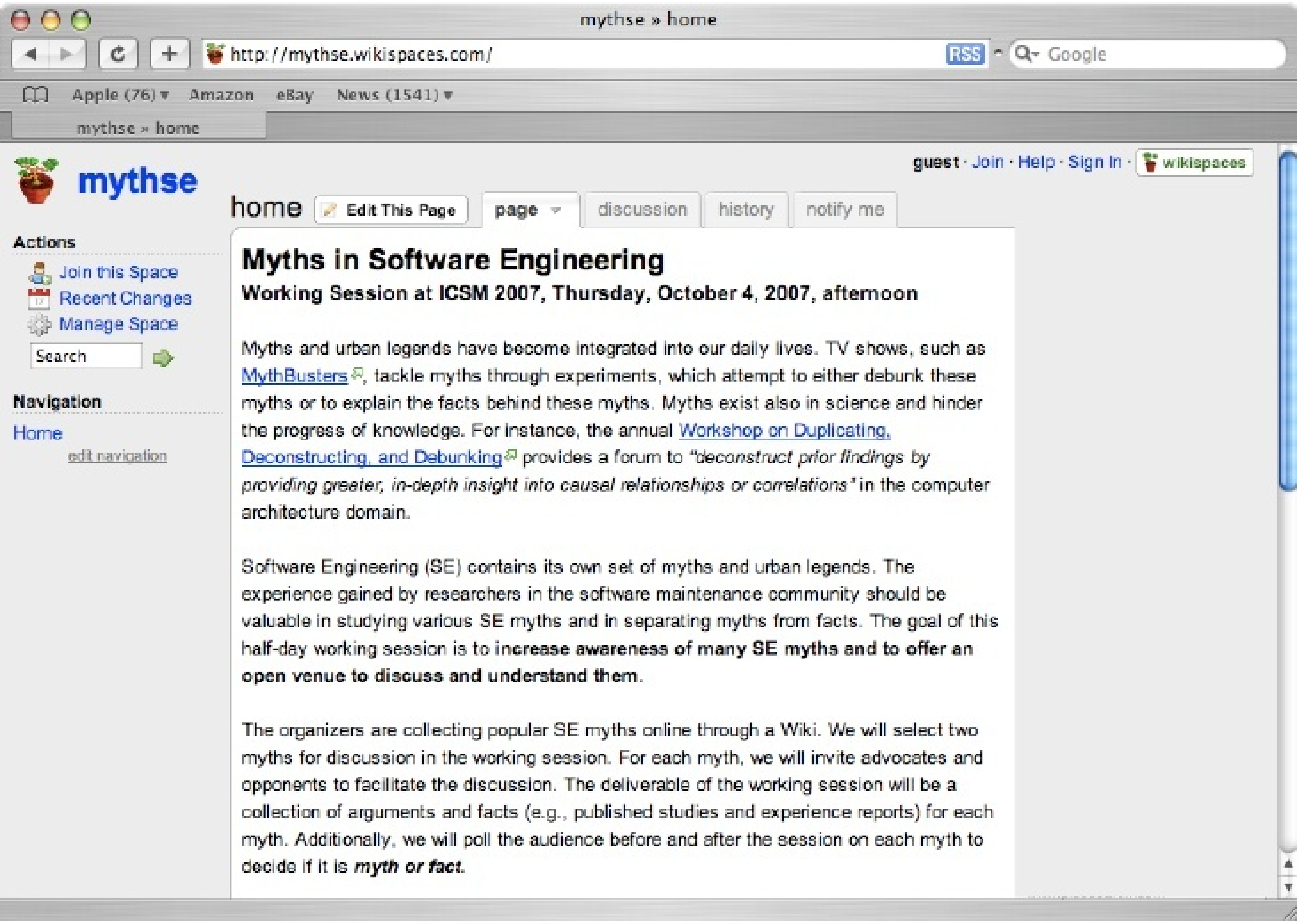
Got Myth?

