Software Design Tools

Derek Rayside
CS446 / ECE452
June 25, 2010

Design Analysis

Can we know properties of the design before we build it?

e.g., Civil Engineers know how much a bridge will hold before they build it.

Math.
Discrete Math:
Logic;
Graph Theory.

Design Prototyping

"Plan to throw one away; you will anyhow" -- Fred Brooks

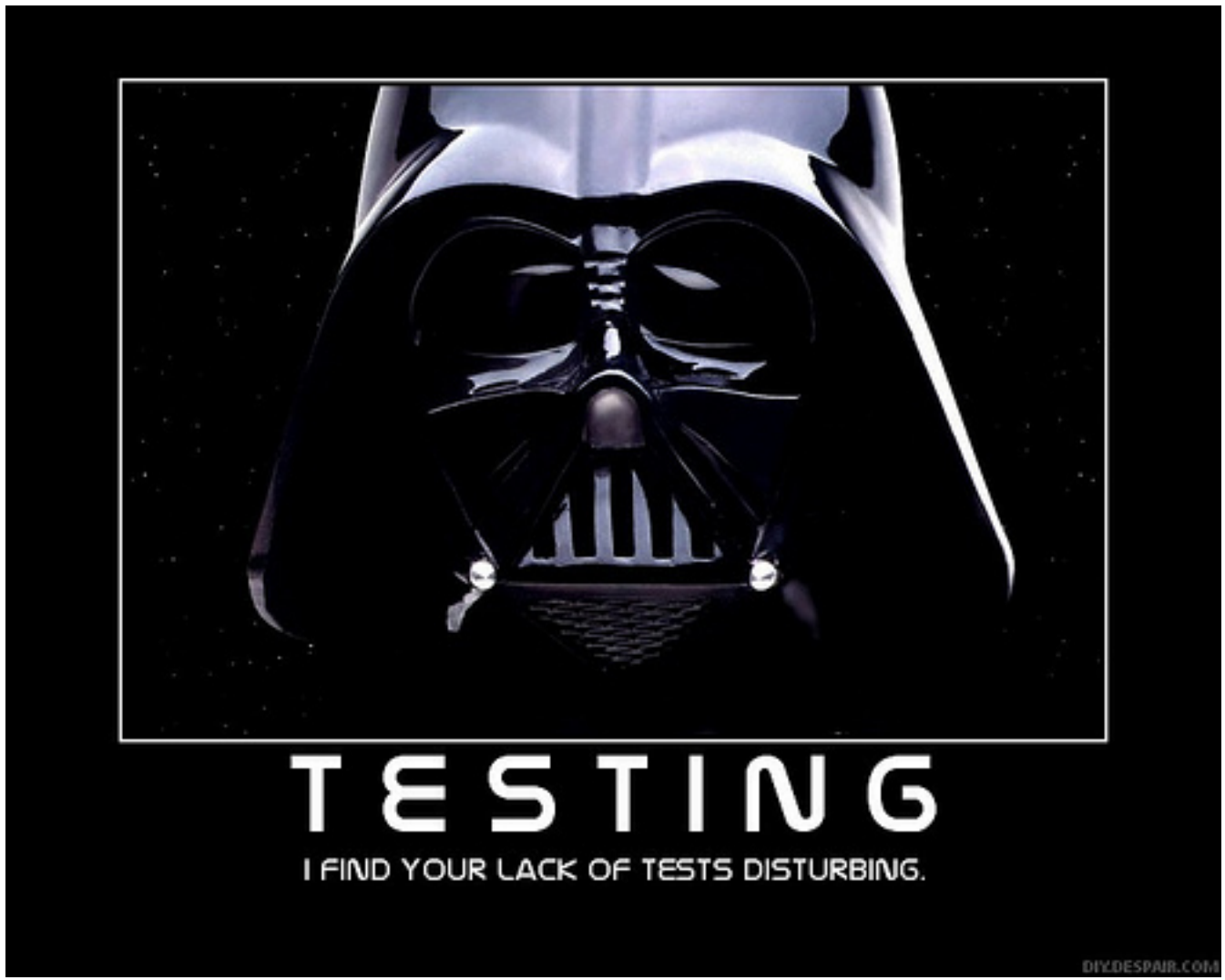
Plan to throw away an inexpensive prototype, rather than a complete system.

e.g., UI design uses paper prototypes.

c.f. "evolutionary prototype" = alpha

Design Conformance Testing

Does the code match the design?



