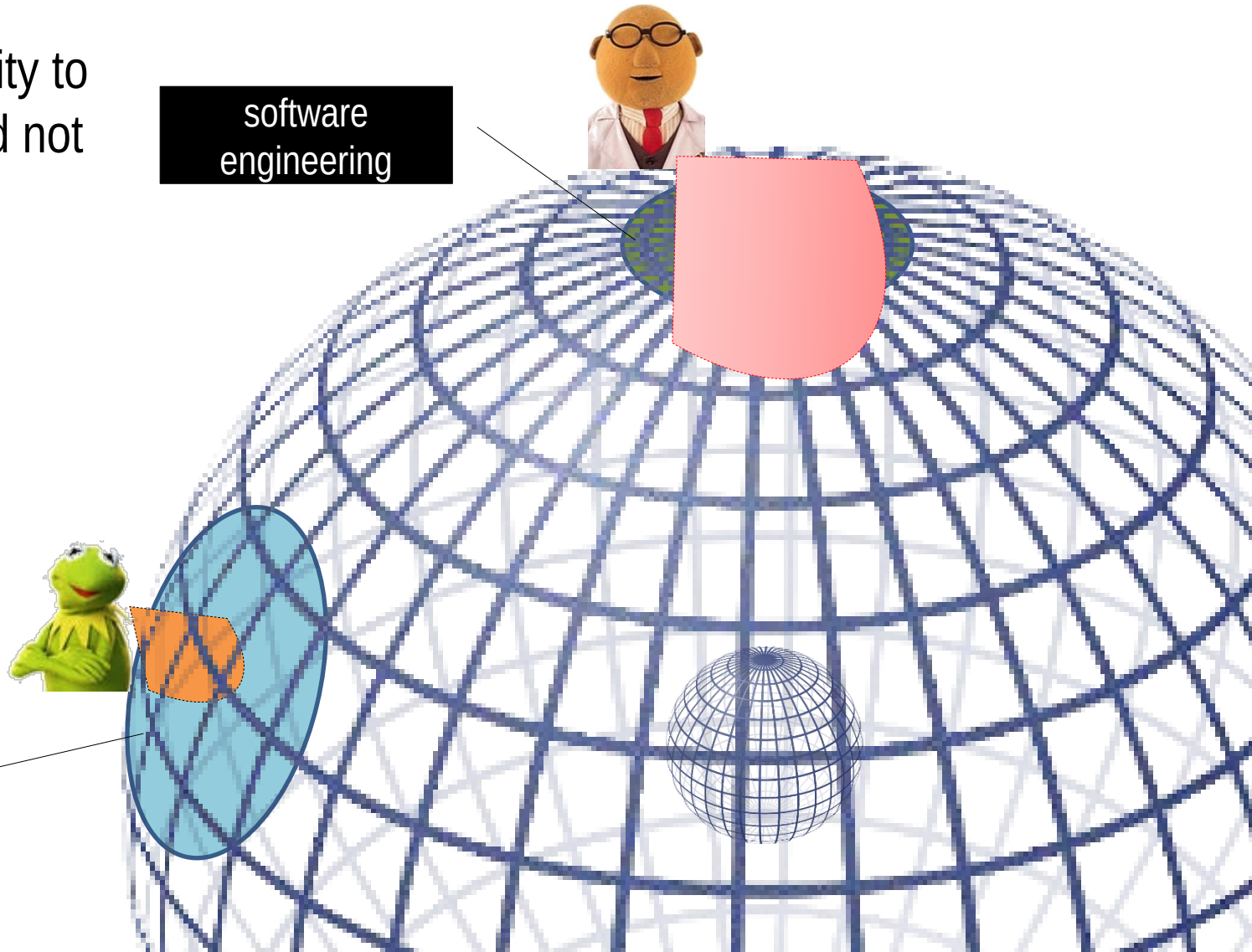
In the 60s-70s

- Some researchers kept up on advances far outside their area
- Dijkstra worked in Operating Systems, Distributed Computing, and Software Design!
- Others *virtually all of us* were near-sighted and happy



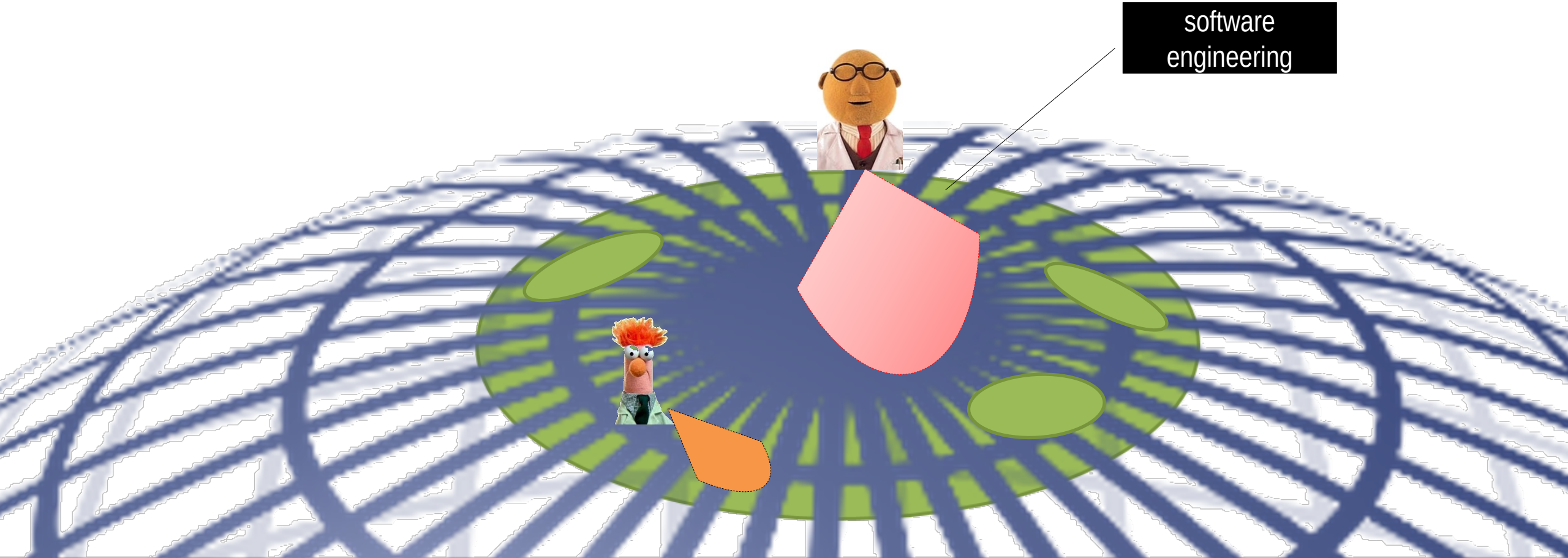
1980s-2000s

- Universe kept growing – but our ability to stay on-top of multiple disciplines did not scale
- Lose track of what others are doing and are typically unaware of their results
- Personal experience: ICSE'89