ICSM Working Session
October 4, 2007, Paris

Tom Zimmermann, University of Calgary, Canada
Ahmed E. Hassan, Queen's University, Canada



Myths in Software Engineering

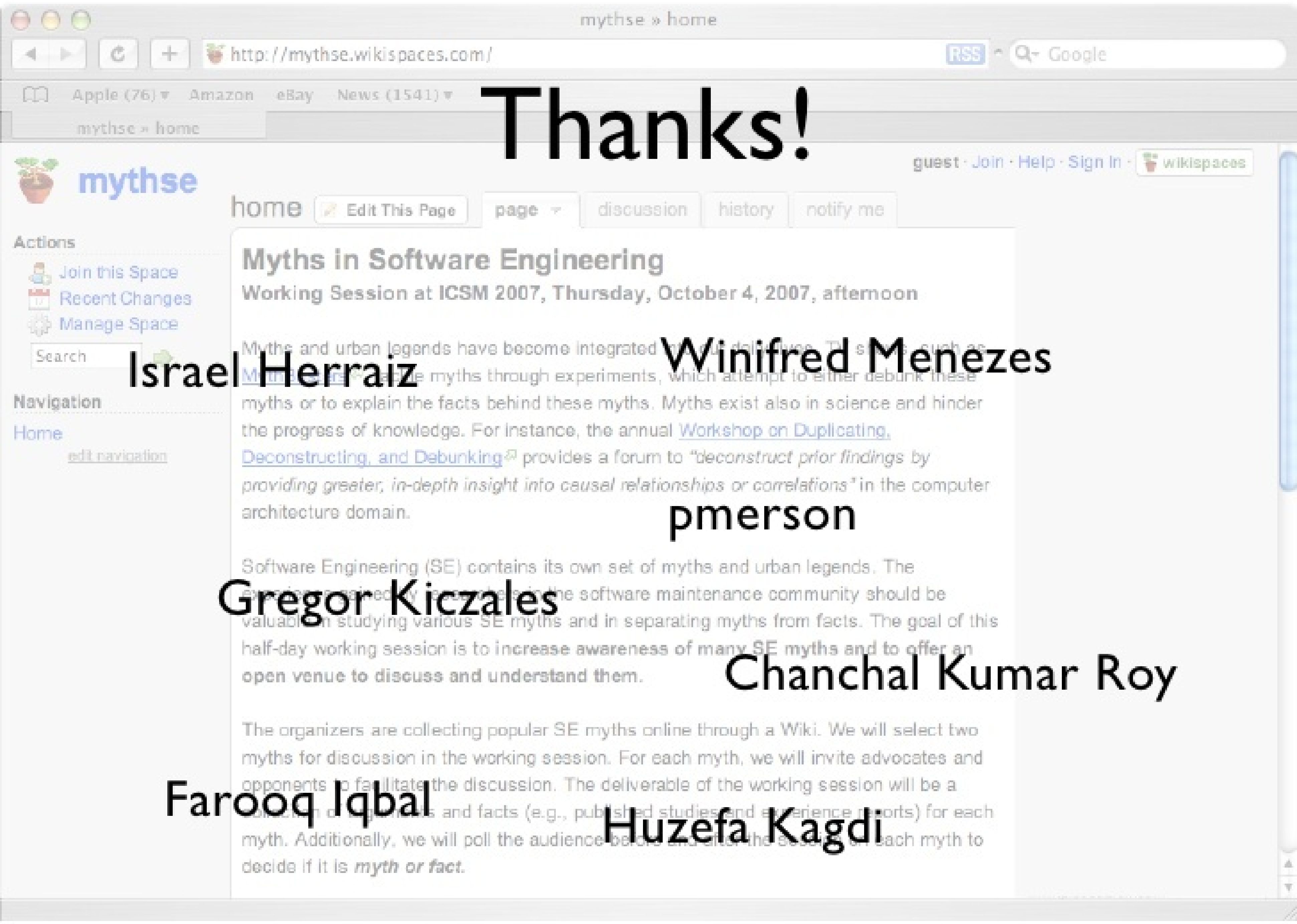
Working Session at ICSM 2007, Thursday, October 4, 2007, afternoon

