Ερευνητικά Θέματα Ανάπτυξης Λογισμικού

Σύνοψη Μαθήματος (1)

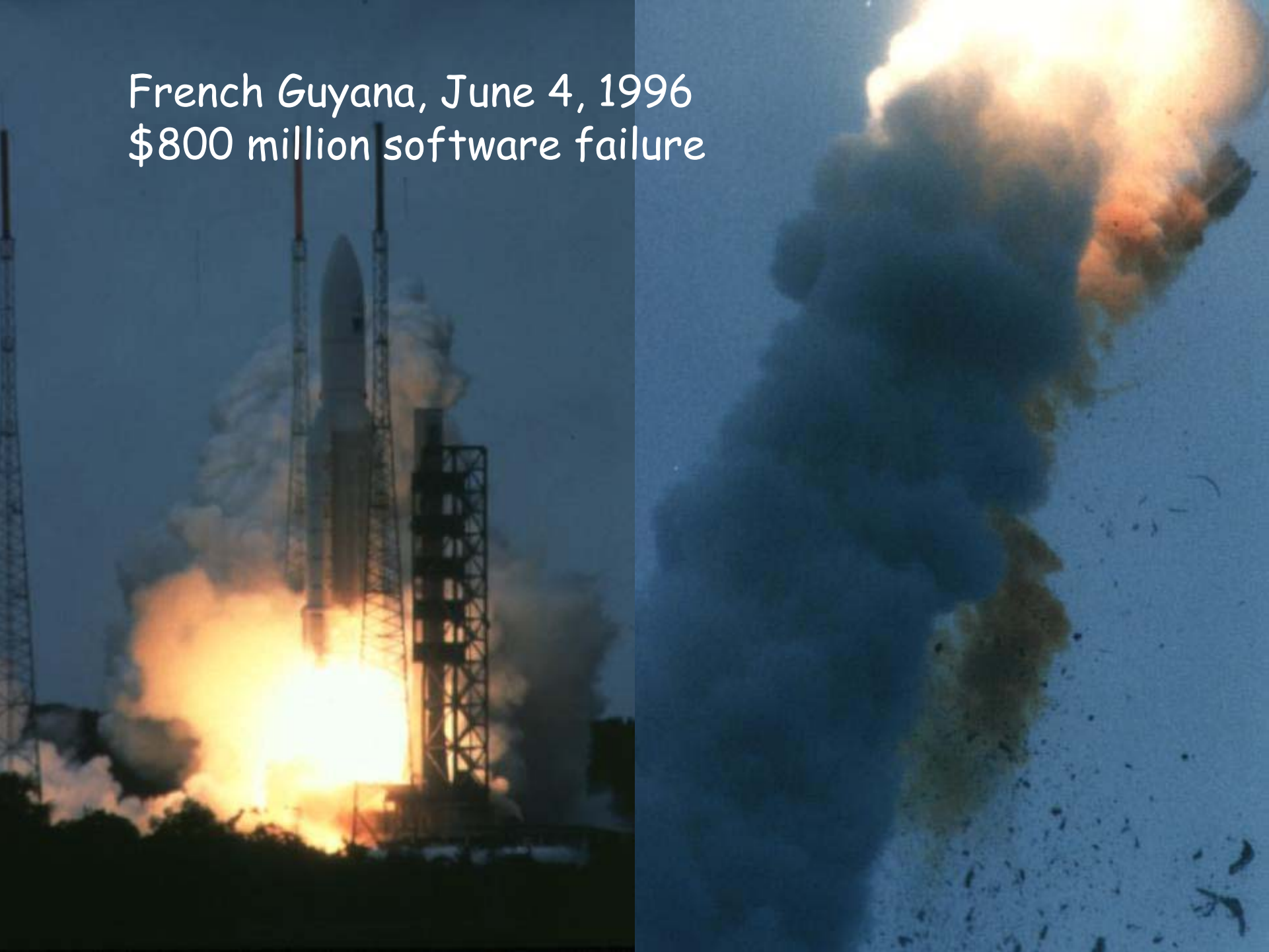
1. **Introduction & Course Overview**
   - How software is built & software defects
   - Introduction to Dataflow Analysis
2. **Introduction to Software Testing**
   - Testing practice
   - Coverage
   - Mutation testing
3. **Random Testing**
   - Property-based testing
   - Effective input generation
4. **Fuzz Testing**
   - Black/white/grey box fuzzing
   - Mutation fuzzing
   - Hybrid fuzzing

Σύνοψη Μαθήματος (2)

