

Program Verification: Theory and Practice

Sriram K. Rajamani
Microsoft Research India

(with thanks to Tom Ball for material from his
course)

Organization

- Instructors
 - Deepak D'Souza, IISc
 - Aditya Nori, MSR India
 - Sriram K. Rajamani, MSR India
- Teaching Assistants
 - Arnab De & Raghavendra, IISc

PROBLEM

Software validation problem

Does the software work?

Software validation problem

I hope some hacker cannot steal all my money, and publish all my email on the web

I hope it doesn't crash!

Does the software **work**?

I hope it can handle my peak transaction load

I hope it still interoperates with my other software in the same way as the previous version!

How do we do software validation?



Testing:

- The “old-fashioned” way
- Run it and see if it works
- Fix it if it doesn’t work
- Ship it if it doesn’t crash!

What is wrong with testing?

program correctness. Today a usual technique is to make a program and then to test it. But: program testing can be a very effective way to show the presence of bugs, but is hopelessly inadequate for showing their absence. The only effective way to raise the confidence level of a program significantly is to give a convincing proof of its correctness. But one should not first make

The Humble Programmer.

by

Edsger W.Dijkstra

As a result of a long sequence of coincidences I entered the programming profession officially on the first spring morning of 1952 and as far as I have been able to trace, I was the first Dutchman to do so in my country. In retrospect the most amazing thing was the slowness with which, at least in my part of the world, the programming profession emerged, a slowness which is now hard to believe. But I am grateful for two vivid recollections from that period that establish that slowness beyond any doubt.

Program Verification

The algorithmic discovery of *properties* of a program by *inspection* of the *source text*

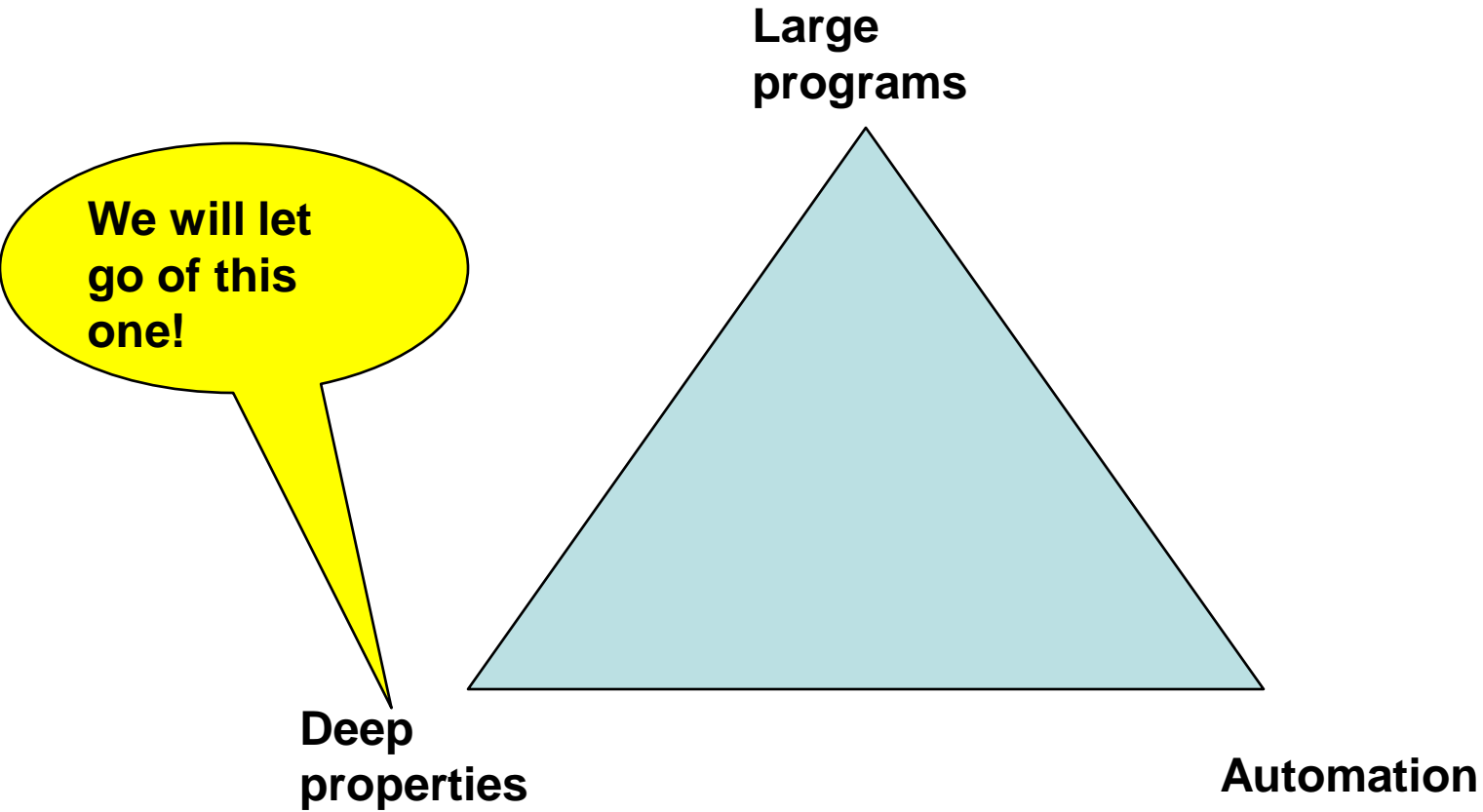
- Manna and Pnueli, “Algorithmic Verification”

Also known as: static analysis, static program analysis, formal methods,....