Myths and urban legends have become integrated into our daily lives. TV shows, such as MythBusters, tackle myths through experiments, which attempt to either debunk these myths or to explain the facts behind these myths. Myths exist also in science and hinder the progress of knowledge. For instance, the annual Workshop on Duplicating, Deconstructing, and Debunking provides a forum to *"deconstruct prior findings by providing greater, in-depth insight into causal relationships or correlations"* in the computer architecture domain.

Software Engineering (SE) contains its own set of myths and urban legends. The experience gained by researchers in the software maintenance community should be valuable in studying various SE myths and in separating myths from facts. The goal of this half-day working session is to **increase awareness of many SE myths and to offer an open venue to discuss and understand them.**

The organizers are collecting popular SE myths online through a Wiki. We will select two myths for discussion in the working session. For each myth, we will invite advocates and opponents to facilitate the discussion. The deliverable of the working session will be a collection of arguments and facts (e.g., published studies and experience reports) for each myth. Additionally, we will poll the audience before and after the session on each myth to decide if it is ***myth or fact.***

Thanks!



Myths in Software Engineering

Working Session at ICSM 2007, Thursday, October 4, 2007, afternoon

Myths and urban legends have become integrated into our daily lives. TV shows, such as MythBusters, tackle myths through experiments, which attempt to either debunk these myths or to explain the facts behind these myths. Myths exist also in science and hinder the progress of knowledge. For instance, the annual Workshop on Duplicating, Deconstructing, and Debunking provides a forum to *"deconstruct prior findings by providing greater, in-depth insight into causal relationships or correlations"* in the computer architecture domain.

Software Engineering (SE) contains its own set of myths and urban legends. The experience gained by researchers in the software maintenance community should be valuable in studying various SE myths and in separating myths from facts. The goal of this half-day working session is to **increase awareness of many SE myths and to offer an open venue to discuss and understand them.**

The organizers are collecting popular SE myths online through a Wiki. We will select two myths for discussion in the working session. For each myth, we will invite advocates and opponents to facilitate the discussion. The deliverable of the working session will be a collection of arguments and facts (e.g., published studies and experience reports) for each myth. Additionally, we will poll the audience before and after the session on each myth to decide if it is *myth or fact.*

Israel Herraiz

Winifred Menezes

pmerson

Gregor Kiczales

Chanchal Kumar Roy

Farooq Iqbal

Huzefa Kagdi

Schedule

14:00-14:20	Intro & Wiki News
14:20-15:30	The Clone War Begins
15:30-16:00	Break
16:00-16:45	Bugs! Bugs! Bugs!
16:45-17:15	Top 100 Myths
17:15-17:30	Future of Myths in SE

Invited talks

Invited talks

Mike Godfrey

Andreas Zeller

Bugs reside in complex code.

