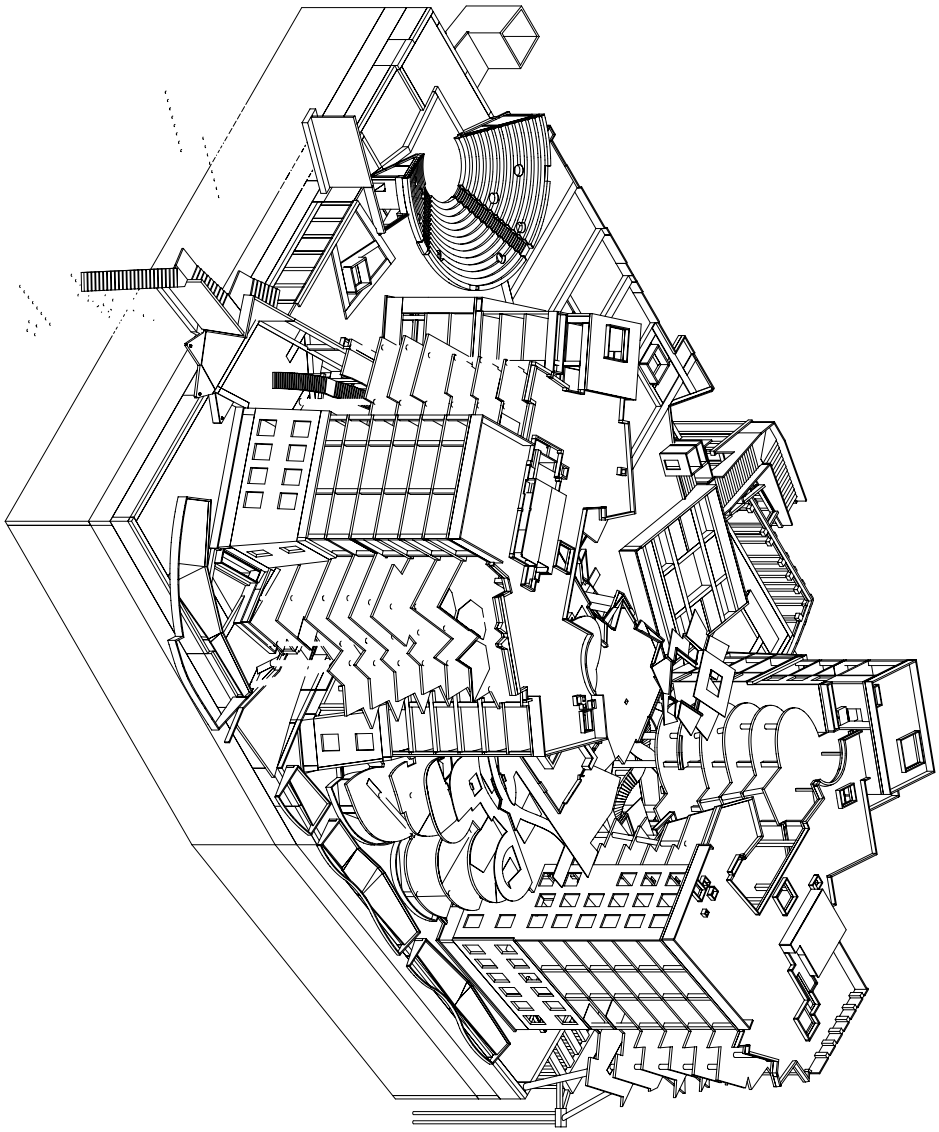
RETHINKING THE ROLE OF DESIGN IN SOFTWARE DEVELOPMENT

Daniel Jackson · Computer Science & Artificial Intelligence Lab · MIT

traditional engineers …

use models ...



	ISSUE	NO.	DATE	ISSUE
04/01	SHELL & CORE (PTL2) 60% SET-NOT FOR CONSTRUCTION	10		
04/01	SHELL & CORE (PTL2) CONSTRUCTION SET	11		
04/01	SHELL & CORE (PTL1-1) CONSTRUCTION/PERMIT SET	12		
		13		
		14		
		15		
		16		
		17		
		18		

NOTES:

THE RAY AND MARIA STATA CENTER

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
32 VASSAR ST. CAMBRIDGE MASSACHUSETTS 02139

FRANK O. GEHRY & AS
1520-A CLOVERFIELD BOULEVARD, SANTA MONICA, CALIFORNIA 90404

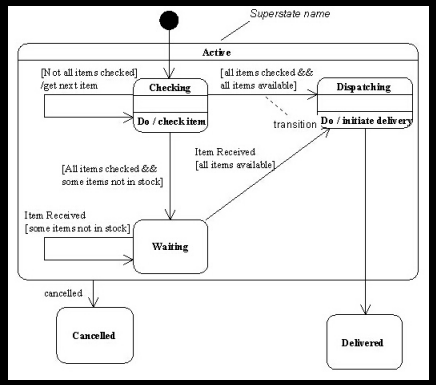
CANNON DESIGN
Associate Architect

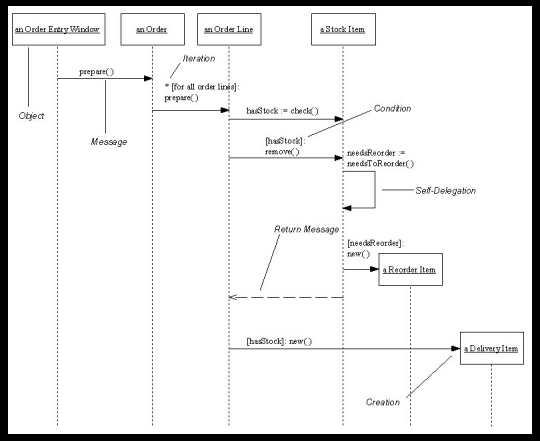
JOHN A. MARTIN & ASSOCIATES, INC.
Structural Engineer