5. **Concurrency Testing**
   - Data race detection
6. **Debugging**
   - Debuggers
   - Debugging without debuggers — Delta debugging
   - Simplifying failure inducing input
7. **Runtime Monitoring**
   - Detecting data races — Eraser
   - Code sanitizers — ASan, TSan
8. **Static Analysis**
   - Basic principles
   - Dataflow analysis
   - Pointer analysis

Σύνοψη Μαθήματος (3)

9. **Static Bug Detection**
   - Bug Patterns — FindBugs, Coverity
   - Scalable analyses — SLAM, Infer
10. **Dynamic Symbolic Execution**
11. **Extended Static Checking**
12. **Model Checking**


French Guyana, June 4, 1996
$800 million software failure

Mars, July 4, 1997
Lost contact due to real-time priority inversion bug

Mars Climate Orbiter

- The 125 million dollar Mars Climate Orbiter is assumed lost by officials at NASA. The failure responsible for loss of the orbiter is attributed to a failure of NASA's system engineer process. The process did not specify the system of measurement to be used on the project. As a result, one of the development teams used Imperial measurement while the other used the metric system of measurement. When parameters from one module were passed to another during orbit navigation correct, no conversion was performed, resulting in the loss of the craft.

$4 billion development effort
> 50% system integration & validation cost


400 horses
100 microprocessors
M NH 2519

```
*** STOP: 0x00000019 (0x00000000,0xC00E0FF0,0xFFFFEFD4,0xC0000000)
BAD_POOL_HEADER

CPUID:GenuineIntel 5.2.c irql:1f  SYSVER 0xf0000565

Dll Base DateStmp - Name                 Dll Base DateStmp - Name
80100000 3202c07e - ntoskrnl.exe         80010000 31ee6c52 - hal.dll
80001000 31ed06b4 - atapi.sys            80006000 31ec6c74 - SCSIPORT.SYS
802c6000 31ed06bf - aic78xx.sys          802cd000 31ed237c - Disk.sys
802d1000 31ec6c7a - CLASS2.SYS           8037c000 31eed0a7 - Ntfs.sys
fc698000 31ec6c7d - Floppy.SYS           fc6a8000 31ec6ca1 - Cdrom.SYS
fc90a000 31ec6df7 - Fs_Rec.SYS           fc9c9000 31ec6c99 - Null.SYS
fc864000 31ed868b - KSecDD.SYS           fc9ca000 31ec6c78 - Beep.SYS
fc6d8000 31ec6c90 - i8042prt.sys         fc86c000 31ec6c97 - mouclass.sys
fc874000 31ec6c94 - kbdclass.sys         fc6f0000 31f50722 - VIDEOPORT.SYS
feffa000 31ec6c62 - mga_mil.sys          fc890000 31ec6c6d - vga.sys
fc708000 31ec6ccb - Msfs.SYS             fc4b0000 31ec6cc7 - Npfs.SYS
fefbc000 31eed262 - NDIS.SYS             a0000000 31f954f7 - win32k.sys
fefa4000 31f91a51 - mga.dll              fec31000 31eedd07 - Fastfat.SYS
feb8c000 31ec6e6c - TDI.SYS              feaf0000 31ed0754 - nbf.sys
feacf000 31f130a7 - tcpip.sys            feab3000 31f50a65 - netbt.sys
fc550000 31601a30 - el59x.sys            fc560000 31f8f864 - afd.sys
fc718000 31ec6e7a - netbios.sys          fc858000 31ec6c9b - Parport.sys
fc870000 31ec6c9b - Parallel.SYS         fc954000 31ec6c9d - ParVdm.SYS
fc5b0000 31ec6cb1 - Serial.SYS           fea4c000 31f5003b - rdr.sys
fea3b000 31f7a1ba - mup.sys              fe9da000 32031abe - srv.sys

Address  dword dump   Build [1381]                              - Name
fec32d84 80143e00 80143e00 80144000 ffdff000 00070b02          - KSecDD.SYS
801471c8 80144000 80144000 ffdff000 c03000b0 00000001          - ntoskrnl.exe
801471dc 80122000 f0003fe0 f030eee0 e133c4b4 e133cd40          - ntoskrnl.exe
80147304 803023f0 0000023c 00000034 00000000 00000000          - ntoskrnl.exe

Restart and set the recovery options in the system control panel
or the /CRASHDEBUG system start option.
```

The Blue Screen

More Blue Screen Embarrassments

CNN.com

PRINT THIS

Powered by clickability

Spread of buggy software raises new questions

NEW YORK (AP) --When his dishwasher acts up and won't stop beeping, Jeff Seigle turns it off and then on, just as he does when his computer crashes. Same with the exercise machines at his gym and his CD player.

"Now I think of resetting appliances, not just computers," says Seigle, a software developer in Vienna, Virginia.