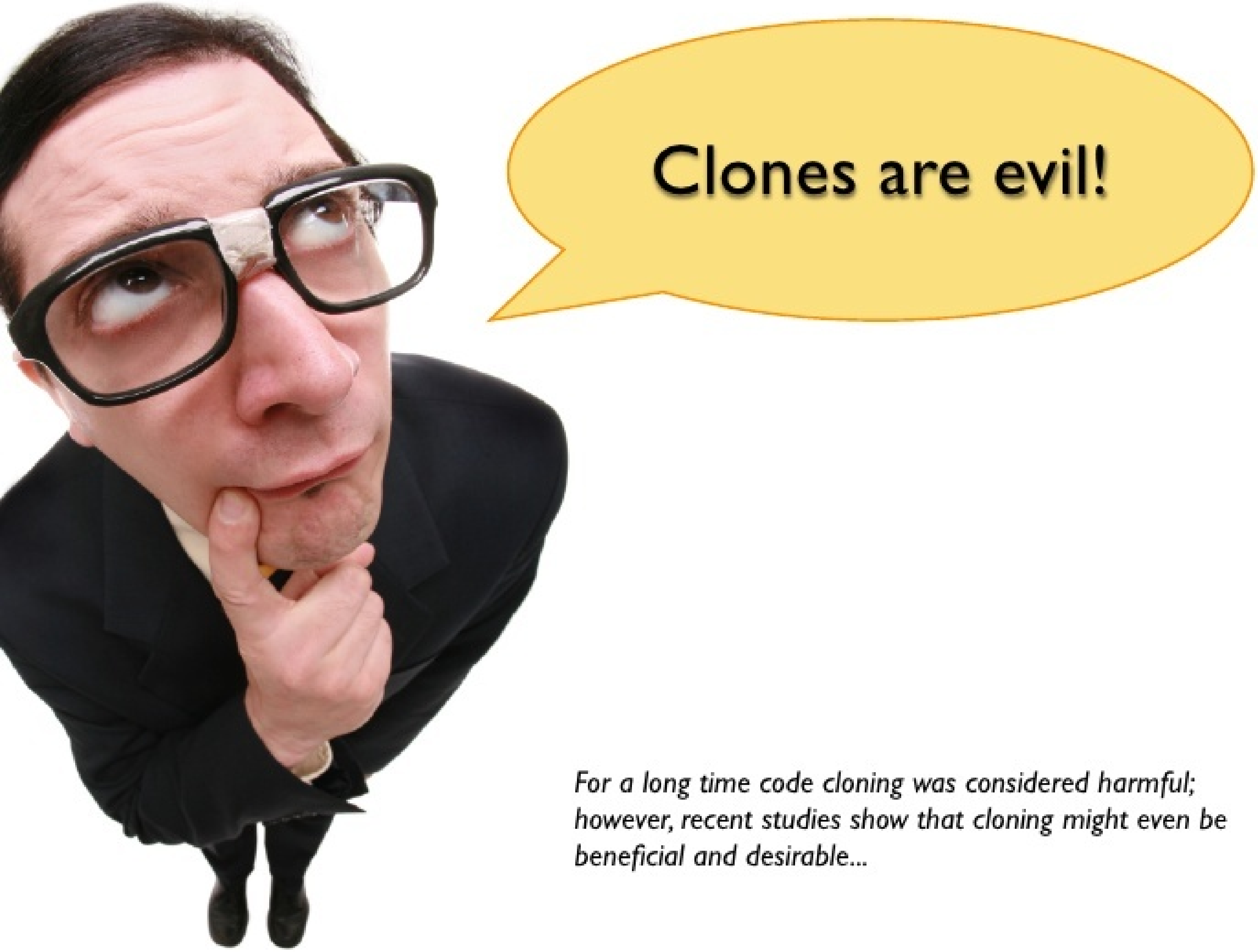
In the quest for metrics that predict bugs, many tools report various code complexity metrics; however, recent studies show that most complexity metrics correlate with just LOC. Is it really complexity that makes programs fail?