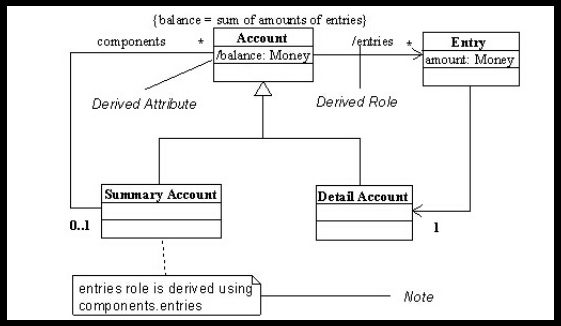
RG VANDE

… of different sorts

three central software models of UML





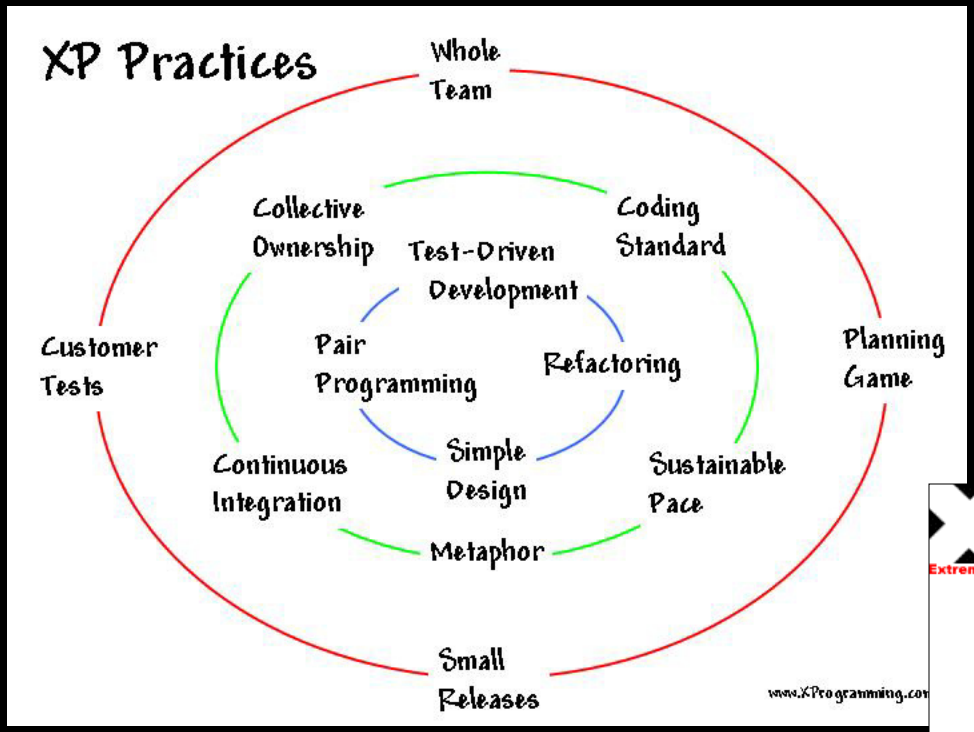


state diagrams
› show transitions
› embedded systems

sequence diagrams
› show messages
› telecomms (eg, SDL)

class diagrams
› show invariants
› relational DB's

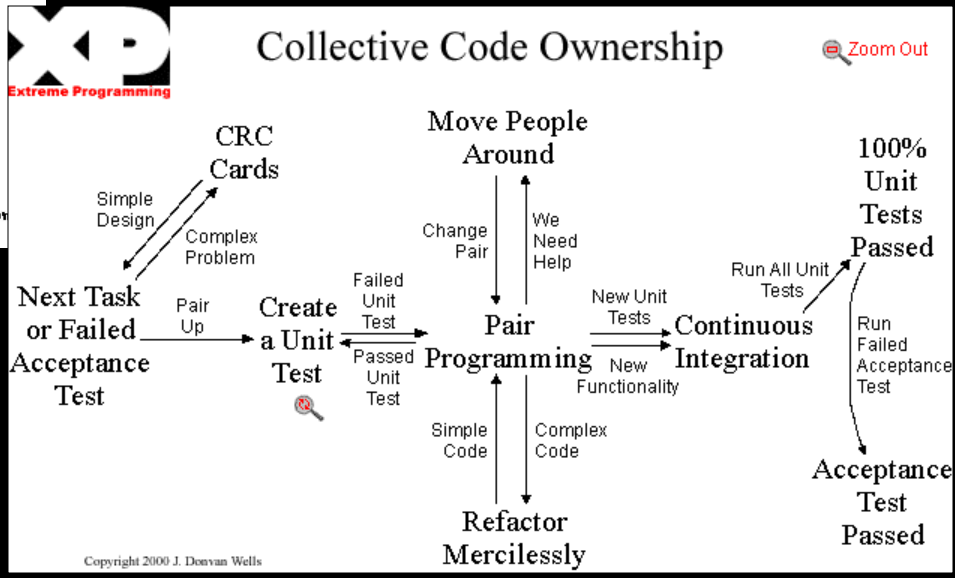
extreme programming



tenets

› no ‘big design upfront’

› design evolves with code by refactoring



from extremeprogramming.org, xprogramming.com

xp on design models

Another strength of design with pictures is speed. In the time it would take you to code one design, you can compare and contrast three designs using pictures. The trouble with pictures, however, is that they can't give you concrete feedback… The XP strategy is that anyone can design with pictures all they want, but as soon as a question is raised that can be answered with code, the designers must turn to code for the answer. The pictures aren't saved. *-- Kent Beck (2000)*