Malfunctions caused by bizarre and frustrating glitches are becoming harder and harder to escape now that software controls everything from stoves to cell phones, trains, cars and power plants.

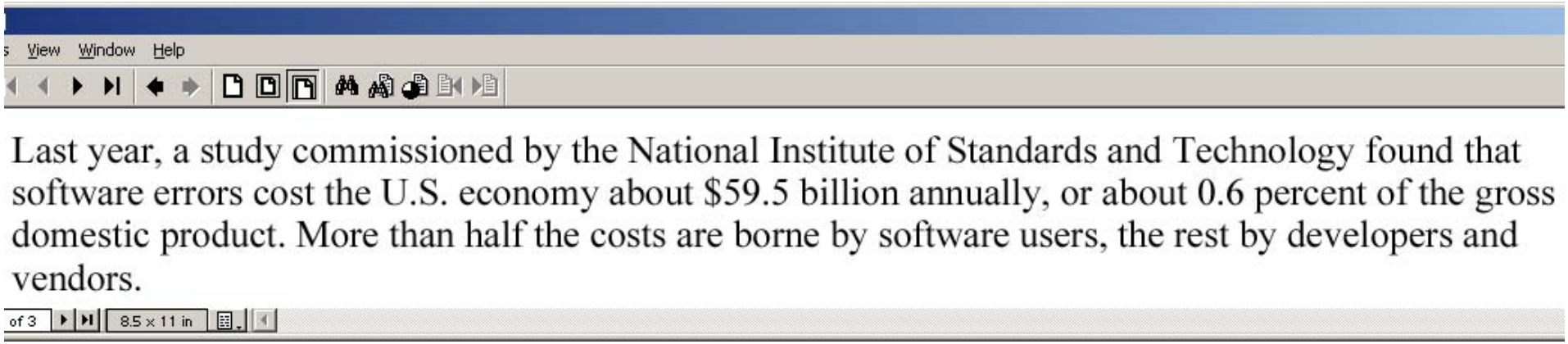
--A poorly programmed ground-based altitude warning system was partly responsible for the 1997 Korean Air crash in Guam that killed 228 people.

--Faulty software in anti-lock brakes forced the recall of 39,000 trucks and tractors and 6,000 school buses in 2000.

--The $165 million Mars Polar Lander probe was destroyed in its final descent to the planet in 1999, probably because its software shut the engines off 100 feet above the surface.