TODAY

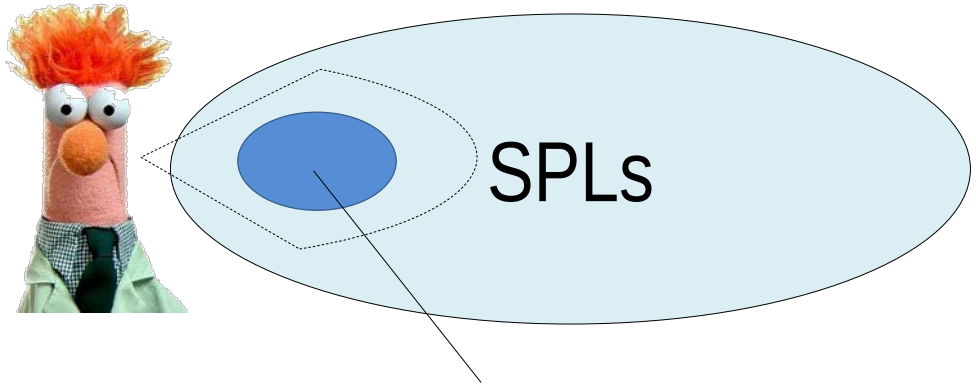
- Software Engineering is now so big that I can't see across it
- As time goes on, expect to see continued Balkanization….



Consequences

- Generally, researcher knowledge is becoming more specialized
  - gradual dilution of ‘big picture’ knowledge
  - recent SE grads or SPL experts think they know a lot, but what they know is shallow

- That was me! And maybe you too!



CONSEQUENCES

- Generally, researcher knowledge is becoming more specialized
  - gradual dilution of ‘big picture’ knowledge
  - recent SE grads or SPL experts think they know a lot, but what they know is shallow

- That was me! And maybe you too!
  - I was proud at how much I knew after my PhD 1980
  - 6 years later, I was amazed at how much more I had learned and how comparatively little I had known after my PhD
  - 6 years later, I was amazed at how much more I had learned, and looked forward to my next 6 years
  - … in short, I began appreciating progressively bigger universes of knowledge and their relationships… *my field of vision was increasing with time*

DIJKSTRA OBSERVATION

Have you noticed that there are child prodigies in mathematics, music, gymnastics –
but not in surgery?

Reason: surgery, like most areas of Science, are not innate. Their complexities must be learned and understood over long periods of time.

#2 IMPROVE REFEREEING AND #3 CONTINUE SUCCESS OF SPLS

#1 IMPROVE REFEREEING IN GENERAL

NOT SPECIFIC TO SPLC

- Pack program committee with real experts – or "experts" review the reviews
  1. referees who have actually programmed SPLs, not just written about it
  2. referees who appreciate theory AND practice
  3. open mind, not an axe to grind
  4. refereeing is NOT a blood sport; attitude should be a note written to a close colleague

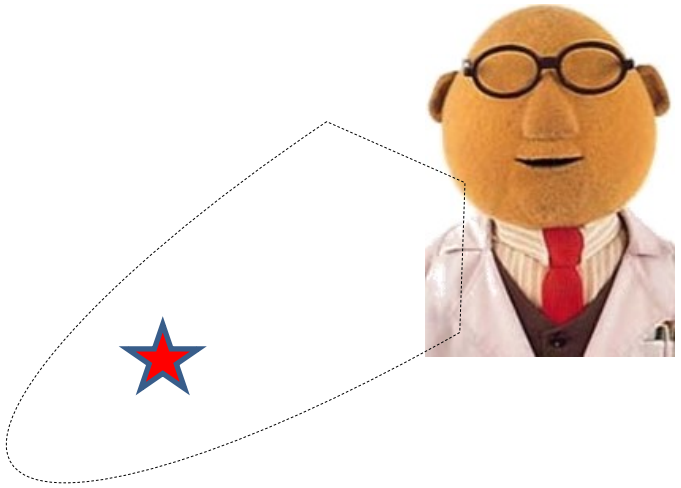
obvious

- WHY? Ultimate goal: attract more and better researchers to SPLs
  - don't discourage students having referees say "you didn't do it MY way"
  - build a sense of true scholarship in this area
  - results should be *good science* *more on this*

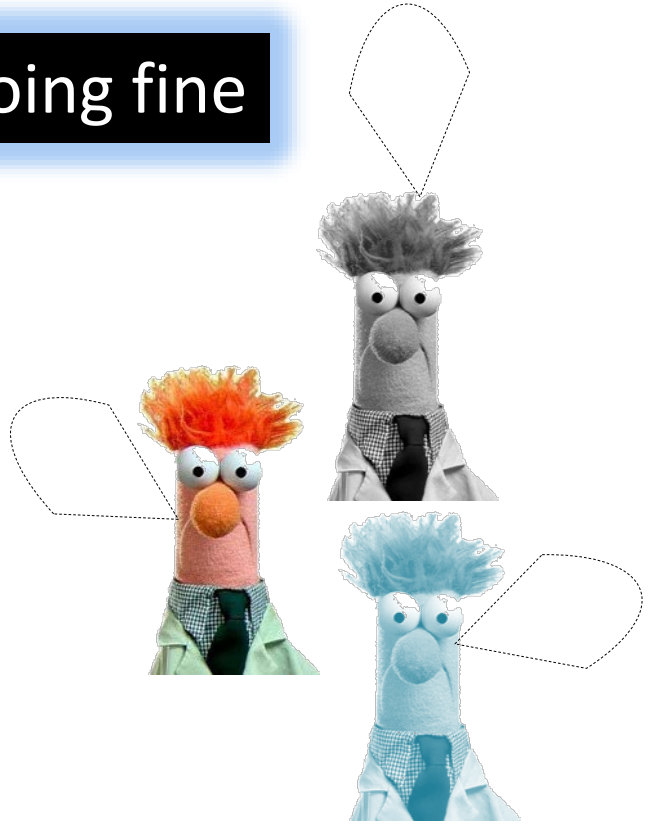
Step up our Game
More ideas offline

#2 WRITING PAPERS

- Common for “smaller problems” to be addressed
  - that’s what people know and understand, much less so than bigger-picture issues
  - easier to explain to others and have them appreciate it
  - because that’s what many of us are doing too…
  - common mistake for those who have a vision – doing too much in a single paper – referees won’t get it



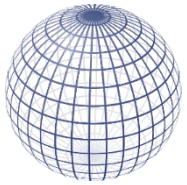
Doing fine



Visionaries of the Past

- Back in the late 60s-early 70s Software Science we had a “Newton”

Donald Knuth



- “Wow…”

Knuth's work is another way of saying

- Interesting, important, innovative work is between galaxies or worlds ★
- Progressively harder for referees to adequately judge such work

- What can I tell you?
- My greatest scientific achievements would not have been funded by NSF
  I had to build it before I would have been funded

Personal Story #1

- Last 4 years I focused on Refactoring
  - 2011 had an idea of how to improve refactoring engines
  - Danny Dig, THE person in refactoring today, said “I don’t believe it will work”
  - Fortunately he earlier asked me to be on a 4-year NSF project 2012-16 on adding transformations as 1st-class entities to Eclipse

- Fast forward to 2016: **It Works!**
  - CS undergraduates can write refactoring scripts – automate design patterns
  - complex script 5min execution time → down to 2 seconds, 141× factor increase
  - 6K LOC Java total, does not require 700K LOC Eclipse infrastructure other than compilation – don’t need a classical program transformation system to do this