Some Software Design Tools

Analysis:

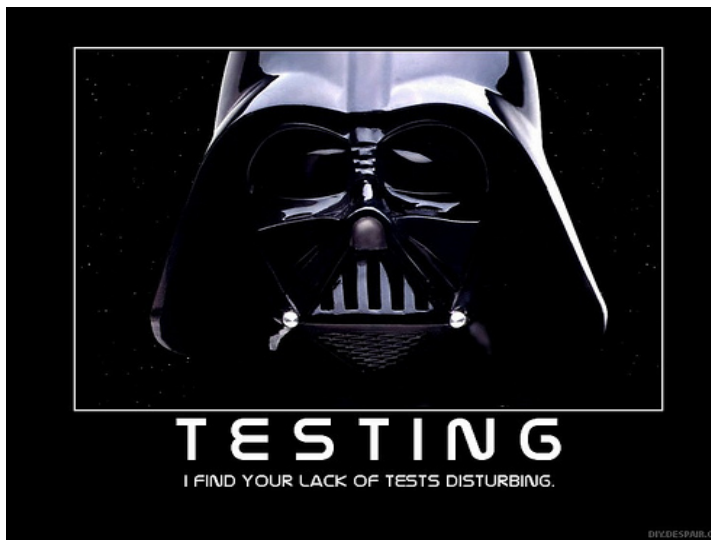
- ArchStudio
- Alloy
- Spin
- Java PathFinder
- SMV/NuSMV

Prototyping:

- Alloy
- Java PathFinder
- Haskell / OCaml
- TXL
- JESS/CLIPS
- Prolog / Datalog
- Crocopat / Grok

Testing:

- LSEdit
- Korat
- Randoop
- ESC/Java
- JForge
- Spec#
- Microsoft SDV

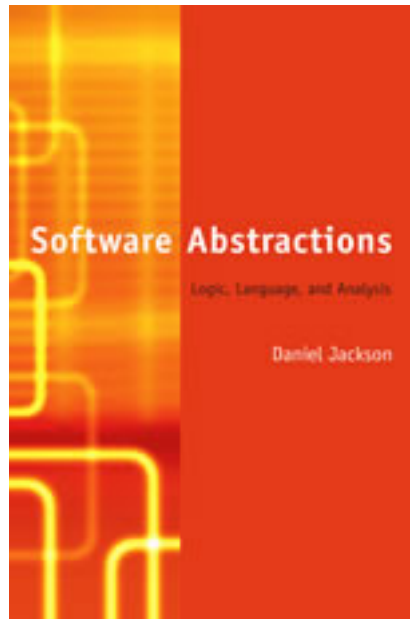


Design Analysis Tools

Alloy

Description:

- first-order logic
  - + relations
  - + transitive closure
  - for reasoning about structures
- ASCII syntax
- visualizer
- open source
- http://alloy.mit.edu
- [D. Jackson book]



Useful for:

- complex structures
- protocols
- logical puzzles

Terminology

Programmers

- table
- row / record

- single-column table
- two-column table

Mathematicians

- relation
- tuple

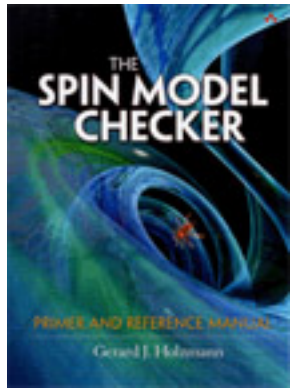
- unary relation
- binary relation

What's a three-letter word for "unary relation"?
Hint: it has the longest definition of any word in the Oxford English Dictionary.

Spin

Description:

- Linear Temporal Logic
  - for reasoning about sequences of events
  - temporal quantifiers: always, eventually, etc
- program in Promela
- property in LTL
- Lucent + NASA
- open source
- http://spinroot.com
- many text books:
  - Holzmann
  - Ben-Ari
  - etc



Useful for:

- concurrent systems
- protocols
- finite state machines
- simple structures

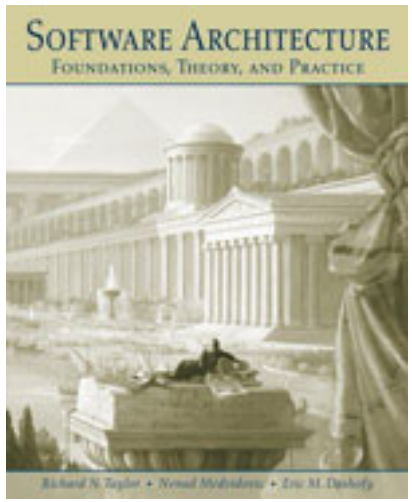
- flood control system
- phone switch [Lucent]
- SIP [Pamela Zave]
- Deep Space 1 [NASA]
- Cassini [NASA]
- Mars Rovers [NASA]
- Deep Impact [NASA]