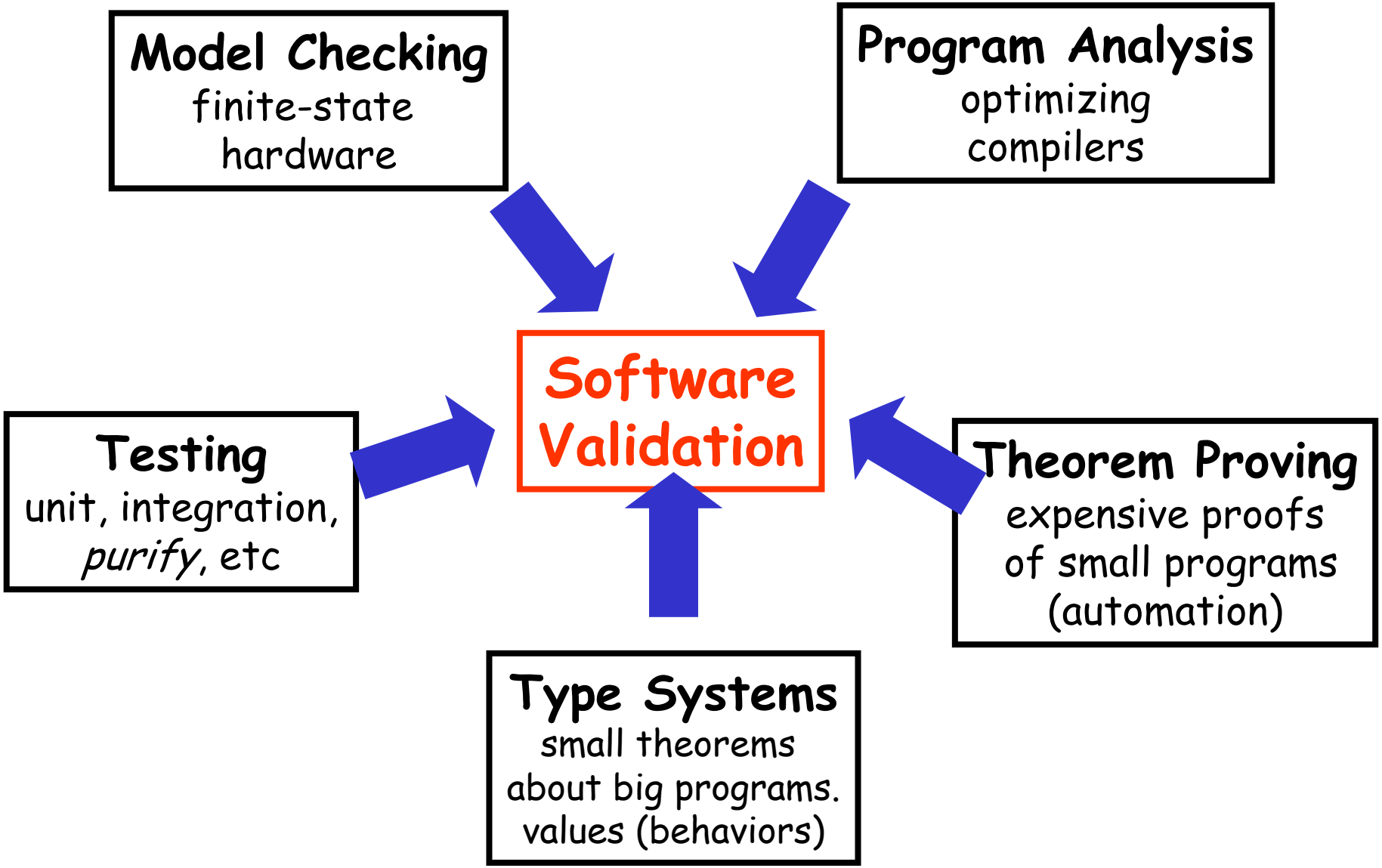
Economic Impact of Software Bugs

- **NIST study**
  - On CNN.com - April 27, 2003



http://www.nist.gov/director/prog-ofc/report02-3.pdf

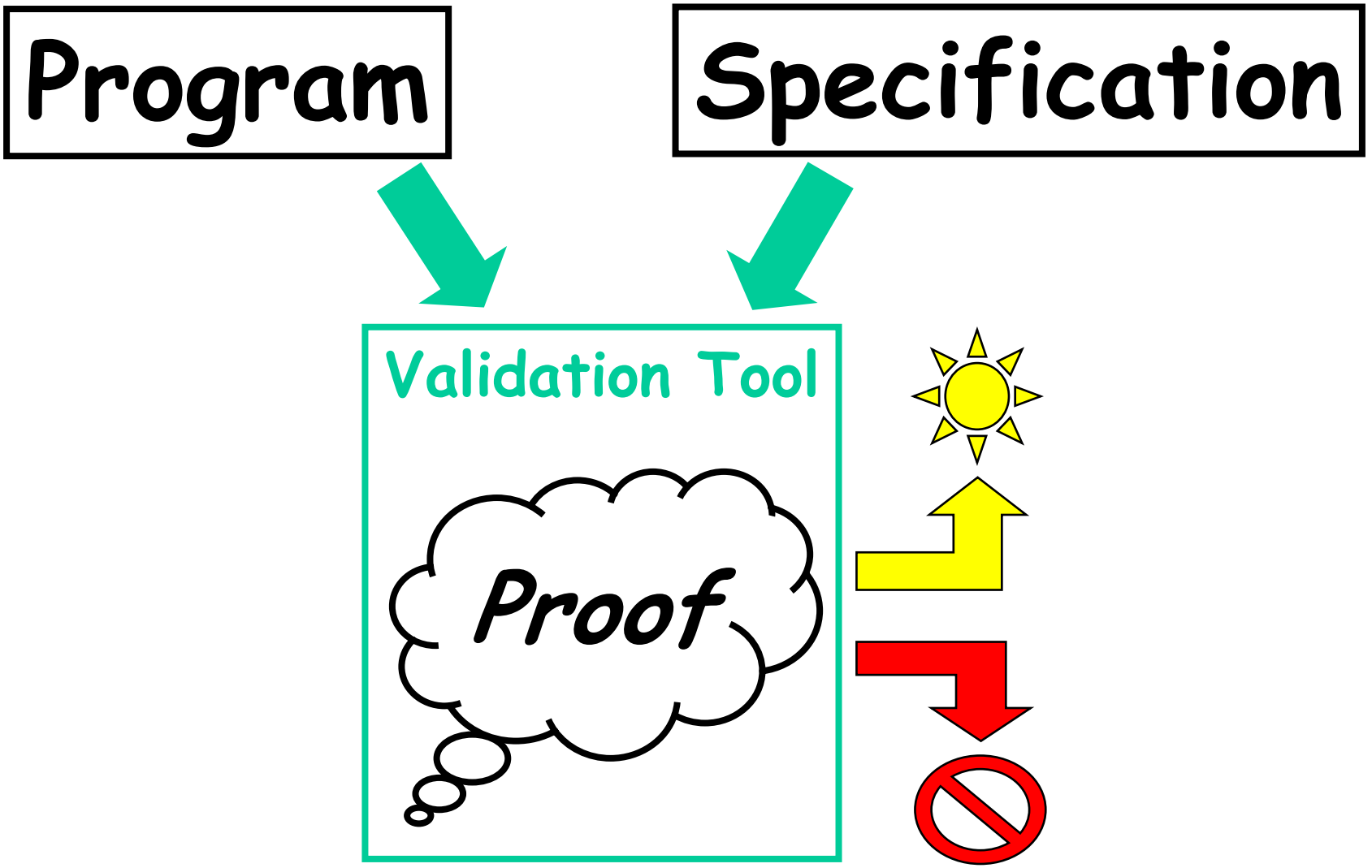