PERSONAL STORY #1

- Last 4 years I focused on Refactoring
  - 2011 had an idea of how to improve refactoring engines
  - Danny Dig … will work”
  - Fortunatel… 2-16 on adding …

I WAS LUCKY TO BE IN THE RIGHT PLACE AT THE RIGHT TIME

- Fast forward to 20…
  - CS undergraduates can write refactoring scripts – automate design patterns
  - complex script 5min execution time → down to 2 seconds, 141× factor increase
  - 6K LOC Java total, does not require 700K LOC Eclipse infrastructure other than compilation – don’t need a classical program transformation system to do this

PERSONAL STORY #2

- In 1995 I was invited by Charles Simonyi to a workshop at MS on Intentional Programming
  - I had just completed GenVoca + was ½ way to AHEAD
  - First m… styles.
    Standa…
  - "If ever… about S…

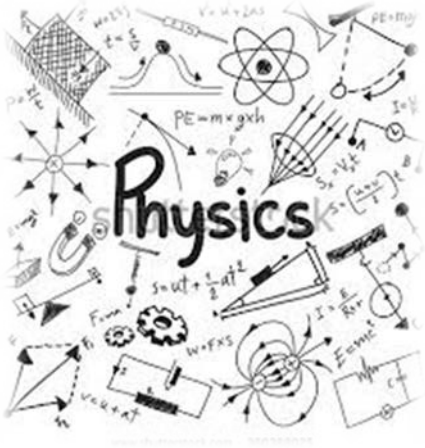
SE CONSIDERED HARMFUL

BECAUSE SE INTERESTS WERE ORTHOGONAL TO MAKE SPLS SUCCESSFUL

BALKANIZE

PREDICTION

- Today's SE / SPL papers are "simple" compared to what papers 20 years from today



- Compare DB papers (SIGMOD, VLDB) in 1986 to those of 2000 and today
    - vast difference in sophistication and use of mathematics

- If you only know SPL engineering, you'll be blown out of the water

MORE BACKGROUND TO ADDRESS ...

#1 WHY IS SOFTWARE SEEM TO BE GETTING WORSE ?

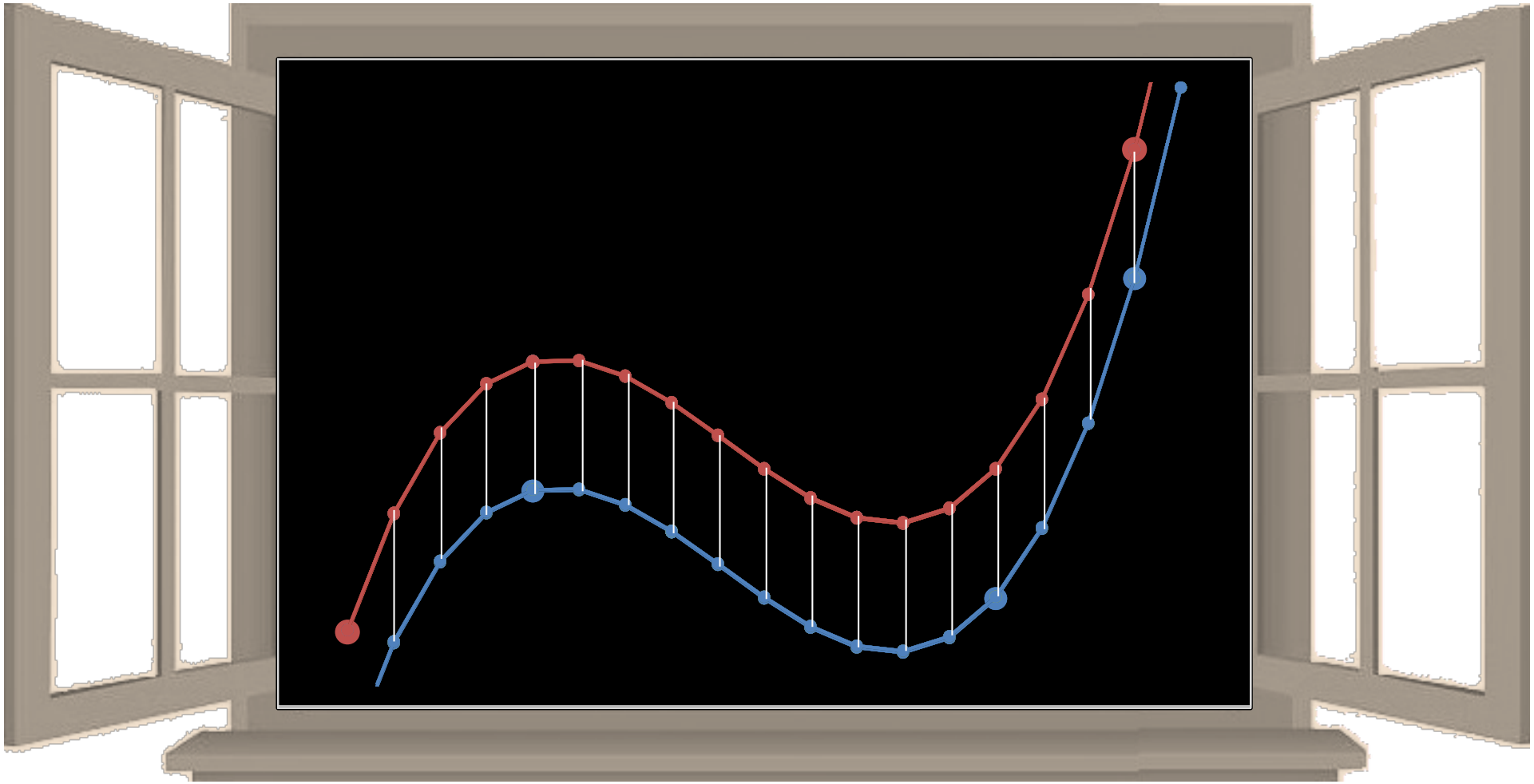
#3 WHERE MIGHT SPL RESEARCH BE IN 20 YEARS ?

WHAT DOES THE NIGHT SKY OF SOFTWARE SCIENCE LOOK LIKE ?