ArchStudio

ArchStudio 4
Software and Systems Architecture Development Environment

Description:

- when boxes & arrows grow up
- Archlight analysis tool
- highly extensible
- UC Irvine
- Eclipse plugin
- open source
- [Taylor et al text book]



Useful for:

- a wide variety of software systems
- software product lines
  - multiple variants (e.g., Tektronix oscilliscope case study from Garlan & Shaw)

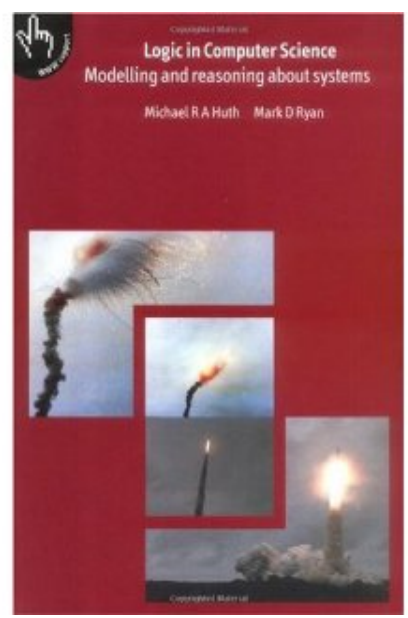
SMV/NuSMV

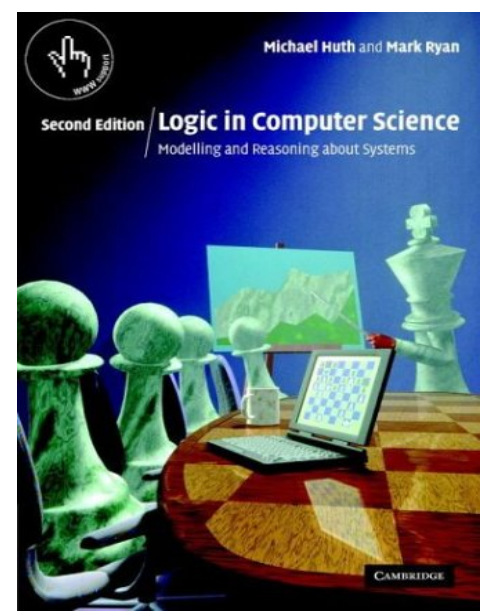
Description:

- Computation Tree Logic
  - for reasoning about sequences of events
  - like LTL but different
- http://nusmv.irst.itc.it/
- open source
- CMU + Italy * 3
- [Huth & Ryan textbook]

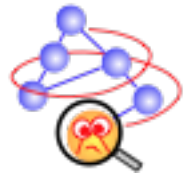
Useful for:

- concurrent systems
- Shuttle Digital Autopilot engines out (3E/O)
- TCAS II air traffic control





Java Path Finder (JPF)



JPF

Description:

- a special JVM
  - program in Java
  - property in Java
- NASA
- open source
- http://babelfish.arc.nasa.gov/trac/jpf
- SE464 can probably support

Useful for:

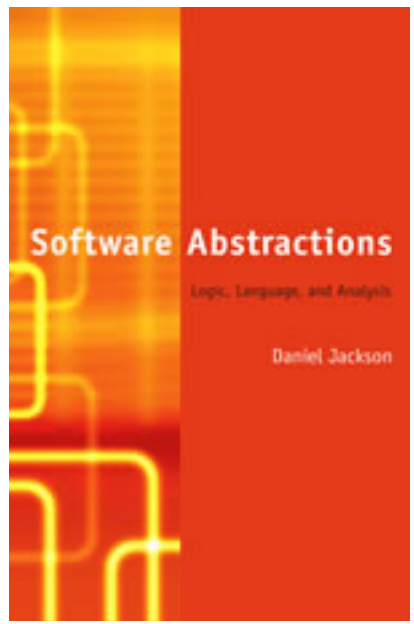
- concurrent systems
- non-deterministic systems
- Java prototypes
  - no I/O
- K9 Rover [NASA]
- Livingstone 2 on EO-1