Course Overview



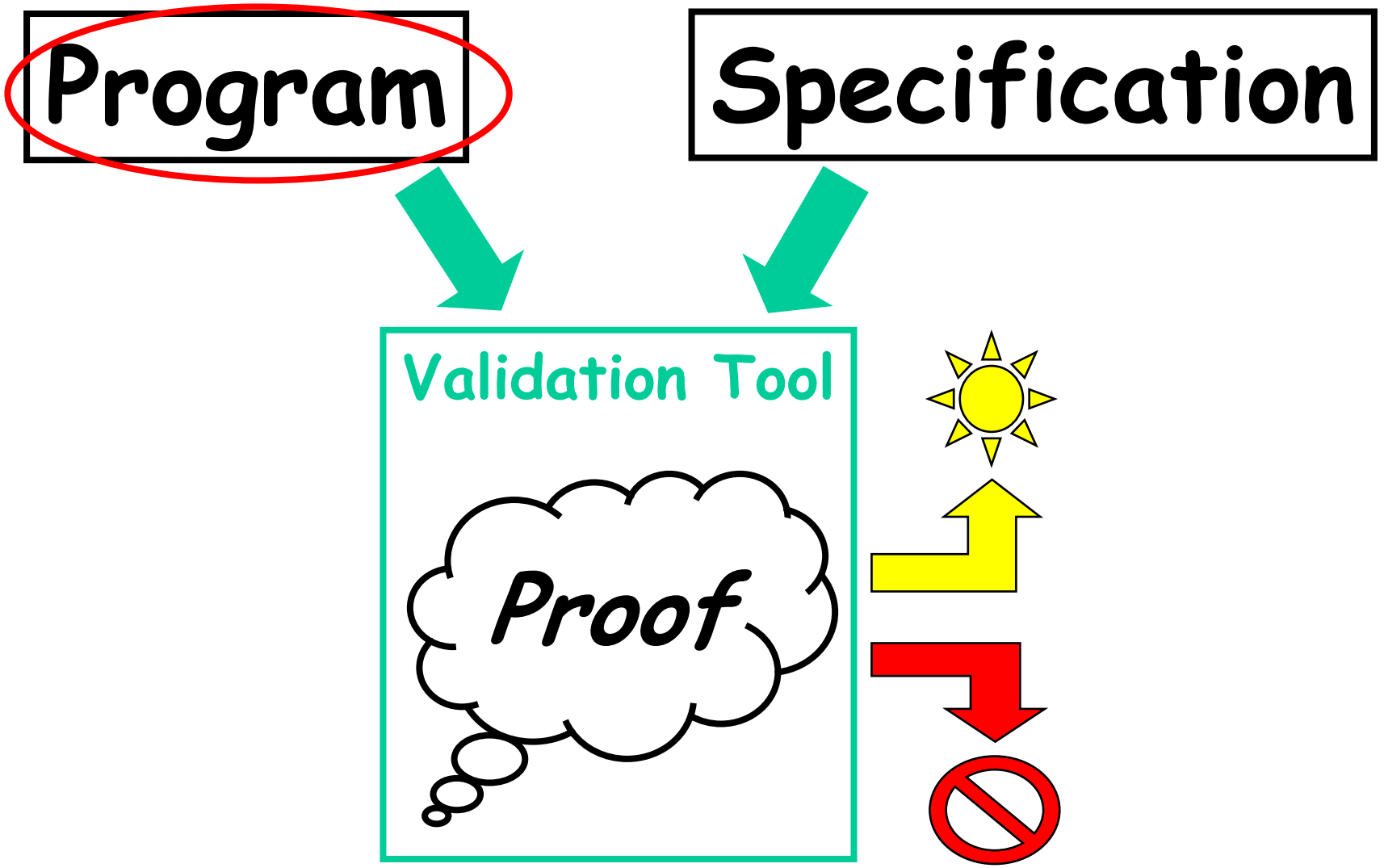
Analysis vs. Validation

- **Analysis**
  - figure out what properties a program satisfies
- **Validation**
  - does a program satisfy a particular property or "specification"