- Stars are results of humans
- Dim stars are perceived insignificant
- Bright stars automatically attract attention

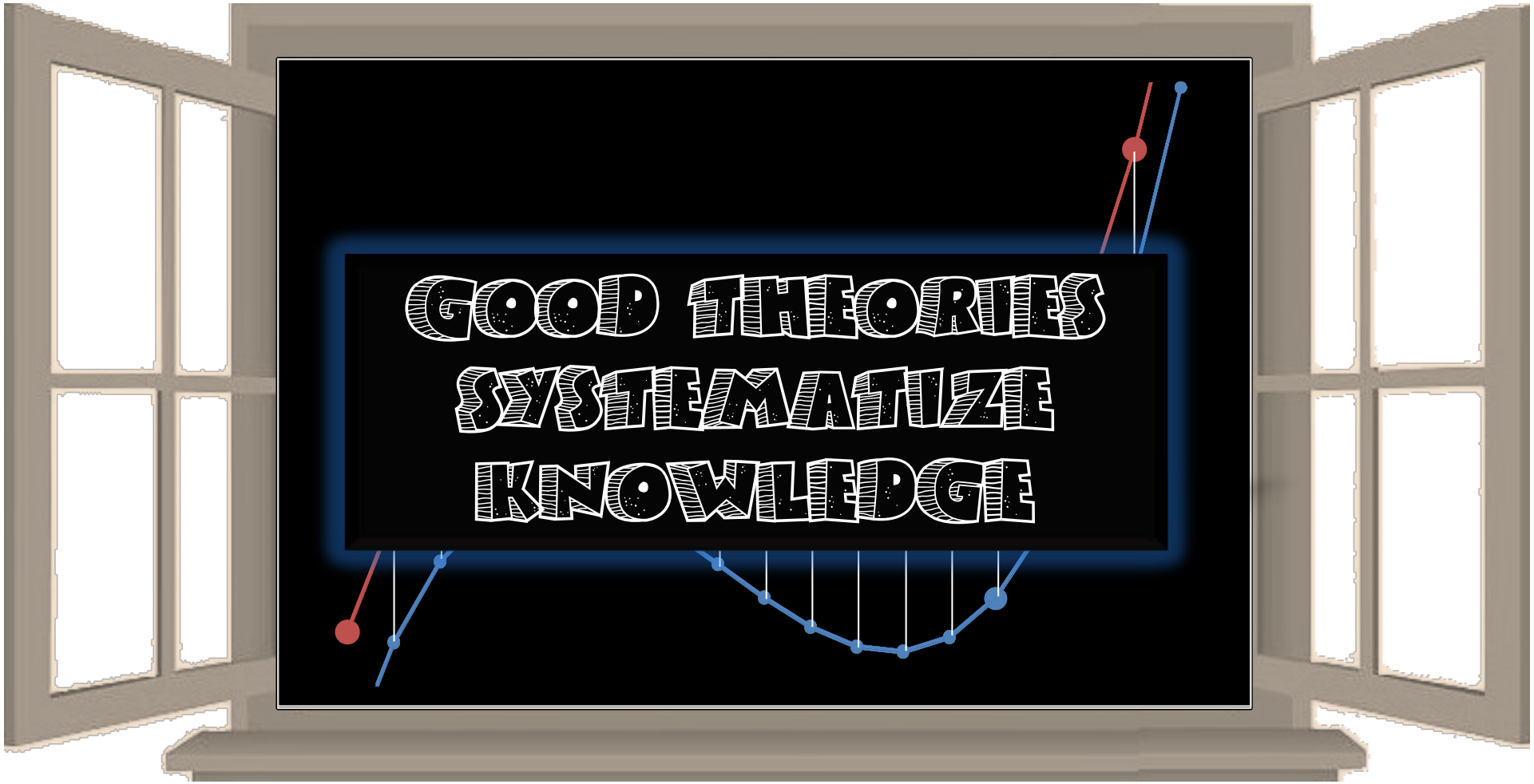
- Some results grow in significance over time
- Where others decline or disappear (AOP)
- Others sprout around key papers

- Just because you see stars *doesn't mean that the problems or results are fundamental*
  - could be – or they might be "fads" of no significance



- Look at deep space – where is nothing
- Incremental results soon populate sky
- Numbers increase over time

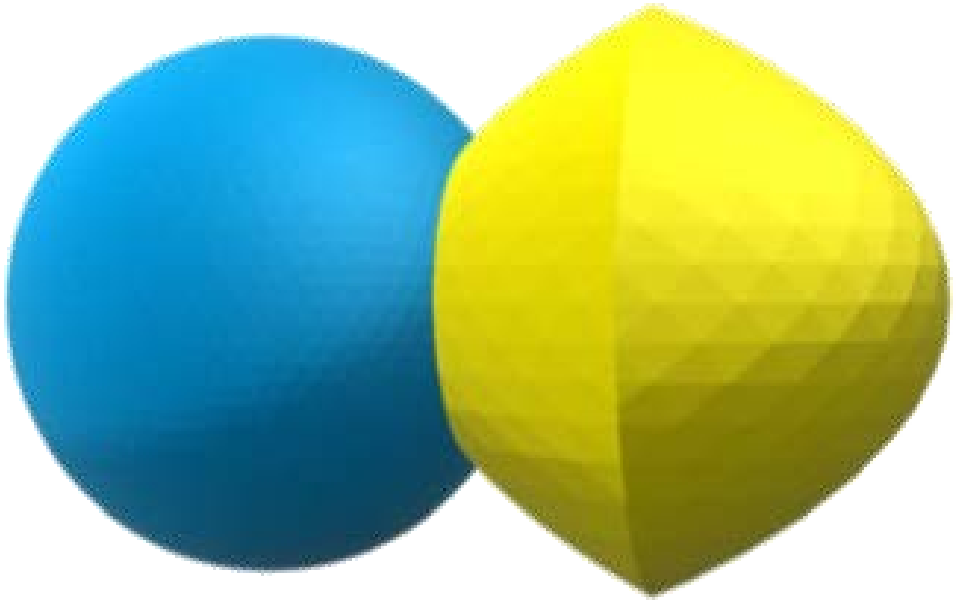
- Not how science proceeds *Pierre Schoebbens*
- Mature science abhors singleton results
- Theory reduces disparate results to a small set of axioms from which these and other results can be derived



- Look at deep space – where is nothing
- Incremental results soon populate sky
- Numbers increase over time

- Not how science proceeds *Pierre Schoebbens*
- Mature science abhors singleton results
- Theory reduces disparate results to a small set of axioms from which these and other results can be derived