Difficulty of program verification

- What will you prove?
 - Specification of a complex software is as complex as the software itself
- “Deep” specifications of software are hard to prove
 - State-of-art in tools and automation not good enough

Elusive triangle




```
void Foo( int * ptr, int const * ptrToConst, int * const constPtr,  
         int const * const constPtrToConst ) {
```

```
*ptr = 0;  
ptr = 0;
```

```
*ptrToConst = 0;  
ptrToConst = 0;
```

```
*constPtr = 0;  
constPtr = 0;
```

```
*constPtrToConst = 0;  
constPtrToConst = 0;  
}
```

```
void Foo( int * ptr, int const * ptrToConst, int * const constPtr,  
         int const * const constPtrToConst ) {
```

```
*ptr = 0; // OK: modifies the pointee
```

```
ptr = 0; // OK: modifies the pointer
```

```
*ptrToConst = 0; // Error! Cannot modify the pointee
```

```
ptrToConst = 0; // OK: modifies the pointer
```

```
*constPtr = 0; // OK: modifies the pointee
```

```
constPtr = 0; // Error! Cannot modify the pointer
```

```
*constPtrToConst = 0; // Error! Cannot modify the pointee
```

```
constPtrToConst = 0; // Error! Cannot modify the pointer
```

```
}
```


<http://www.microsoft.com/whdc/devtools/tools/sdv.mspx>

http://en.wikipedia.org/wiki/Microsoft_Platform_SDK

<http://www.gotdotnet.com/team/fxcop/>

<http://research.microsoft.com/specsharp/>

http://en.wikipedia.org/wiki/Worse_is_Better

Worse is better, also called the **New Jersey style**, is the name of a [computer software design](#) approach (or [design philosophy](#)) in which simplicity of both [interface](#) and implementation is more important than any other system attribute (including correctness, consistency, and completeness).

http://en.wikipedia.org/wiki/Robert_Tappan_Morris



Today on
CNET

News

Reviews

Compare
prices

Tips &
Tricks

Today on News | **Business Tech** | Cutting Edge | Access | Threats | Media 2.0 | Market

Search:

Nimda still a global threat

By [Robert Lemos](#)

Staff Writer

Published: September 24, 2001, 12:30 PM PDT

[TalkBack](#) [E-mail](#) [Print](#)

The multifaceted Nimda worm continued to spread over the weekend, hitting North America on average five times harder than any other region.

Antivirus company Trend Micro's [World Virus Tracking Center](#) reported that 120,000 new infections were detected worldwide in a 24-hour period ending noon Monday PDT, bringing the total number of copies of the Nimda worm found by Trend Micro to 1.3 million.

"With most of our corporate customers, we are really in cleanup mode," said [Trend Micro](#) spokeswoman Susan Orbuch, adding that the numbers indicate it's one of the largest epidemics that the antivirus firm had ever seen. "It's hard to explain the spike. Many corporations could be directing their end users to our (online) House

Inside the Windows Security Push

The Microsoft Windows development team spent two months in 2002 analyzing product design, code, and documentation to fix security issues. The results of this security push include a new process and several lessons learned for future projects.

During February and March 2002, all feature development on Windows products at Microsoft stopped so that the complete Windows development team could analyze the product design, code, test plans, and documentation for security issues. Our team at Microsoft named the process the Windows Security Push. The security push process was derived from the .NET Framework Security Push project conducted in December 2001.

Most Windows feature teams finished the Windows Security Push in March, and since then, the company has performed many other pushes across its product line including SQL Server, Office, Exchange, and others. This article outlines the Windows Security Push process in detail as well as the rationale behind it and some of the lessons learned from it.