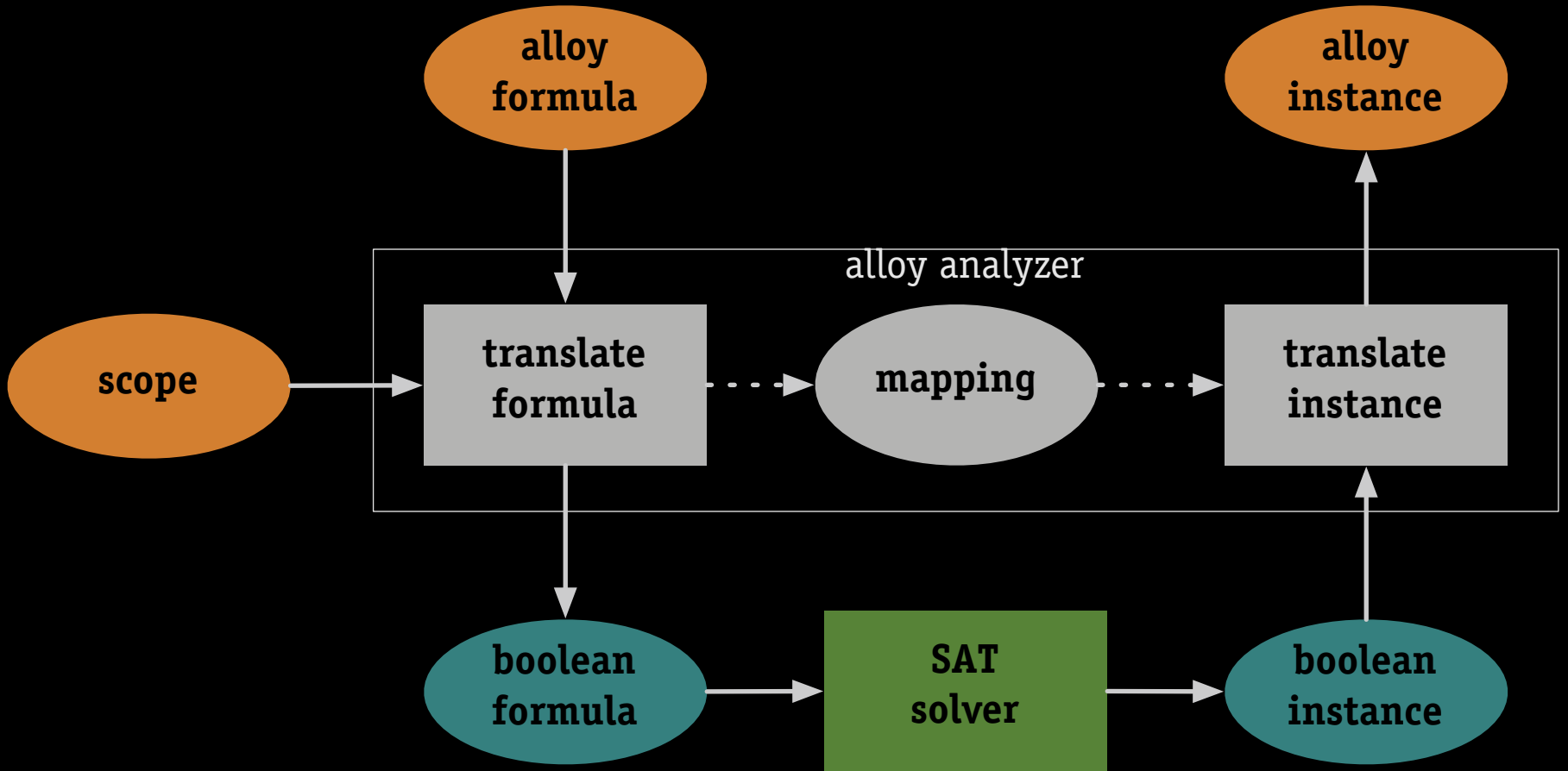
alloy: analyzable, incremental models

a missing ingredient
› design models must be analyzable
› are software engineers the only ones not to use computers?

alloy: a new approach
› a simple, powerful modelling language
› an analysis tool for concrete feedback
› based on properties: to characterize system & expectations

unlike XP
› user doesn't write any test cases or expected results
› model is partial: focus on what matters
› exhaustive analysis: billions of cases in seconds

how alloy works



example: formatting text

Preface

Introduction

In every large software system, there is a small model trying to get out. It's the model that you'd get if you cleared away all the clutter – all the irrelevant details, unused features, performance hacks and workarounds. It captures the essence of the system – what it's about, its key concepts and how they fit together.

Rationale

If the designers had written it down, the maintainers wouldn't need to struggle through the source code, putting the model together like a jigsaw puzzle. The testers would know where weak spots in the implementation are likely to be, and wouldn't have to fumble in the dark. The users wouldn't have to learn the system function by function, because the authors of the user manual would have seen more clearly what all the functions had in common.

Preface

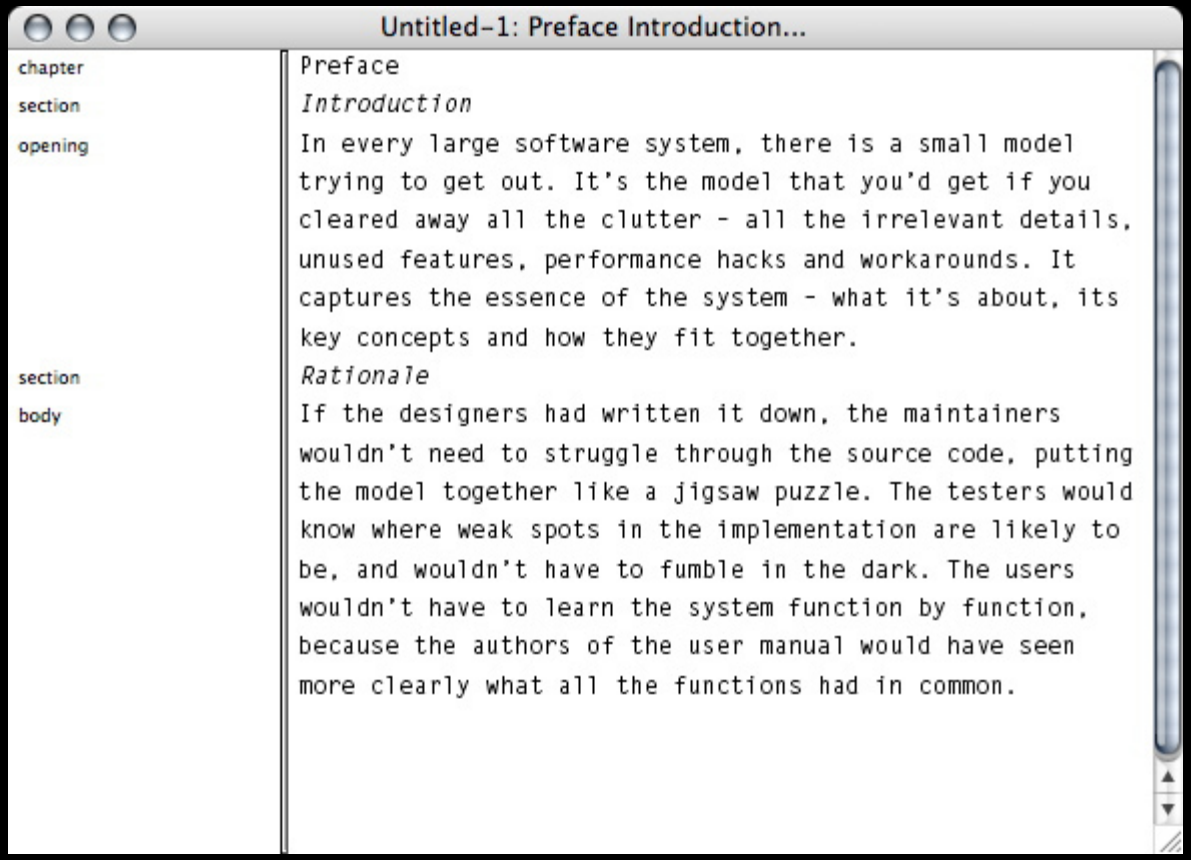
Introduction

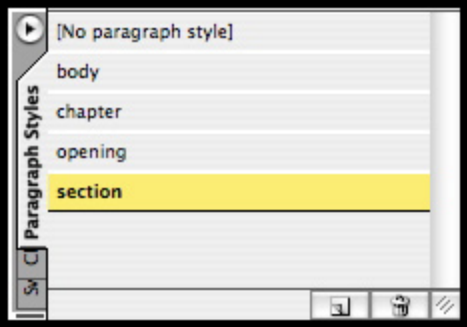
In every large software system, there is a small model trying to get out. It's the model that you'd get if you cleared away all the clutter – all the irrelevant details, unused features, performance hacks and workarounds. It captures the essence of the system – what it's about, its key concepts and how they fit together.