Design Prototyping Tools

Alloy

Description:

- first-order logic
  - + relations
  - + transitive closure
  - for reasoning about structures
- ASCII syntax
- visualizer
- open source
- http://alloy.mit.edu
- [D. Jackson book]

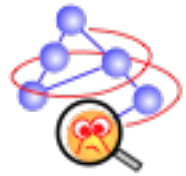


Useful for:

- complex structures
- protocols
- logical puzzles

- NP-complete computations (e.g., scheduling)

Java Path Finder (JPF)



JPF

Description:

- a special JVM
  - program in Java
  - property in Java
- NASA
- open source
- http://babelfish.arc.nasa.gov/trac/jpf
- SE464 can probably support

Useful for:

- concurrent systems
- non-deterministic systems
- Java prototypes
  - no I/O
- K9 Rover [NASA]
- Livingstone 2 on EO-1

TXL

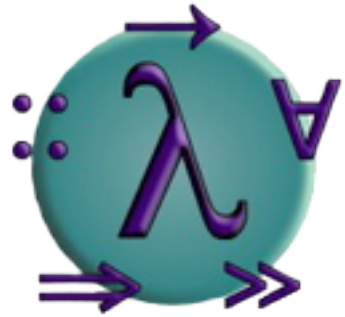
Description:

- rule-based tree transformations
- Queens
- http://txl.ca/

Useful for:

- prototyping
  - data transformations
  - programming languages
-

Haskell / OCaml



Description:

- functional programming languages
- with a strong type system
- http://haskell.org
- http://caml.inria.fr/
-

Useful for:

- prototyping symbolic computations
  - avoid I/O
- seL4

JESS, CLIPS, Prolog

Description:

- rule engines
- rules match facts
  - blackboard style
- NASA [CLIPS]
- Sandia Nat. Labs [JESS]
- France [Prolog]
-
- http://www.jessrules.com/
- http://clipsrules.sf.net/
-

Useful for:

- prototyping
- diagnostic systems
  - eg, medicine
- business rules
  - eg, JBoss Drools
- AI

Crocopat / Grok

Description:

- a relational calculator
- like a relational query engine (eg, SQL) for text files
-
- http://www.sosy-lab.org/~dbeyer/CrocoPat/
- http://swag.uwaterloo.ca

Useful for:

- prototyping manipulations of large datasets
- prototyping complex SQL queries

Design Testing Tools

LSEdit

- architectural conformance
  - concrete architecture vs conceptual architecture
  - c.f.: other tools on these slides are very low-level

- you've already seen this in action:
  - Bowman et al paper on the Linux Kernel
  - Hassan et al paper on web servers (Apache, etc)
  - guest lecture by Ian Davis

Spec#, JForge, ESC/Java

Description:

- check code against specs
- first-order logic
- static analysis
- Spec# [Microsoft]
- JForge [MIT]
- ESC/Java [DEC etc]

Useful for:

- checking code vs specs
- heap-manipulating computations
  - not numerical
  - not I/O
  - single-threaded

Microsoft Static Driver Verifier (SDV)

Testing device drivers is hard:

- asynchronous and massively re-entrant
- make complex use of kernel APIs
- evolve over different HW and OS versions
- it's hard to directly observe the driver interacting with the OS in a bad way
- subtle errors may only occur in exceptional situations in the field, and be hard to re-create in the lab

Notes:

- static analysis (tests code without running it)
- passing SDV is required in order to ship with Windows

Korat

- automatically generates test inputs
- from invariants written in repOk() method
  - invariants define legal states of an object
- Korat generates all (non-isomorphic) legal states
- use the generated test-inputs for your tests
- dynamic analysis
- http://korat.sourceforge.net/

Randoop

- creates random sequences of method calls
- looks for "something bad" to happen
  - object contract violations
    - x.equals(x)
    - x.equals(y) <=> y.equals(x)
    - x.equals(y) => x.hashCode()==y.hashCode()
  - programmer-defined badness
- finds bugs fast!
- dynamic analysis
- http://code.google.com/p/randoop/

Some Software Design Tools

Analysis:

- ArchStudio
- Alloy
- Spin
- Java PathFinder
- SMV/NuSMV

Prototyping:

- Alloy
- Java PathFinder
- Haskell / OCaml
- TXL
- JESS/CLIPS
- Prolog / Datalog
- Crocopat / Grok

Testing:

- LSEdit
- Korat
- Randoop
- ESC/Java
- JForge
- Spec#
- Microsoft SDV