The rationale for the push

The driving force behind the Windows Security Push was Bill Gates' "Trustworthy Computing" memo of January 2002 (see www.microsoft.com/presspass/exec/craig/10-02trustworthywp.asp for a copy). In that memo, Gates outlined the need for more secure software

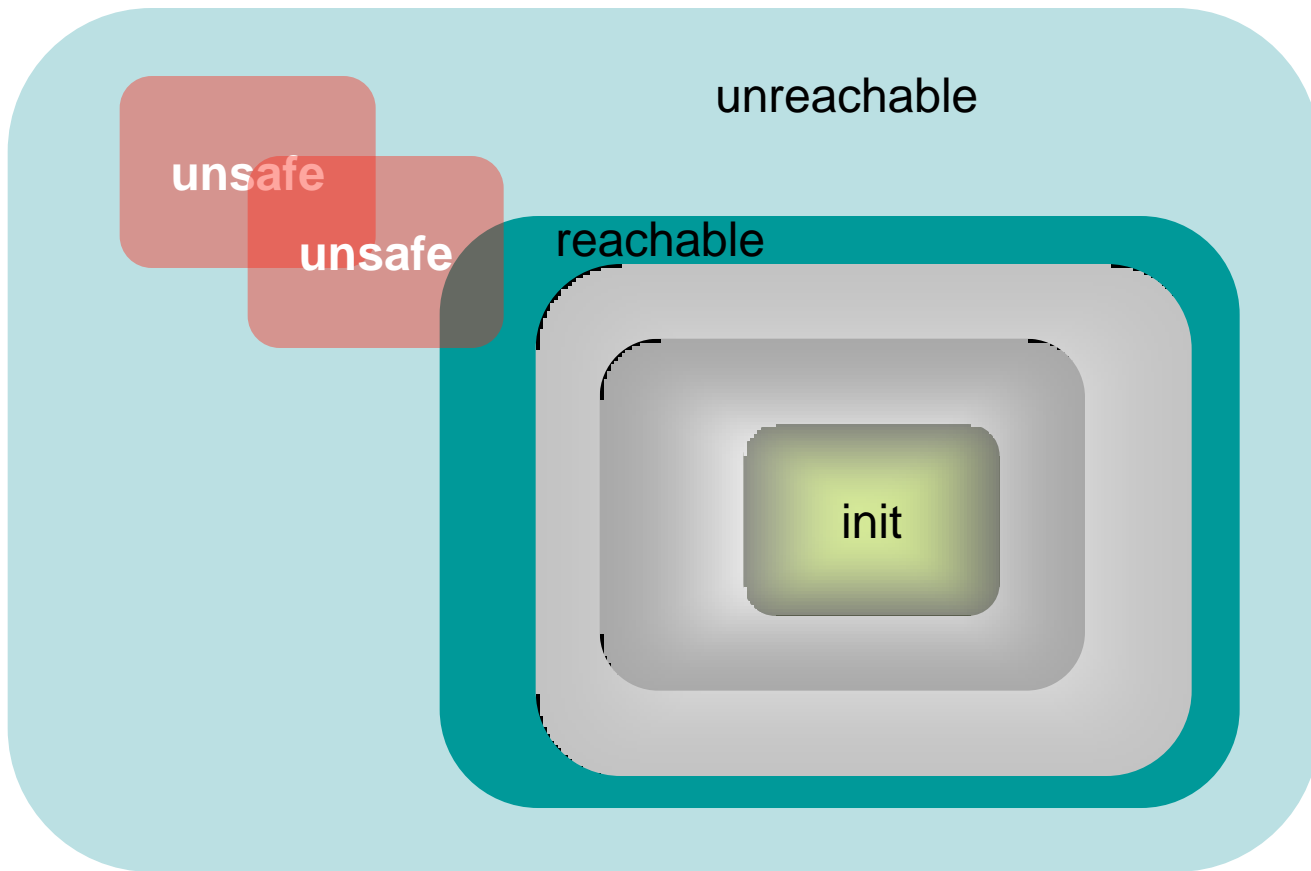
and felt that one big push at the end (in which all team members spent time focusing on nothing but security) would be worthwhile, even if it meant slipping the schedule.

And it was worthwhile: the team made a few important changes to the .NET Framework including placing the State Service (an optional component used to maintain HTTP state) and the ASP.NET worker process (used to create dynamic content on a Web server) in lower privileged accounts, rather than SYSTEM. Running code as SYSTEM gives administrators minimum hassle, but such a configuration ignores the principle of "least privilege." If hackers were to compromise the server, for example, they could have used the ASP.NET process to run attack code with SYSTEM privileges. Thus, changing the default configuration so that ASP.NET runs with minimal privileges was a good idea.

The team also let administrators opt in support for downloading and executing Internet-based code rather than executing such partially trusted code with no interaction. Many threats come from malicious Internet-borne code, so it made perfect sense to provide that functionality as an option, not as a default.



MICHAEL
HOWARD
AND STEVE
LIPNER
Microsoft



States

Reading Assignment (for next class)

- “Righting Software” paper

http://research.microsoft.com/~larus/Papers/IEEE_SW_Righting_Software.pdf

- Chapter 1 (only written chapter 😊) of Ball & Rajamani’s book
- “Automating software testing using static analysis” paper

Reading assignment (for class after next)

- Read “Findbugs” paper

<http://portal.acm.org/citation.cfm?doid=1108792.1108798>

- Read “Java Bytecode Verification” paper

<http://Gallium.inria.fr/~xleroy/publi/survey-bytecode-verification.ps.gz>