Rationale

If the designers had written it down, the maintainers wouldn't need to struggle through the source code, putting the model together like a jigsaw puzzle. The testers would know where weak spots in the implementation are likely to be, and wouldn't have to fumble in the dark. The users wouldn't have to learn the system function by function, because the authors of the user manual would have seen more clearly what all the functions had in common.

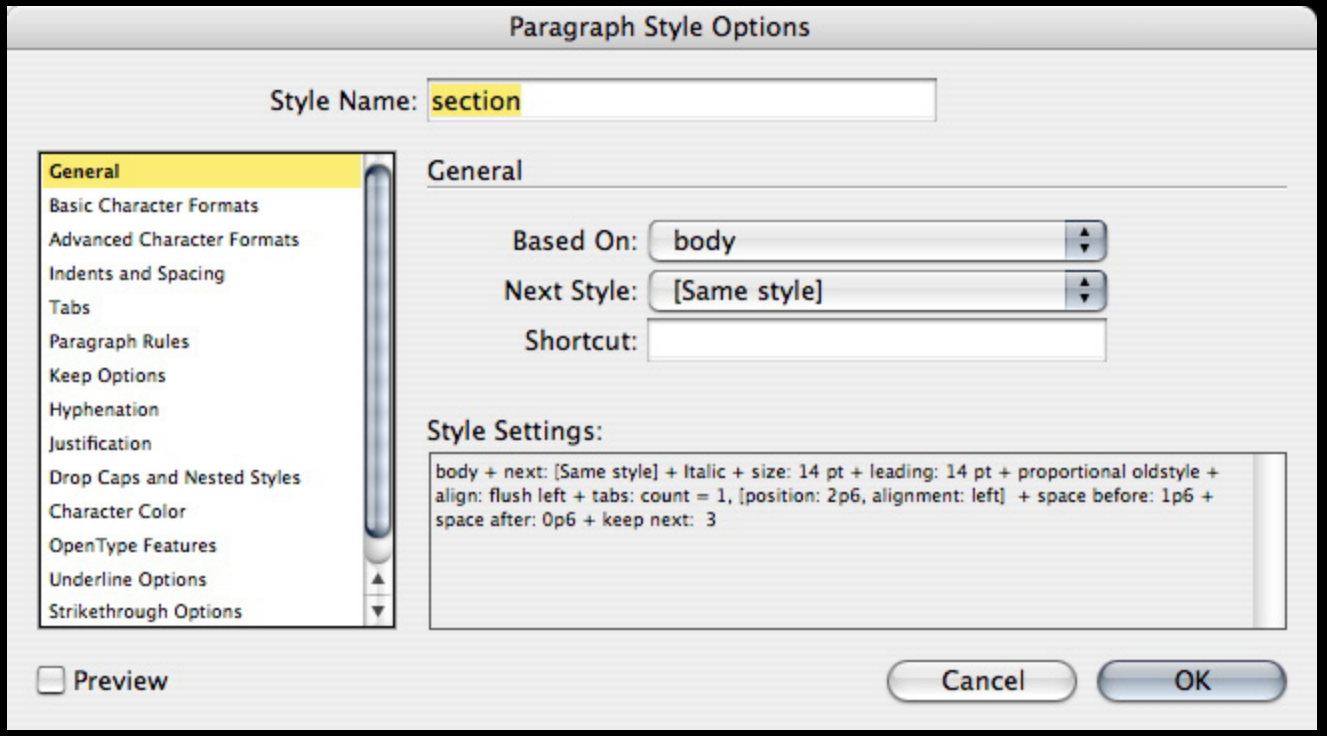
a styled document





Bravo, Xerox PARC (1973-79), Lampson/Simonyi
first use of style sheets in software