Bugs reside in complex code

- In an 80s study on a ~90KLOC program for satellite planning, Basili and Perricone noted that
  - The larger modules (size in LOC) were less error prone
  - Error-prone modules did not have higher (cyclomatic) complexity than error-free modules. In fact, the defect rate of modules decreased with the increase in size and complexity
  - These results were quite surprising. So, bugs reside in complex: a myth or not a conclusive fact yet that needs further establishment?
- In 2007 Herraiz and colleagues showed that
  - LOC and SLOC are highly correlated with classical complexity metrics for the case of the C programming language
  - The study was performed on a sample of 700,000 files, obtained from all the software packages included in FreeBSD

Bugs reside in complex code

- I guess it depends on what is meant by complexity - many paths through the code, many jumps backward and forward, many loops. Clearly the more LOC - the more possibility there is for any or all of the above
- I have done research on software defect models. I did a survey and most software people think that complexity is a good predictor of defects.
  - But some analysis done with defect data of various projects revealed to me that complexity (cyclomatic complexity) is not correlated with defects. I think software defects depend upon multiple factors out of which one can be 'complexity'.



For a long time code cloning was considered harmful; however, recent studies show that cloning might even be beneficial and desirable...

Clones are evil!

- Cordy reports that removing clones increases risk for large systems
- Kim et al. examine the evolution of clones and conclude that clone refactoring would not help many of the long lived clones
- Kapser and Godfrey describe several patterns of clonning and discuss their benefits for the long term evolution of software systems

VACUUM PUMP
PRESSURE BOX
Pascal = N
Crack
MOMENT OF IMPACT
MYTHBUSTERS

Clones are evil!

- Aversano et al. study co-change analysis and show that most of the cloned code is consistently maintained, particularly while bug fixing of cloned fragments. Their study indirectly shows that clones may not be harmful to software maintenance
- A prototype tool, CloneTracker, has been developed by Lozano et al. to evaluate the harmfulness of cloning. Although their case study intends to show that clones are harmful, they were unsure of the results

Clones are evil!

- Monden et al. conducted an experiment for evaluating the relation between clones and software quality attributes, namely reliability and maintainability with industrial systems. Their study shows that
  - modules/files with code clones are on average 1.7 times more reliable than files/modules without code clones
  - However, modules/files with larger clones are less reliable than others
- In the case of maintainability quality attribute, their results also support the usual hypothesis that clones have a bad impact on software maintenance
- Koschke has a survery of state of research in cloning

Aspect-oriented programming seems to be a story of successes; however, after ten years of active research (including its own conference), it is not clear whether aspect-oriented programs are any easier to maintain than traditional programs.

Aspect-oriented programs are easy to maintain

- No rational AOP proponent would stand by this claim
- For one thing, a badly written AOP program is not easy to maintain. Moreover, a program that really needs AOP is an inherently more complex thing than a program that really doesn't need AOP — so there's an apples vs. oranges error to be careful for there
- In the end, the only reasonable claim one could make about AOP is that
  - When a system's functionality inherently has crosscutting concerns, proper use of AOP leads to an implementation that is easier to maintain than the implementation one would get without AOP
- Slides for AOP myths and realities

The mythical man month

Many software project managers still believe that men and months are interchangeable commodities in software project schedules.

Mythical Man Month

- Project managers use the number of men and months as interchangeable units for measuring the project effort.
- "The number of men and months are interchangeable measures in software projects" is a myth, Brooks 1975
- McConnell argues that Brooks' Law is less pervasive than it appears
- Today
  - Many managers still ignore the myth and add staff to software projects running behind schedule.
  - Coordination issues amplified with global software development. Geographically distributed teams suffer with the lack of face-to-face communication

Clones

- What is a clone?
  - Task oriented definition of a clone (can I refractor it)
  - Intent matters (you meant it)
  - Find problematic code patterns instead
- Effect of clones in the long term:
  - Refactoring is challenging but we provide tool support.
    - Should we provide clone support
  - Awareness is very important
- How can we show that clones are good/bad?
  - What type of studies are needed
    - Scale/Industry vs. open source
- What are bad clones?
  - Any patterns of bad clones

Clones

- Where should research go?
  - New clone detection tools…
  - Study the long term effects of clones
  - Clone maintenance tools
- How can we integrate clones into practice?
  - Clones as a project management/release tool
  - Clones as a way of coding (the new reuse!)