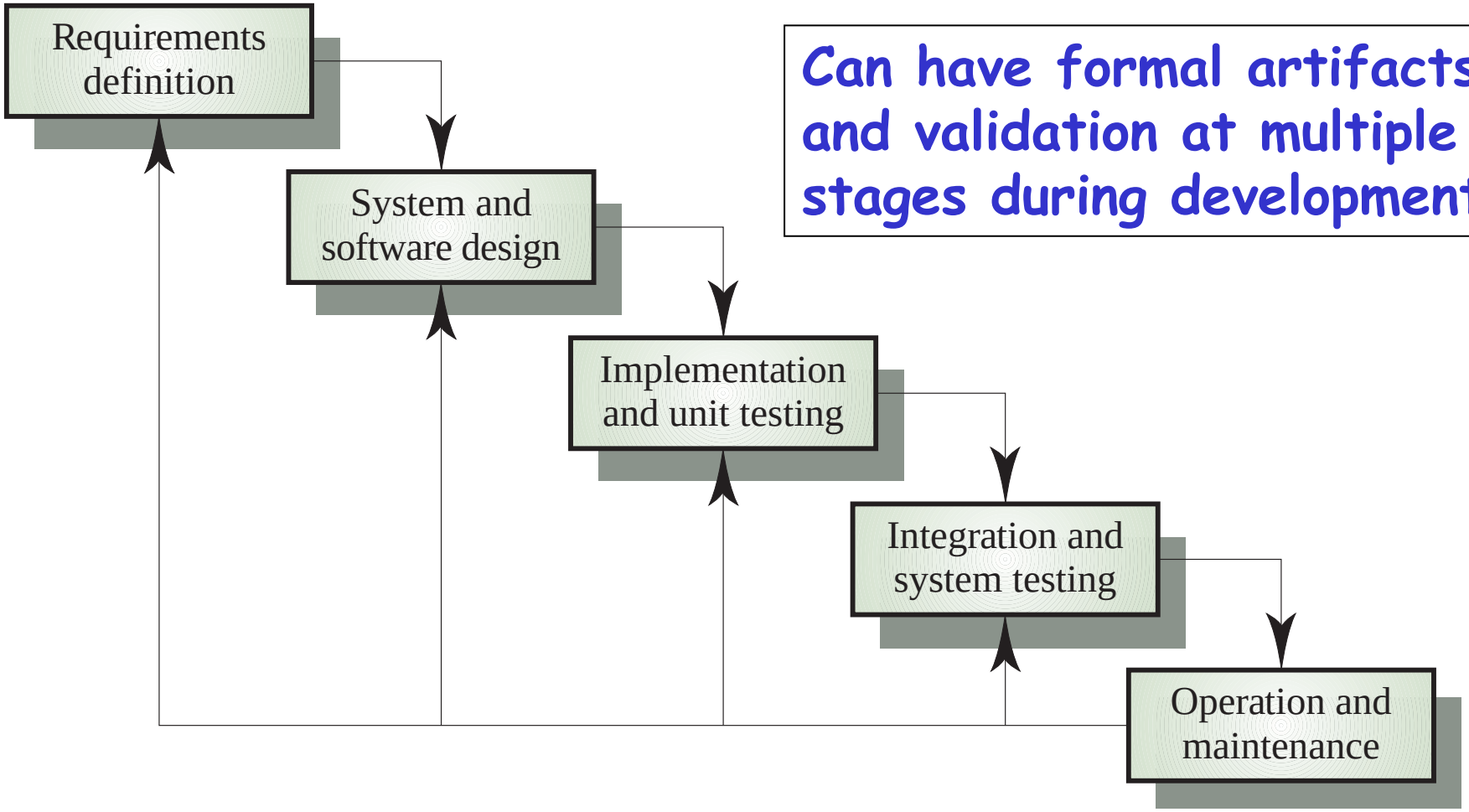
Software Validation



Software Validation

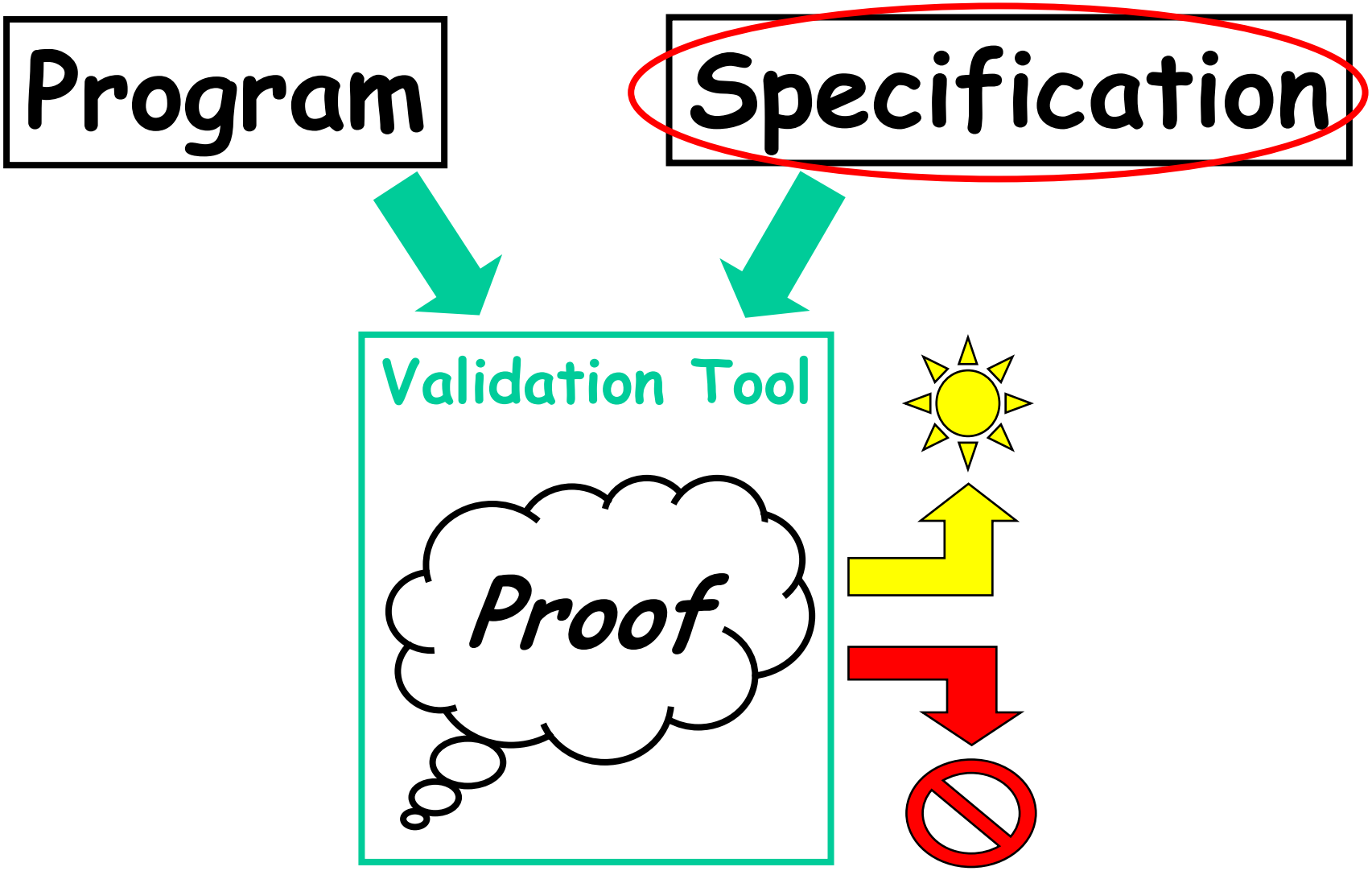


Waterfall Model



Can have formal artifacts and validation at multiple stages during development

Software Validation



Specifications

- **Safety**
  - something "bad" will never happen
  - finds most bugs
- **Liveness**
  - something "good" will eventually happen
  - (we don't know when)

For Sequential Programs

- **Safety**
  - the program will never produce a wrong result
    - "partial correctness"
- **Liveness**
  - the program will produce a result
    - "termination"