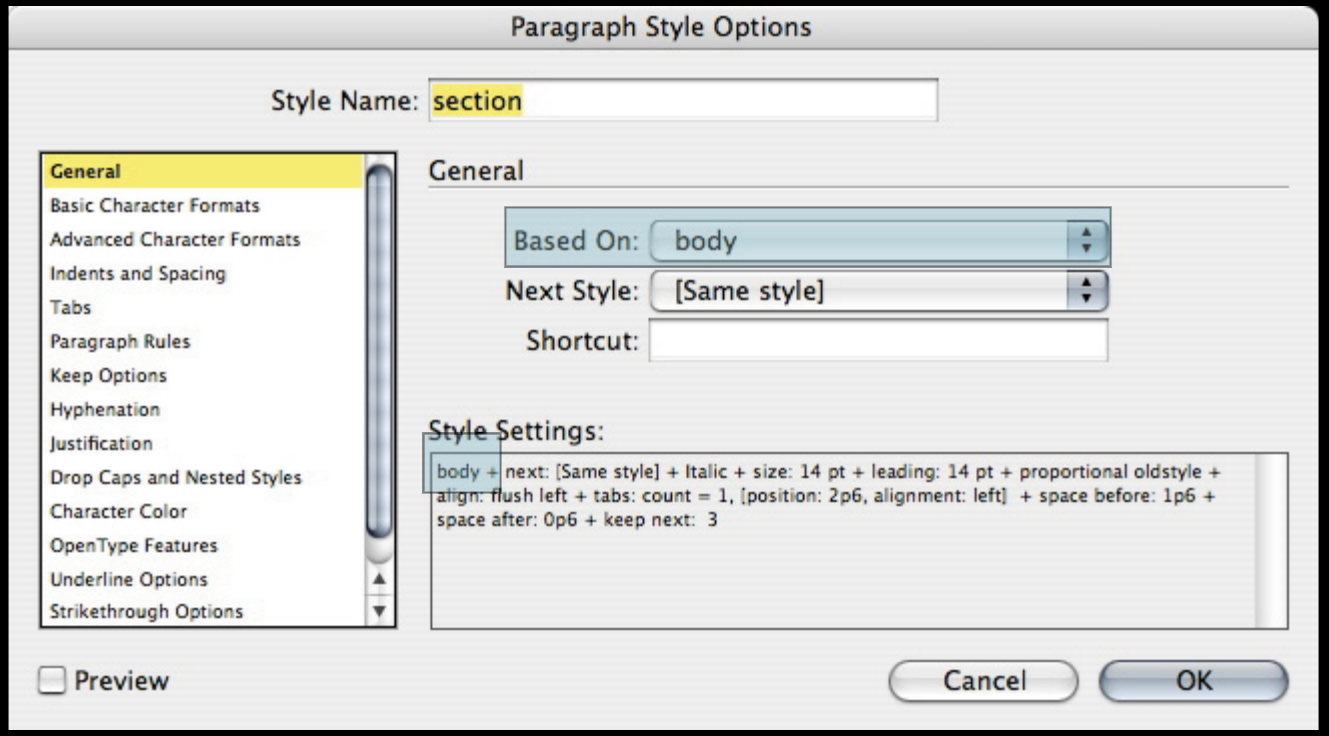
a style sheet dialog



There is no problem in computer science that cannot be solved by an extra level of indirection *--David Wheeler*

style hierarchy

for consistent formatting
› styles arranged in hierarchy, and inherit properties
› change to parent affects child automatically



from microsoft.public.office.word

Certain styles (Heading 1 to 3, Normal...) must exist in the document structure, so they always appear. Word can't let you delete them because the document binary structure would implode if it did.

I have been writing my thesis paper and it has grown to be about 100 pages. The problem that I am having is with the styles preferences... The problem I am having is if I choose and highlight maybe two words to bold, the entire document becomes bold.

I'm having problems with Word 2002 automatically creating new styles. For example, I have a style called "Figures", but for some reason Word has created another style called "Figures Char Char3 Char Char Char Char Char1".

First, make sure you have updated Word 2002 with at least SP-1. That solved a lot of the erroneous Char Char Char problems. Second, it's worth understanding how these 'char' styles are created. In Word 2002, you can have a paragraph in, say, Body Text. If you select *part* of that paragraph and apply a different paragraph style (say, Style 2), then Word creates a kind of hybrid, called Style 2 Char. It's part-paragraph style and part-character style. This only happens if, when you applied the style, some characters were selected, but not the whole paragraph including the paragraph mark.

You can get very bizarre behavior when importing one style that is based on another style without importing the based-on style. This won't be a problem when you update all styles based on a newly attached template. Even if you import the whole set, you will get anomolies unless you import the set multiple times. I import (copy) three times.

I'll look some more, but it seems that the "RedefineStyle" command is buggy in Word2002/2003. Redefining a style shouldn't touch manual formatting. But it seems that "RedefineStyle" removes all manual formatting from all paragraphs formatted in that style. It sure didn't work like this up to Word2000, and whoever thought it a good idea to change this must have ample access to psychedelic chemicals.

a first simulation in alloy

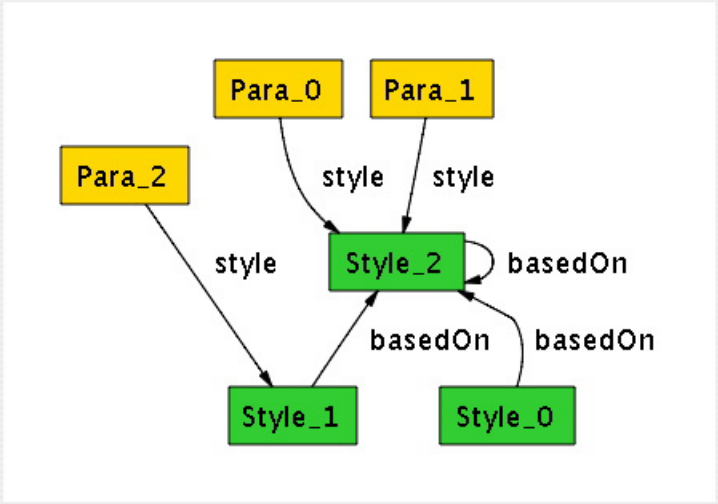
Alloy Analyzer

New Open Save Prefs show [3] Build Execute Viz Tree Text Msgs Layout

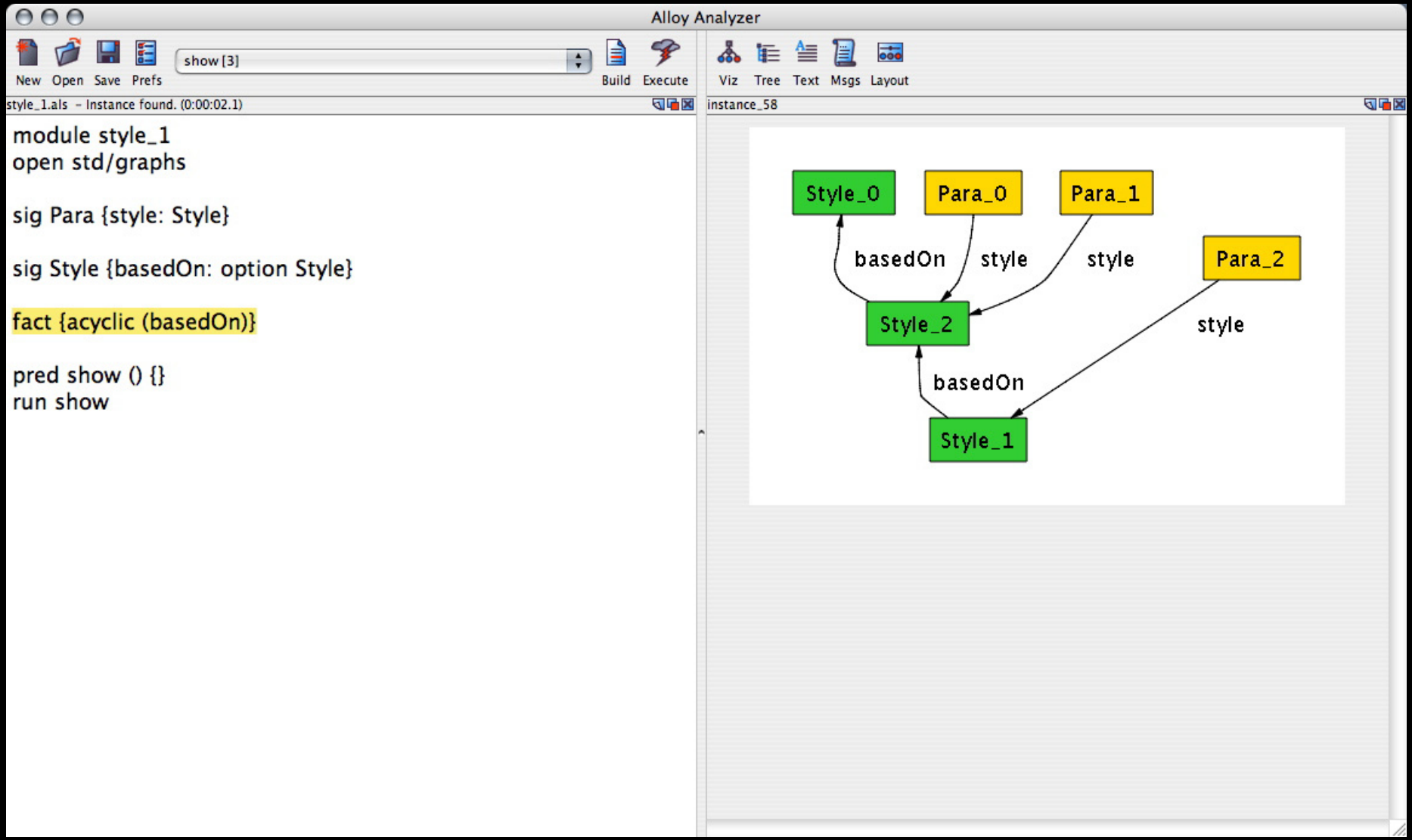
style_1.als - Instance found. (0:00:02.1)

```
module style_1

sig Para {style: Style}

sig Style {basedOn: option Style}

pred show () {}
run show
```