Practical value of a good theory or



Applause for Graphics, Please

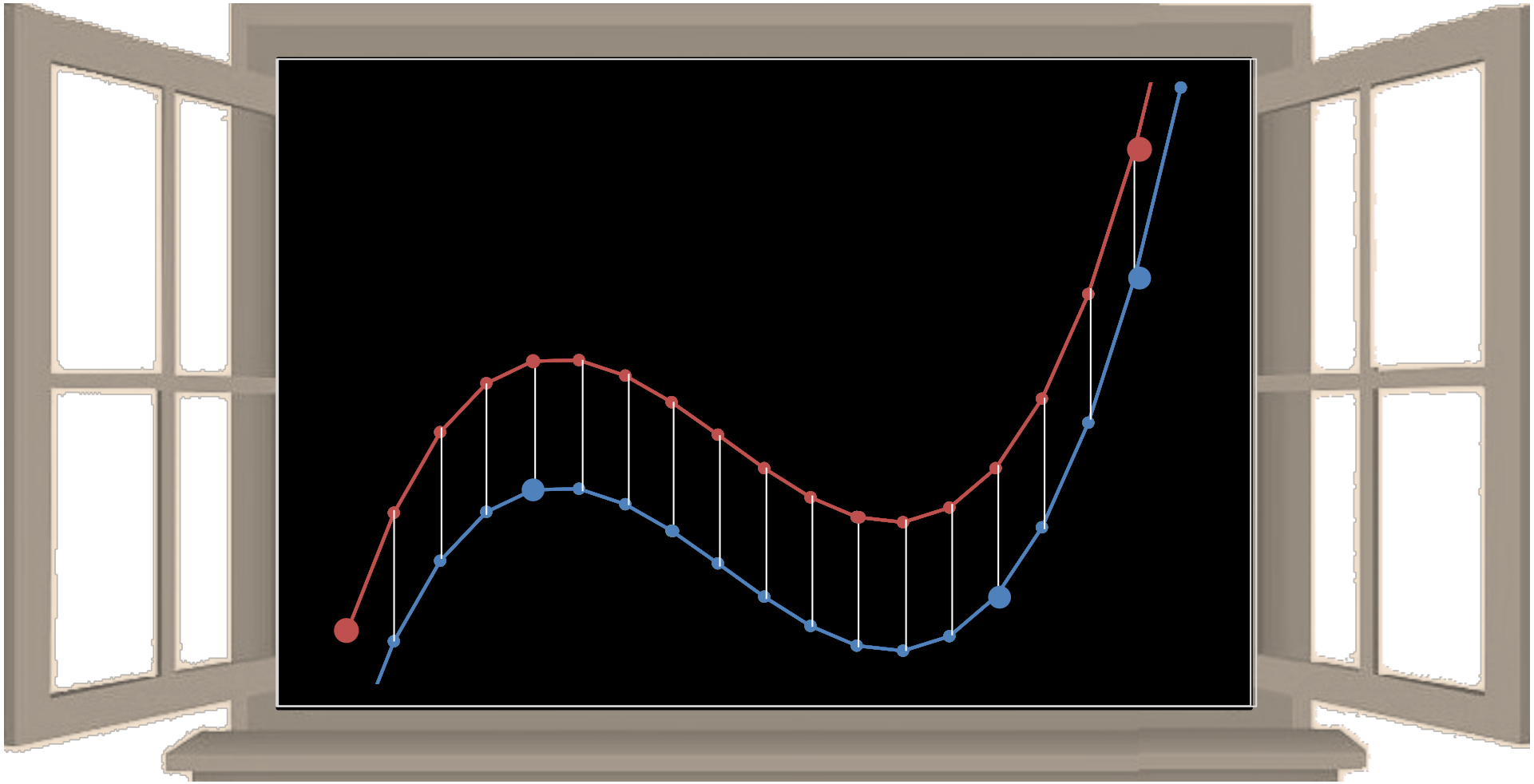
PRACTICAL VALUE OF A THEORY OR A TOOL BASED ON A GOOD THEORY OR EXPERIMENTS THAT LEAD TO GOOD THEORIES

GOOD THEORIES
GREATLY ACCELERATE
EXPANSION OF THE UNIVERSE

THIS IS WHAT A
TEST IN TIME AWARD
SHOULD RECOGNIZE

#1 WHY IS SOFTWARE SEEM TO BE GETTING WORSE ?

#3 WHERE MIGHT SPL RESEARCH BE IN 20 YEARS ?



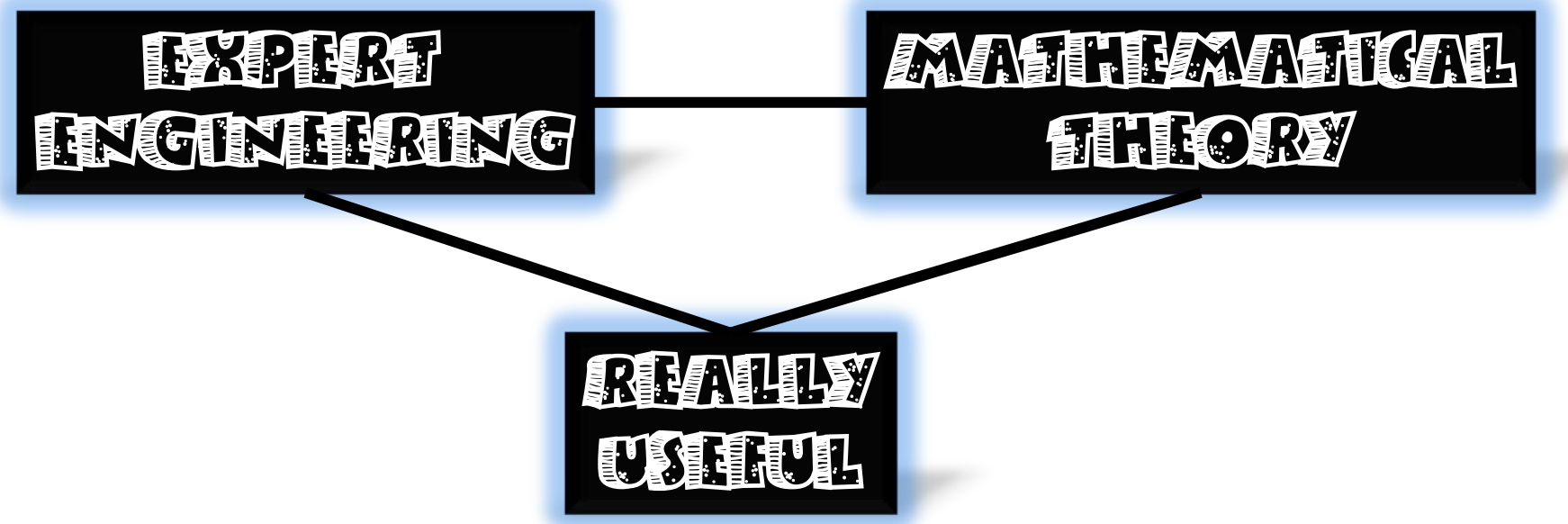
- The characteristic hallmark of great science – finding foundations for different phenomena

- This is also the characteristic hallmark of SPLs – finding foundations of families of systems

DISTILLING INSTANCES TO A THEORY

- Is clearer in SPLs than anywhere else in Software Science
  - ev… roof of concept of …
  - in… …ogramming
  - incremental step to understand verification in compositional programming

GRAND CHALLENGES IN SOFTWARE SCIENCE



There may be science in designing 1-of-a-kind systems … but

- How many times in last 30 years have you heard that the real gold is in the domain itself?
- SPL Modularity is fundamentally different than Object-Oriented modularity – Features

