concepts & software

Daniel Jackson · Google, Cambridge · May 8, 2012
#1

the good & the bad
how to add a signature in acrobat
-- open document in acrobat
-- Tools->Advanced Editing->Touchup Object Tool
-- right click at desired point | Place Image...
then select jpg

how to add date
-- Tools->Typewriter
## Hypothesis

<table>
<thead>
<tr>
<th>weak concepts</th>
<th>strong concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>hard to use</td>
<td>intuitive, predictable</td>
</tr>
<tr>
<td>a mess to maintain</td>
<td>decoupling &amp; localization</td>
</tr>
<tr>
<td>unreliable &amp; buggy</td>
<td>more dependable</td>
</tr>
</tbody>
</table>
#2
how to do it?
what we’re already doing

- thinking & sketching
- simulating features
- normal design practice
- copying good ideas
- evaluating products
- user feedback
- discarding failed designs
- “refactoring”

To design something really well, you have to get it. You have to really grok what it’s all about. It takes a passionate commitment to really thoroughly understand something, chew it up, not just quickly swallow it. Most people don’t take the time to do that. --Steve Jobs
what we’re not doing

being explicit

focusing: what are the concepts?
relating: how are they related to each other?
analyzing: what properties do they have?
#3
an approach
**semantic concepts**

- **atom**
  - indivisible
  - immutable
  - uninterpreted

- **relation**
  - collection of atom tuples

- **set**
  - collection of atoms
  (ie, a unary relation)
graphical syntax

A ⊆ B

A1 ⊆ B, A2 ⊆ B
A1 ∩ A2 = Ø

R maps m A’s to each B
R maps each A to n B’s

+ one or more
* zero or more
! exactly one
? at most one
omitted = *

A

m

R

n

B

A

A1

A2

B

Person

Man

Woman

father

mother

wife

husband

father

mother

wife

husband
the alloy analyzer: a model finder

API

elaborate

typecheck

visualize

alloy front end

alloy command

visual output

alloy formula

bounds

alloy instance

kodkod engine

translate formula

mapping

translate instance

API

boolean formula

SAT solver

boolean instance
i'm my own grandpa
#4

some generic concepts
There is no problem in computer science that cannot be solved by introducing another level of indirection. --David Wheeler

rough edges
› Indesign: can’t tell whether you assigned color or swatch
› CSS: formatting rules aren’t independent
“pseudo style”
"composite"

rough edges
› Lightroom: “collection sets”
› IMAP vs Apple Mail: folder holding message and folder?
› Google docs: collections a bit scary?
As always, the battle is set between the haves, and the rest. The haves have just lost out, and the movement back towards the greater good has begun. Let’s hope the American public sees the sense in what the French and Greeks have done yesterday, by reclaiming their governments to represent the people for a change. The stock markets will likely be unhappy.

May 7, 2012 at 3:23 p.m. · RECOMMEND 8

Ok, I’ve rolled my own theme, I’ve made a custom jQuery UI pack (progress bar, date picker, slider) and installed it all. Seems to (mostly) work except for two things:

1. When my page first loads the datepicker div is visible; and
2. The text "Next" and "Prev" are visible in large font underneath my icons. None of the examples seem to have this problem.

Now (1) I'm currently solving by:

    .ui-datepicker-div { display: none; }

in another CSS file but again none of the demos seem to need this.

What am I missing?

javascript query css jquery-ui

Was this post useful to you? Yes No

rough edges

› your suggestions?
#5

three conceptual models
rough edges
› special role of Normal style, etc
› hidden memory of inherit vs replace with same

the origins of paragraph styles
Bravo-X at Xerox PARC: Tim Mott, Larry Tesler, Charles Simonyi; first commercialized in Word, now ubiquitous (Pages, Indesign, Quark,...)
javascript objects

rough edges
› add slots to all objects? is 23 an object?
origins & referrers
a strategy for XSS and CSRF
browser tracks “origin” of each request with HTTP request, includes origin as “referrer” if referrer is not self, server rejects it

critical property
$s = \text{origin}(r)$ iff $s$ really is cause of $r$
modeling origins

[Diagram]

- **Requested**: HTTPEvent
- **Response**: Embeds Request, Response
- **Redirect**: from, to, origin
- **Client** causes embeds
- **Server**

Arrows indicate relationships such as `causes`, `embeds`, and `from, to, origin`
define basic concepts

abstract sig HTTPEvent {from, to, origin: EndPoint}
abstract sig EndPoint { causes: set HTTPEvent }
  { causes = {e: HTTPEvent - Embedded | e.from = this} + causes.embeds }

sig Client, Server extends EndPoint {}

sig Request extends HTTPEvent { response: Response }
  { from in Client and to in Server }

sig Response extends HTTPEvent { embeds: set Embedded }
  { from in Server and to in Client }

sig Embedded extends Request {}
fact {Embedded = Response.embeds}

sig Redirect extends Response {}
fact RequestResponse {
  response in Request one -> one Response
  all r: Request | r.from = r.response.to and r.to = r.response.from
}
define origin tracking