instance_57



fixing the problem



adding value

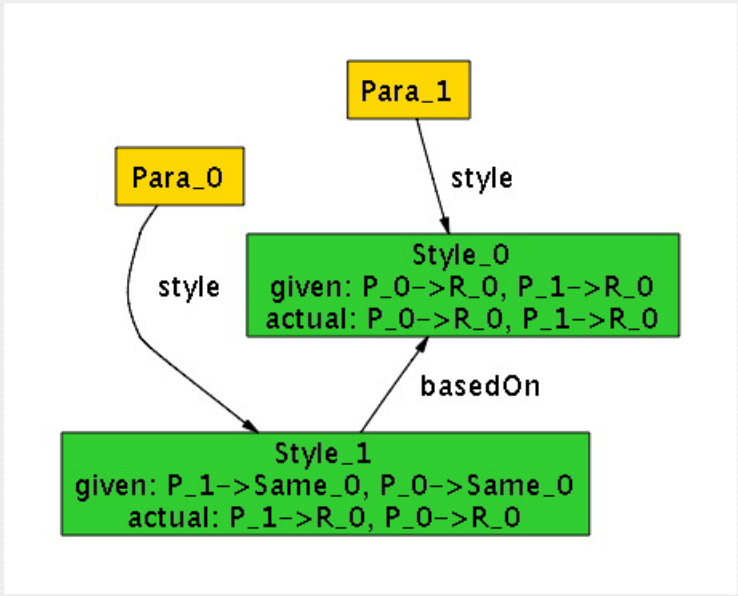
Alloy Analyzer

New Open Save Prefs show [2] Build Execute Viz Tree Text Msgs Layout

style_1.als - Instance found. (0:00:01.9)

```
module style_1
open std/graphs

sig Para {style: Style}

sig Property {}

abstract sig Value {}
sig Real extends Value {}
static sig Same extends Value {}

sig Style {
  basedOn: option Style,
  given, actual: Property ->? Value
  } {
  all p: Property | p.actual =
    if p.given in Same then p.(basedOn.@actual)
    else p.given
  }

fact {acyclic (basedOn)}

pred show () {some given}
run show for 2
```

instance_60



checking an assertion

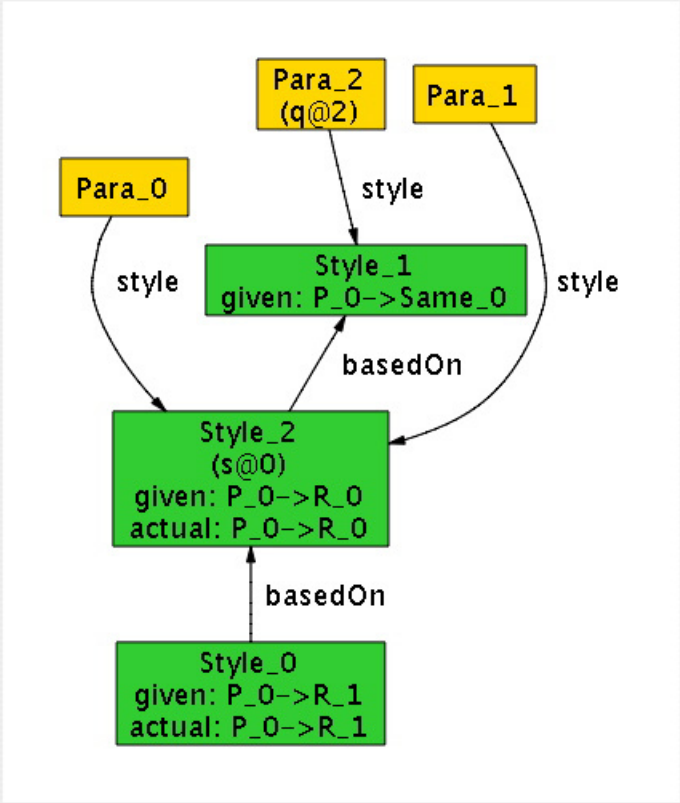
Alloy Analyzer

New Open Save Prefs | AllParasHaveRealFormat [3] | Build Execute | Viz Tree Text Msgs Layout

style_1.als – Counterexample found. (0:00:05.0)

```
module style_1
open std/graphs

sig Para {style: Style}

sig Property {}

abstract sig Value {}
sig Real extends Value {}
static sig Same extends Value {}

sig Style {
  basedOn: option Style,
  given, actual: Property ->? Value
  } {
  all p: Property | p.actual =
    if p.given in Same then p.(basedOn.@actual)
    else p.given
  }

fact {acyclic (basedOn)}

assert AllParasHaveRealFormat {
  all s: Style, p: Property | some s.given[p]
  => all q: Para, p: Property | some q.style.actual [p] & Real
  }
check AllParasHaveRealFormat
```