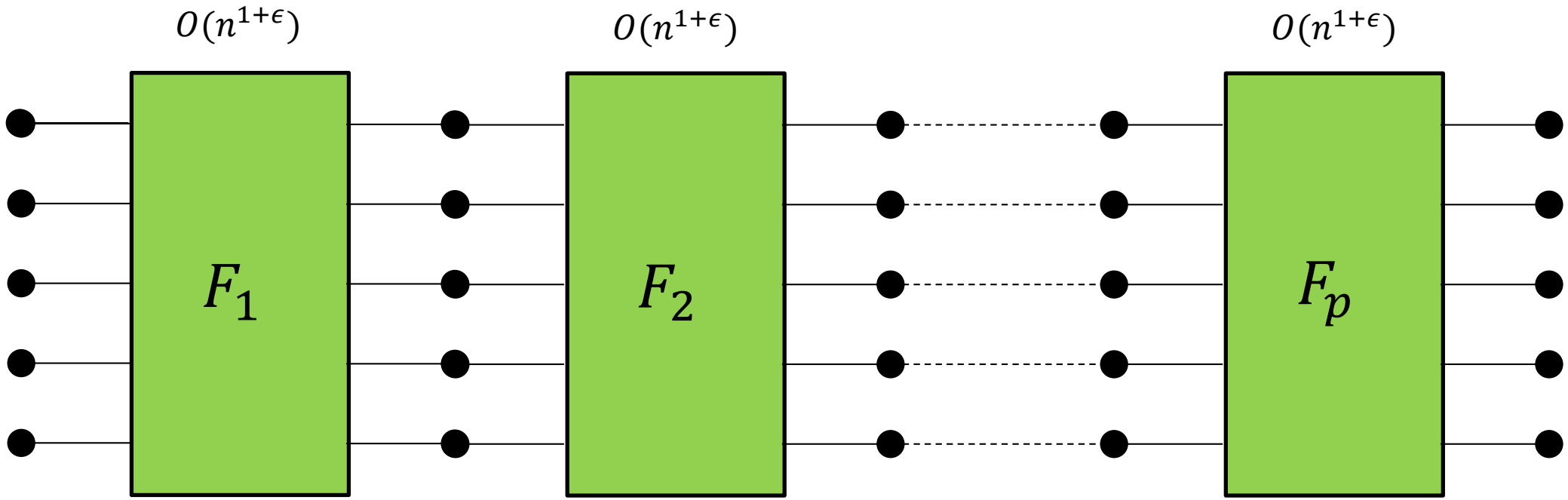
- SPL galaxy centers on building families of products incrementally one feature at a time
  - uninteresting to SPLs if only one product is built this way
  - small sub-universe is interesting to a vast majority of people in SE & CS, though

Not taking enough advantage of this

PRAGMATICS

cost of incrementally building a program is ***additive*** $O(p \cdot n^{1+\epsilon})$

- Of *incremental development,* building a program one feature *increment of functionality* at a time
- Each feature is at most $O(n^2)$ complexity ***more likely*** $\boldsymbol{O(n^{1+\epsilon})}$

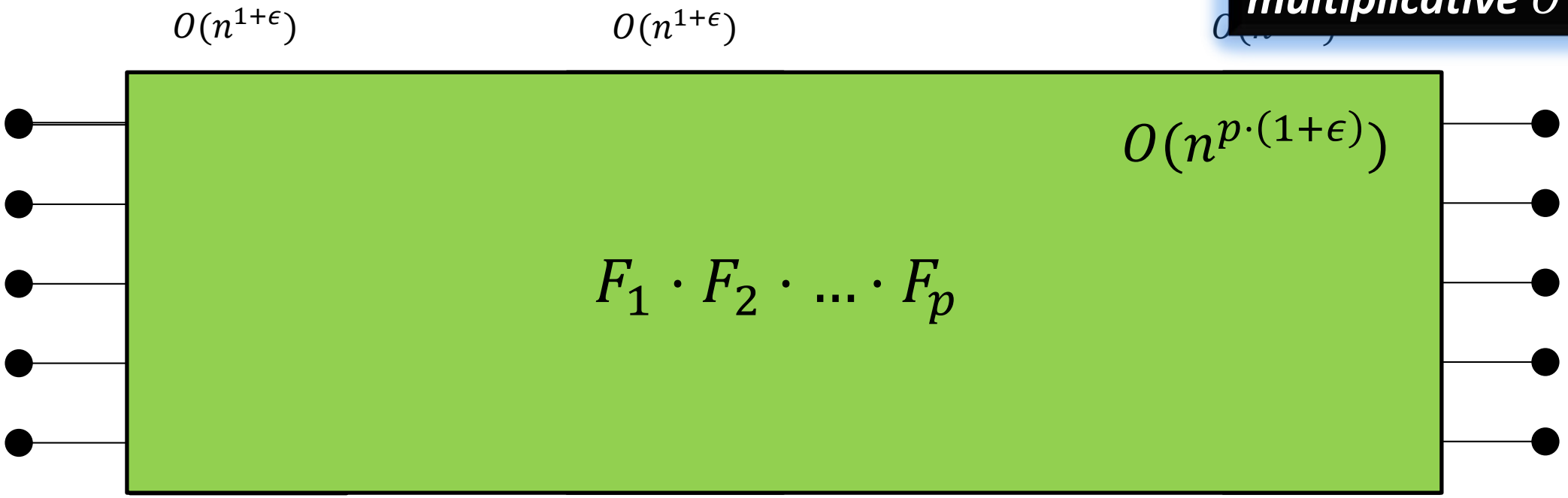


PRAGMATICS

cost of incrementally building a program is ***additive*** $O(p \cdot n^{1+\epsilon})$

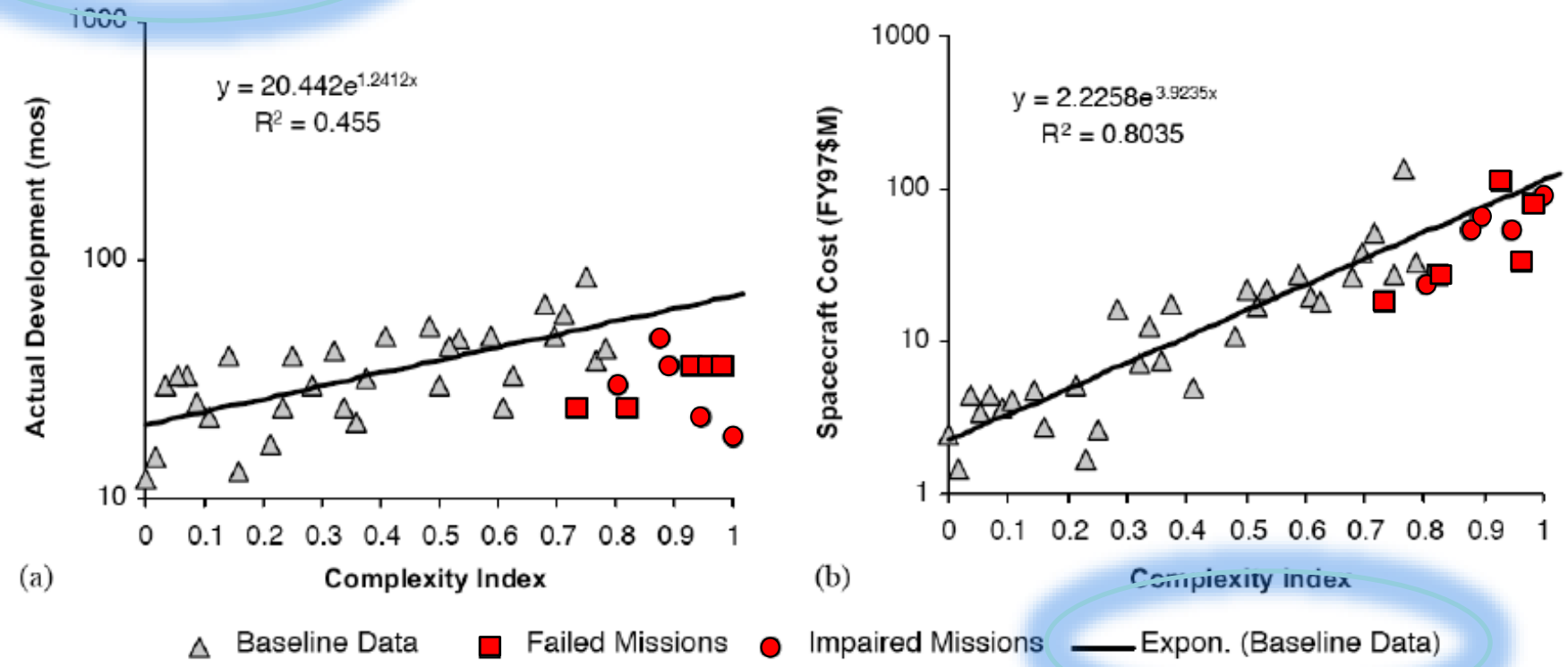
cost of building a monolithic program is ***multiplicative*** $O(n^{p\cdot(1+\epsilon)})$

- Of *incremental development,* building a program one feature *increment of functionality* at a time
- Each feature is at most $O(n^2)$ complexity ***more likely*** $\boldsymbol{O(n^{1+\epsilon})}$



HAVE YOU NOTICED...