Clones

- How can we show that clones are good/bad?
  - What type of studies are needed
    - Scale/Industry vs. open source
- What are bad clones?
  - Any patterns of bad clones
- Where should research go?
  - New clone detection tools
  - Study the long term effects of clones
  - Clone maintenance tools
- How can we integrate clones into practice?
  - Clones as a project management/release tool
  - Clones as a way of coding (the new reuse!)

Bugs! Bugs! Bugs!

- Watch out for confounding factors. It’s easy to be tricked.
- History gives mostly correlations, rarely causation.
- Are we using the wrong metrics?
  - OO complexity is rarely captured
- Certain kinds of bugs matter more.
- Different models for different kind of bugs?
- Bugs are some form of disease.

- Where should research go?
  - Better complexity metrics
  - More (controlled) experiments
  - Insure software against bugs (new business idea?)

True or False?

+

Using a cell phone while pumping gas
can cause an explosion

Your myths?!

1. Software complexity/maintenance cost can be calculated by measuring structural complexity
2. **AOP promotes modularity**
3. All new requirements can be implemented in an existing codebase
4. Overtime in necessary and useful
5. Cutting QA will save us money
6. **Open source is better quality than proprietary software**
7. Linux will become un-maintainable because of high internal coupling
8. **Service orientation will lead to better (more maintainable) systems**
9. Open source applications are mainly developed by people working in their free time on a voluntary basis and without being paid for that
10. Aspects != cross-cutting concerns
11. Open source software development process is different from industrial projects

12. Software testing is tedious but not hard (Not worth teaching in universities)
13. **Linux is a representative open source project**
14. **Requirements are not useful in open source projects**
15. Software visualization is helpful
16. **C++ is faster than Java**
17. UIs can be separated from their application
18. **Design patterns make better code**
19. Quality is free
20. **God classes are a problem**
21. Software maintenance (is software really “maintained”)
22. OO systems are easier to maintain than procedural
23. Gotos are harmful
24. Features added to Java are novel and not invented in the 70s.
25. Good software can only be built with good requirements
26. **Good tools make good programmers**

27. Good software cannot be built with a weakly typed language (like pythons)
28. Complexity of code will soon become unmanageable
29. **Cobol is dead**
30. It is not possible for a query language to be seriously logical and seriously object oriented at the same time
31. **XML will solve world hunger problems!**
32. **Legacy=bad**
33. Macs are better than PCs

MYTHBUSTERS

Vote BUSTED

or CONFIRMED

(The following slides report what was voted by the majority of MythSE participants.)

MYTHBUSTERS

AOP promotes modularity

BUSTED

MYTHBUSTERS

Open source is better quality than proprietary software

BUSTED

MYTHBUSTERS

Service orientation will lead to better (more maintainable) systems

BUSTED

MYTHBUSTERS

Linux is a representative open source project

BUSTED

MYTHBUSTERS

C++ is faster than Java

CONFIRMED

True or False?

+

Using a cell phone while pumping gas
can cause an explosion

MYTHBUSTERS

Design patterns make better code

BUSTED + CONFIRMED

(This one was a tie)

MYTHBUSTERS

God classes
are a problem

CONFIRMED

MYTHBUSTERS

Good tools make good programmers

BUSTED

MYTHBUSTERS

Cobol is dead

BUSTED

MYTHBUSTERS

Legacy = bad!

BUSTED

Future of Myths in SE?

- Another working session?
- Workshop on Negative Results?

Got Myth!

Update the MythSE Wiki:
mythse.wikispaces.com

Myths in SE?

Myths in SE?

Bugs reside in complex code

Myths in SE?

Bugs reside in complex code

Clones are evil!

Myths in SE?

Bugs reside in complex code

Aspect-oriented programs are easy to maintain

Clones are evil!

Myths in SE?

Bugs reside in complex code

Aspect-oriented programs are easy to maintain

Mythical man month

Clones are evil!