Example Specifications

Basic specifications

- no null dereference
- no bounds errors
- no segmentation faults

Resource management

- no memory leaks

Concurrency

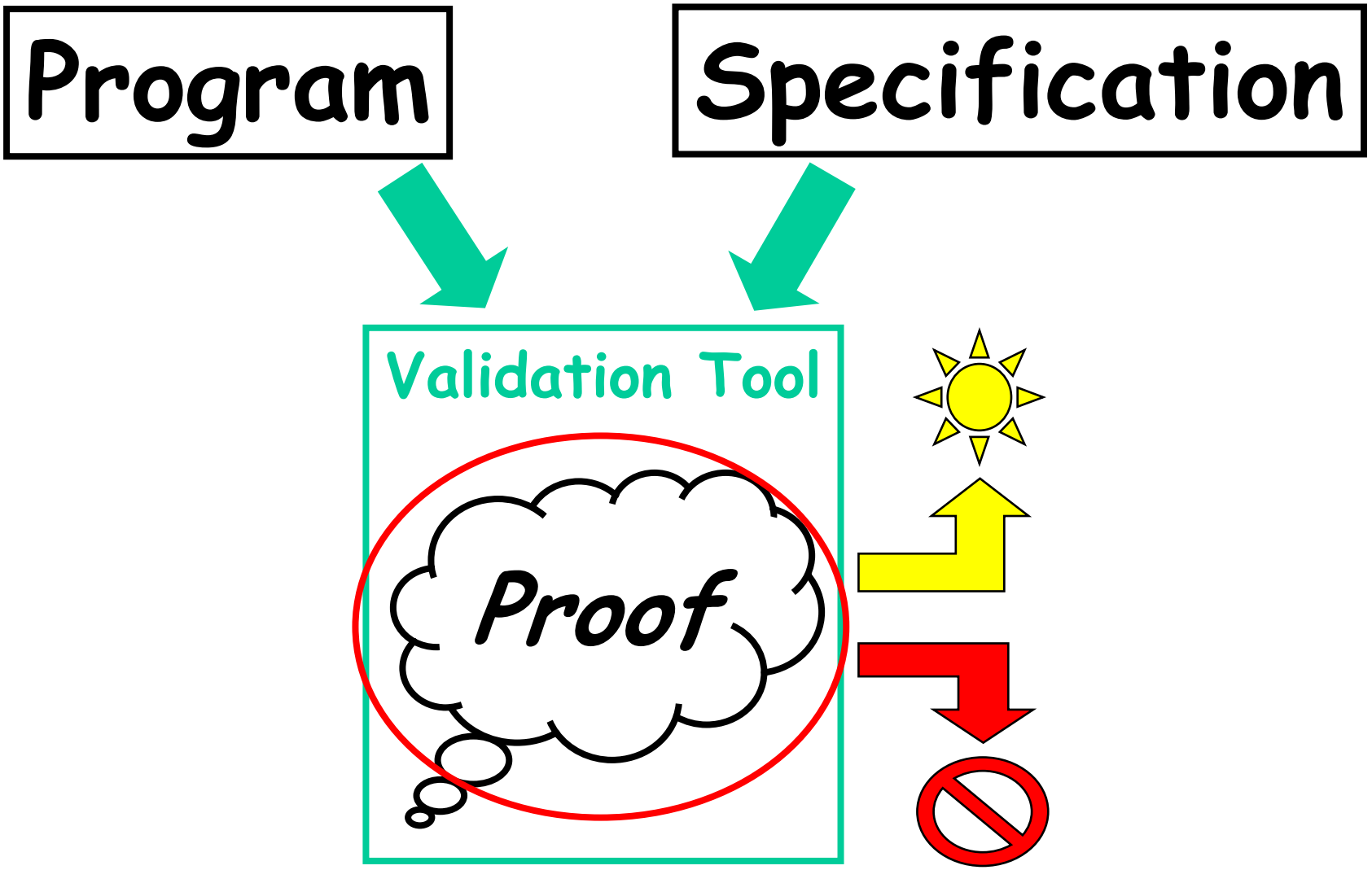
- no race conditions
- no deadlock

Example Specifications

- **Data invariants**
  - shape properties (L is an acyclic list)
- **Security**
  - integrity, confidentiality

- **API Usage rules**
  - files
  - UNIX sockets

Software Validation



Undecidability

- **Does a program P satisfy a specification S ?**
  - everything interesting about infinite-state programs is undecidable
    - consequence of halting problem
  - no sound, complete, terminating algorithm

Avoiding Undecidability

- **Finite state systems**
  - model checking
  - automatic
  - effective for small systems
- **Miss errors (unsound)**
  - testing, bounded model checking
  - test coverage problem
- **False alarms (incomplete)**
  - program analysis, type systems
  - only consider certain proofs

Program Analysis Fundamentals

Uses of Program Analysis

- **Historically: Optimize (some aspect of) a program's execution by the compiler**

- **More recently:**
  - Finding software bugs
  - Detect security vulnerabilities
  - Detect policy/privacy violations
  - Detect errors in "smart contracts"
  - …

Culture

- **Emphasis on low-complexity techniques**
  - Because of emphasis on usage in tools
  - High-complexity techniques also studied, but often don't survive
- **Emphasis on complete automation**
- **Driven by language features**
  - Particular languages and features give rise to their own sub-disciplines