- The complexity of software is growing at an ever-increasing rate?
  - low exponential in 2009 FSWC_final_report – most complex systems tend to fail

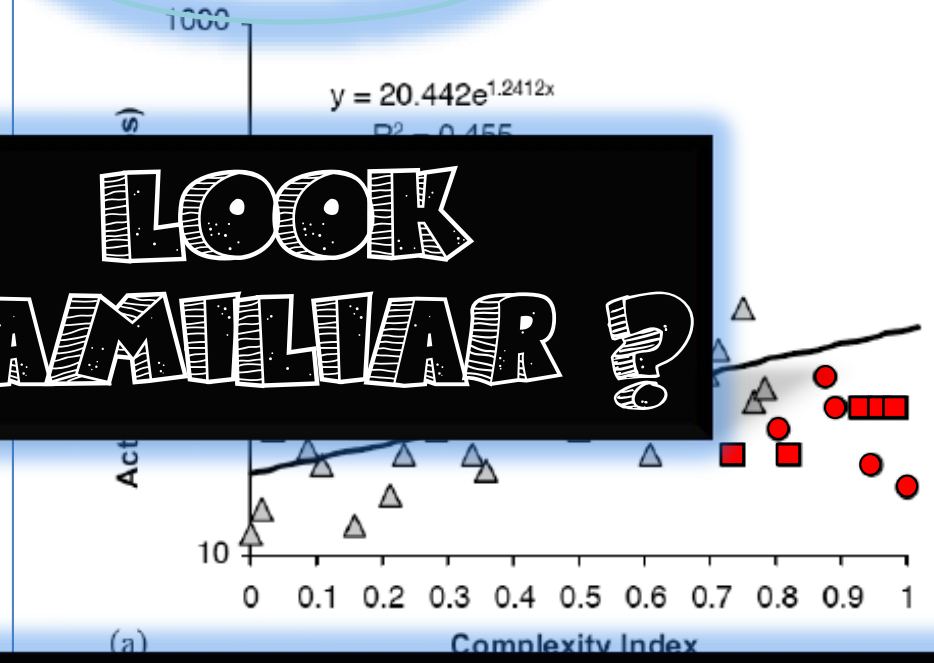


Cost and schedule as a function of complexity. (a) Schedule as function of complexity, (b) Spacecraft cost as function of complexity.

Figure 10: Impact of Complexity on Mission Cost and Success.

HAVE YOU OBSERVED…

- The complexity of software is growing at an ever-increasing rate?
  - low exponential in 2009 FSWC_final_



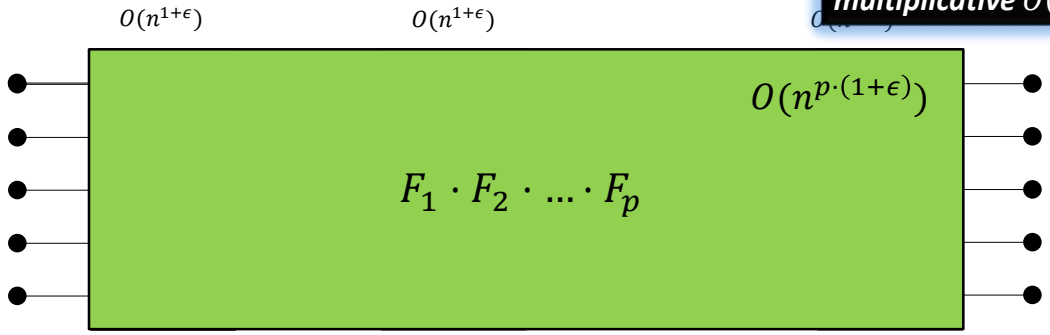
LOOK FAMILIAR ?

PRAGMATICS

- Of *incremental development,* building a program one feature *increment of functionality* at a time
- Each feature is at most $O(n^2)$ complexity ***more likely*** $O(n^{1+\epsilon})$

cost of incrementally building a program is ***additive*** $O(p \cdot n^{1+\epsilon})$

cost of building a monolithic program is ***multiplicative*** $O(n^{p\cdot(1+\epsilon)})$



Seville-51

PHENOMENON HAS A NAME …

Figure 10: Impact of Complexity on Mission Cost and Success.

FEATURITIS

Featuritis

The Wenger Giant Swiss Army Knife demonstrates why more features don't mean better usability.

FEATURITIS

Feature creep, creeping featurism or **featuritis** is the ongoing expansion or addition of new features in a product, especially in computer software and consumer and business electronics.

Feature creep - Wikipedia
https://en.wikipedia.org/wiki/Feature_creep

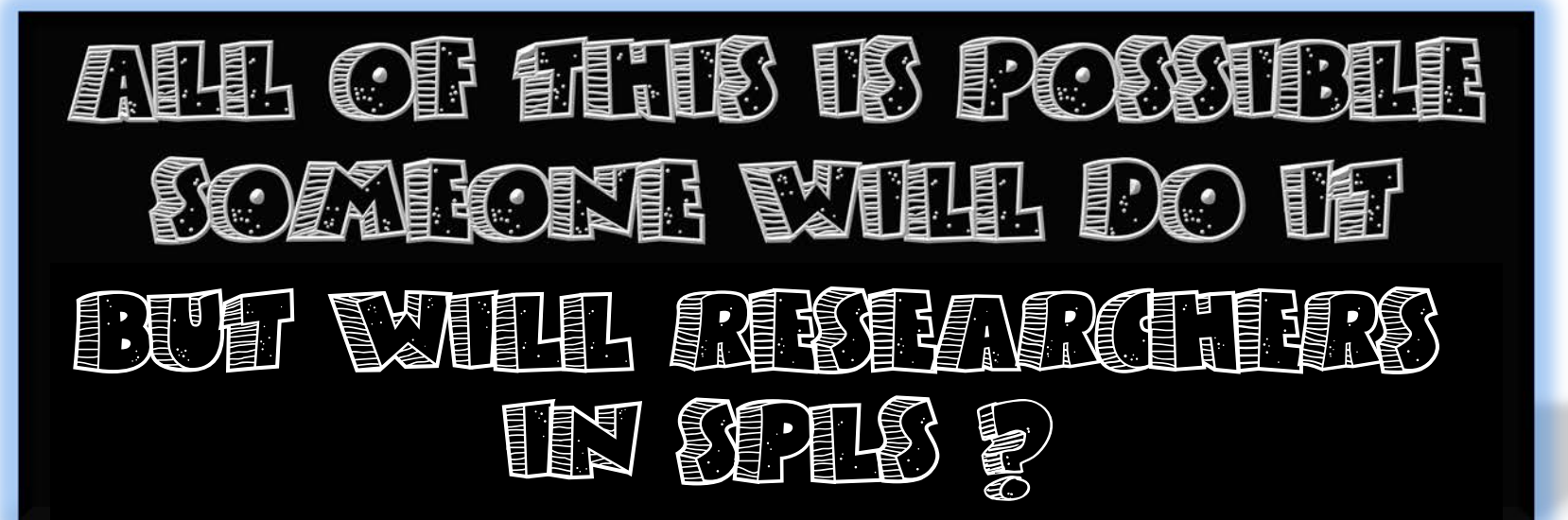
WE SHOULD BE THE EXPERTS ON HOW TO DEAL WITH FEATURITIS !!