counterexample_63



doing it for real

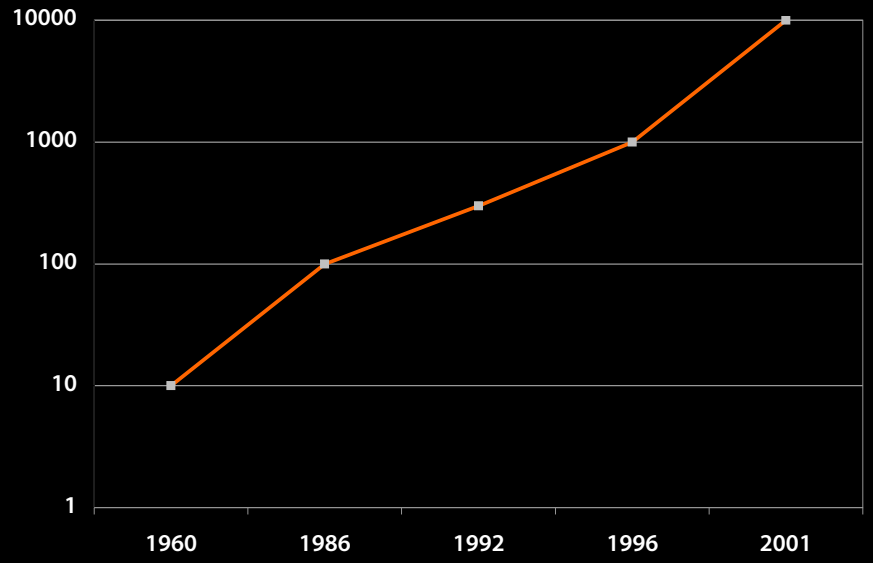
where might this example go?
› multiple style sheets
› operations, eg deleting a style
› numbering scheme

typical experience
› a few hundred lines of Alloy
› analyses from seconds to minutes
› lots of backtracking, thinkos, refactoring

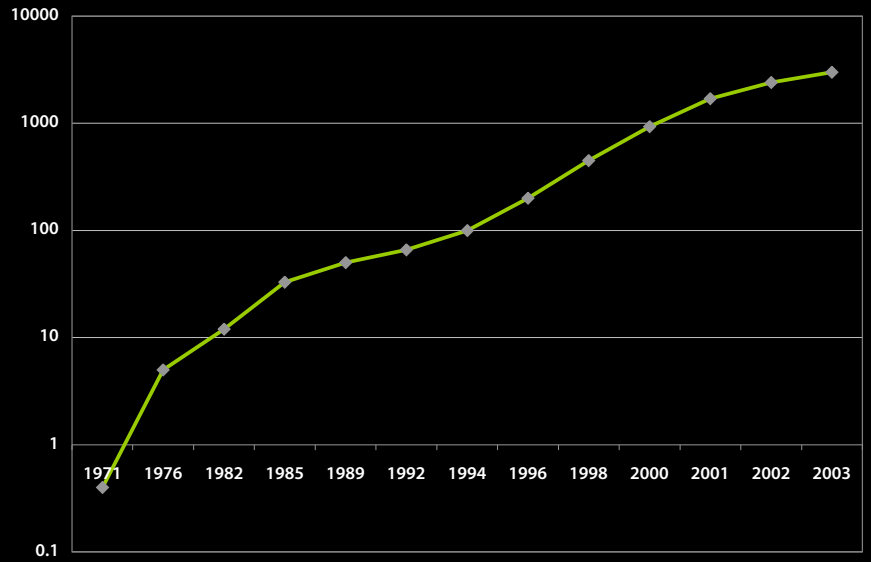
does it scale?
› this example: 62 bits ~ 10^{20} cases
› limit is currently about 500 bits ~ 10^{160} cases

technology advances

advances in SAT solvers
› size of solvable constraint
› in #boolean variables
from Sharad Malik



advances in processors
› speed in MHz
from intel.com



since 1990: factor of 100 from Moore's law, 10^{30} from SAT advances

what has alloy been used for?

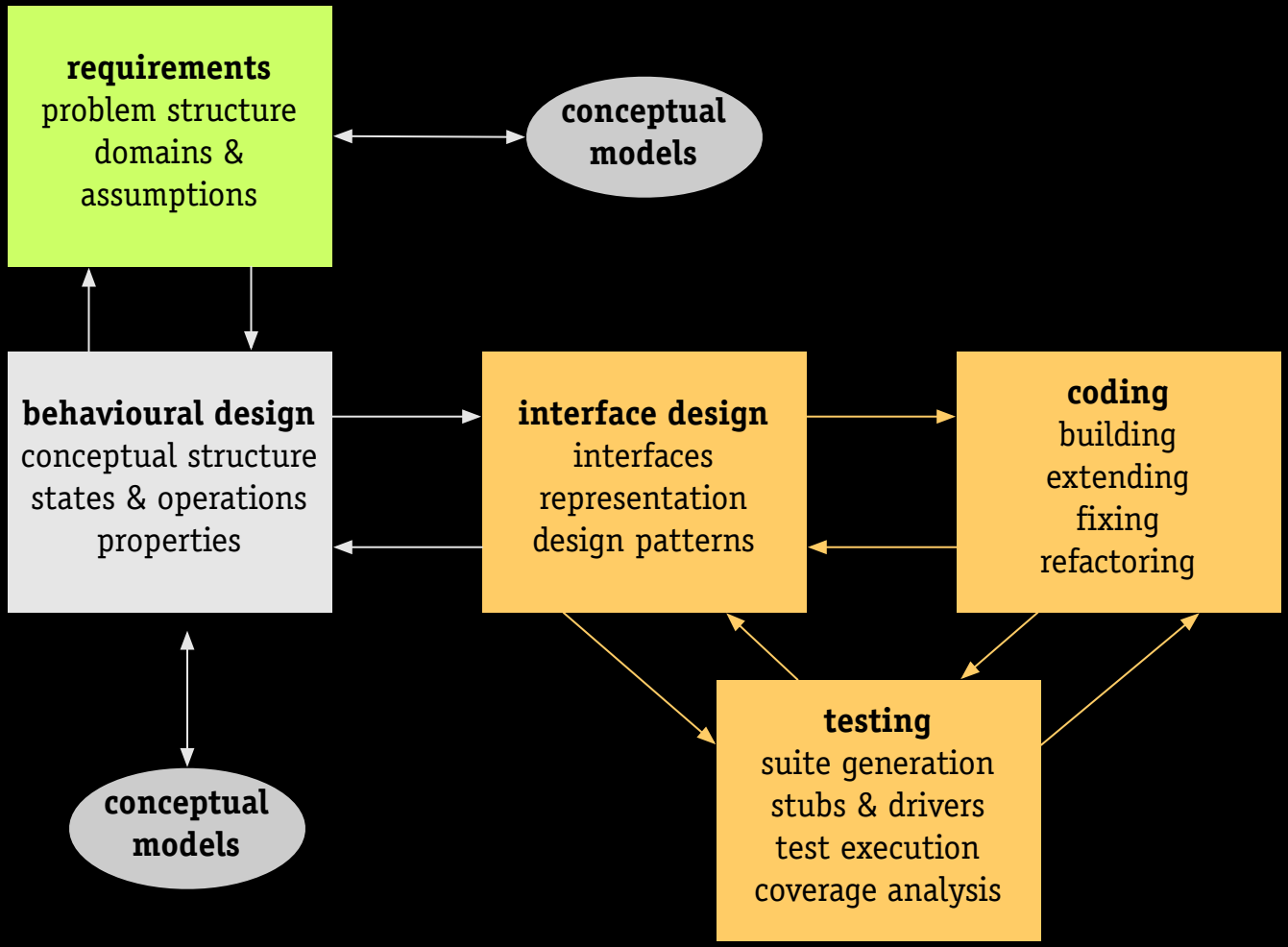
- firewire configuration protocol
- unison file sychronizer
- IMPP presence protocol for instant messaging
- query interface in COM
- key distribution for multicast
- intentional naming
- Chord distributed hash table
- role-based access control
- web ontologies
- military simulation
- telephone switch feature configuration
- proton beam scheduling