**fact** Origin {
    // for a redirect, origin is same as request, else server
    all r: Request | r.response.origin =
        (r.response in Redirect implies r.origin else r.response.from)

    // embedded requests have the same origin as the response
    all r: Response, e: r.embeds | e.origin = r.origin

    // requests that are not embedded come from the client
    all r: Request - Embedded | r.origin = r.from
}
does the policy work?

check {
  no server: Server, attacker: Server - server {
    // no direct request to attacker
    no r: Request | r.to = attacker and r.origin in Client
    // trusted server obeys origin policy
    server.appliesSOP
    // and attacker still gets request through
    some r: attacker.causes | r.to = server
  }
}
for 6 but 1 Client, 2 Server
counterexample!
Towards a Formal Foundation of Web Security [2010]
Akhawe, Barth, Lam, Mitchell & Song

generic model of web security
HTTP, certificates, cookies, script contexts
about 2,000 lines of Alloy

<table>
<thead>
<tr>
<th>Case Study</th>
<th>Lines of new code</th>
<th>No. of clauses</th>
<th>CNF gen. time (sec)</th>
<th>CNF solve time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin Header CORS</td>
<td>25</td>
<td>977,829</td>
<td>26.45</td>
<td>19.47</td>
</tr>
<tr>
<td>Referer Validation</td>
<td>80</td>
<td>584,158</td>
<td>24.07</td>
<td>82.76</td>
</tr>
<tr>
<td>HTML5 Forms</td>
<td>35</td>
<td>974,924</td>
<td>30.75</td>
<td>9.06</td>
</tr>
<tr>
<td>WebAuth</td>
<td>35</td>
<td>976,174</td>
<td>27.67</td>
<td>73.54</td>
</tr>
</tbody>
</table>

applied to 5 case studies
in each, found vulnerabilities
2 known, 3 unknown
more examples: alloy.mit.edu

alloy: a language & tool for relational models

about alloy

Alloy is a language for describing structures and a tool for exploring them. It has been used in a wide range of applications from finding holes in security mechanisms to designing telephone switching networks.

An Alloy model is a collection of constraints that describes (implicitly) a set of structures, for example: all the possible security configurations of a web application, or all the possible topologies of a switching network. Alloy's tool, the Alloy Analyzer, is a solver that takes the constraints of a model and finds structures that satisfy them. It can be used both to explore the model by generating sample structures, and to check properties of the model by generating counterexamples. Structures are displayed graphically, and their appearance can be customized for the domain at hand.

At its core, the Alloy language is a simple but expressive logic based on the notion of relations, and was inspired by the Z specification language and Tarski's relational calculus. Alloy’s syntax is designed to make it easy to build models incrementally, and was influenced by modeling languages (such as the object models of OMT and UML). Novel features of Alloy include a rich subtype facility for factoring out common features and a uniform and powerful syntax for navigation expressions.

The Alloy Analyzer works by reduction to SAT. Version 4 was a complete rewrite that included Kodkod, a new model finding engine that optimizes the reduction, and a new front end.
#6 anti-patterns
non-uniformity
members of set have different properties or behaviors
eg: in Photoshop, base layer is different

coupling
concepts are not independent
eg: in OS X, folder view vs. network access
eg: in CSS, element position vs. wrap around

over-generalization
distinct concepts merged
eg: in mail clients, trashed messages have no deletion date
unity of purpose?

Conceptual integrity is the most important consideration in system design. It is better to have a system omit certain anomalous features and improvements, but to reflect one set of design ideas, than to have one that contains many good but independent and uncoordinated ideas.

-- Fred Brooks, 1975
thank you!
public class Sudoku {
    private int[][] grid = new int[9][9];

    @Ensures({
        "all row in {0..8} | this.grid[row][int] = {1..9}",
        "all col in {0..8} | this.grid[int][col] = {1..9}",
        "all r, c in {0, 1, 2} |
        this.grid[{r*3..r*3+2}][{c*3..c*3+2} = {1..9}"
    })
    @Modifies("this.grid[int].elems | _<2> = 0")
    public void solve() { Squander.exe(this); }

    public static void main(String[] args) {
        Sudoku s = new Sudoku();
        s.grid[0][3] = 1; ...; s.grid[8][8] = 5;
        s.solve();
        System.out.println(s);
    }
}
performance

\textit{n-queens}

\textbf{hamiltonian path, none}

\textbf{hamiltonian path, some}
Rubicon specs

**RSpec test**

```ruby
it "user included in list of users" do
  user = Factory(:user)
  get :index
  assigns[:users].should include user
end
```

**Rubicon spec**

```ruby
it "all users included in list of users" do
  User.forall do |user|
    get :index
    assigns[:users].should include(user)
  end
end
```
prototype Apache analyzer