SOFTWARE IS AWFUL BECAUSE PROGRAMMERS CAN'T UNDERSTAND CODEBASE WHOSE COMPLEXITY IS BEYOND THEIR COMPREHENSION

∞ A COROLLARY TO LIMITED FIELD OF VISION

SPLs can make a fundamental contribution to software science

- Why?
  - more advanced modularization
  - better control of program complexity
  - promote an important and practical form of compositional programming
  - lay groundwork for verification in compositional programming



Where are things going ?

- We know how to build SPLs (mostly) – and we know how to build products automatically – and we know the benefits of SPLs now.

What we are manually doing now in SPLs will be (semi) automated

- How do we **restructure legacy applications** (with tool support) into SPLs?
  - automate knowledge of program experts
  - how do we encode expert knowledge for machine application?

- How do we address **autonomic systems** – systems that know how to repair themselves?
  - must have some build-in notion of ‘feature model’ in these systems
  - how to encode knowledge of how to optimize/reconfigure a system automatically?

CONCLUDING REMARKS

Concluding Remarks

- I've sketched several futures of SPLs & SPLC that I think is possible

- I've made recommendations for the community to
    - reach out and get the best students/engineers that we can
    - remain practical and better appreciate role of theory
    - give our community a name for being visionary

- Because if we don't, we are in trouble...

Our work is cut out for us but it will be fun !

Concluding Remarks

- We've gone through a hyper-inflationary period of growth in Software Science
    - universe is now gargantuan in size and we can't see all of it or even a small part of

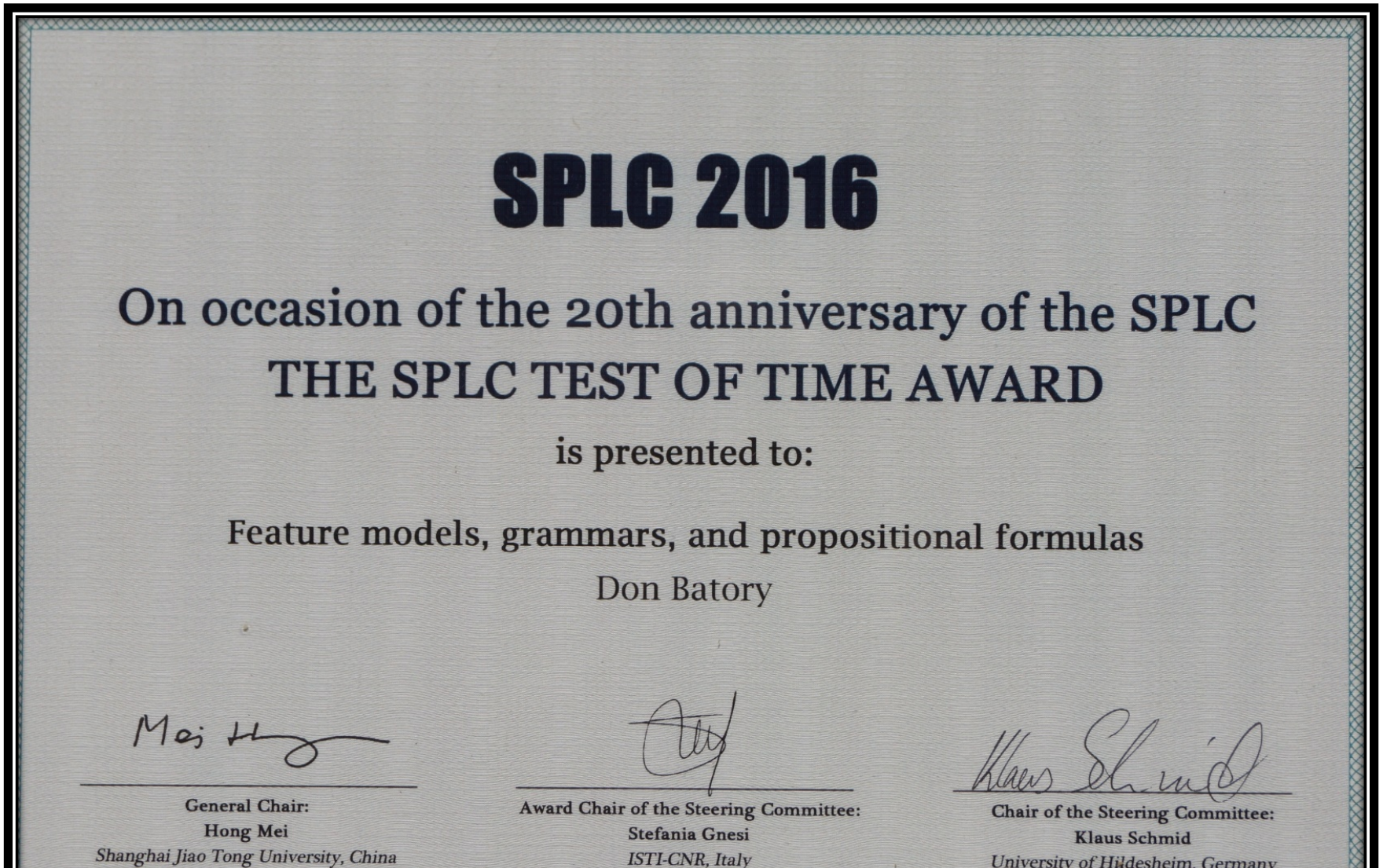
- Software Science is now more like traditional, mature science
    - now exhibits all the problems of a mature science

- There are things that SPL researchers can clearly do to improve the lot of many

Hope next generation can meet this challenge

THANK SPLC STEERING COMMITTEE!

For the 1st SPLC Test of Time Award!



SPLC 2016

On occasion of the 20th anniversary of the SPLC

THE SPLC TEST OF TIME AWARD

is presented to:

Feature models, grammars, and propositional formulas

Don Batory

General Chair:
Hong Mei
Shanghai Jiao Tong University, China

Award Chair of the Steering Committee:
Stefania Gnesi
ISTI-CNR, Italy

Chair of the Steering Committee:
Klaus Schmid
University of Hildesheim, Germany