is alloy hard to learn?

typical learning experience
› a few years of programming
› minimal background in discrete math
› writing small alloy models in a week
› modelling with confidence in a month

taught in courses
› in US, Canada, UK, Italy, Belgium, Switzerland, Australia, New Zealand, Singapore
› mostly in masters of software engineering degrees

two kinds of design



fred brooks on conceptual integrity

I will contend that 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. *1975*

I am more convinced than ever. Conceptual integrity is central to product quality. *1995*

concluding comments

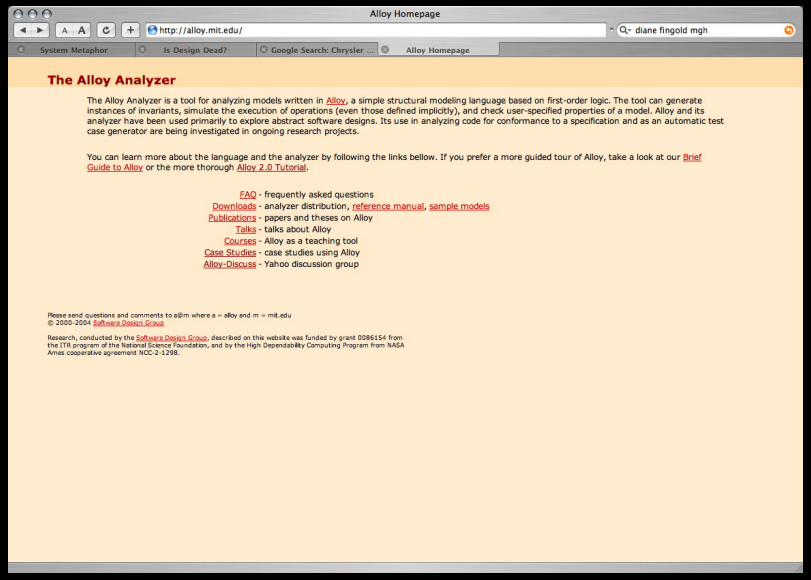
lightweight, analyzable design models
› better medium than code
› better feedback than code
› especially good for conceptual design

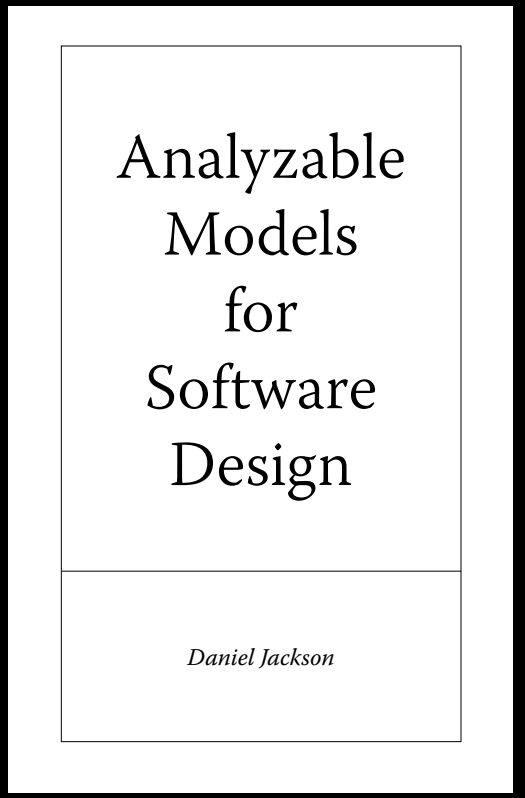
extreme programming
› a risk-driven approach, just like formal methods
› but models have benefits over programs

interface design
› also needs lightweight models

for more information

alloy.mit.edu
› case studies
› courses
› tutorial
› downloads





upcoming book
(Fall 04)

books
› Martin Fowler, *Analysis Patterns*
› Erich Gamma et al, *Design Patterns*
› Michael Jackson, *Software Requirements & Specifications*
› Kent Beck, *Extreme Programming Explained*

extra slides

reactions to UML

too complicated
› UML Reference Manual
    576 pages; #62,915 in amazon.com
› Fowler, UML Distilled
    192 pages; #1,516; 300,000 sold

too burdensome
› inflexible process
› big documentation, little insight

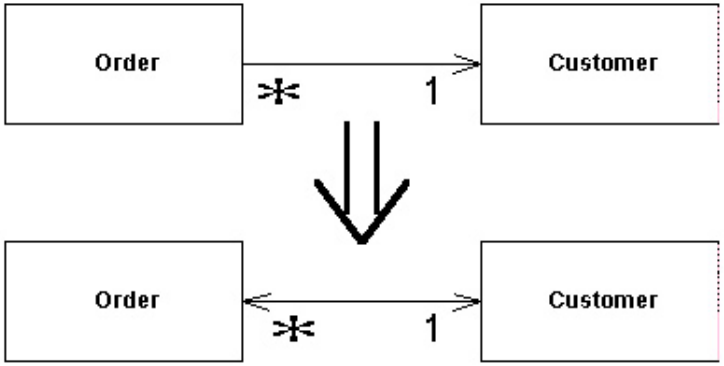
revolution!
› programmers vs. managers
› Elvis vs. Mort

a sample refactoring

Change Unidirectional Association to Bidirectional

You have two classes that need to use each other's features, but there is only a one-way link.

Add back pointers, and change modifiers to update both sets.



For more information see page 197 of *Refactoring*

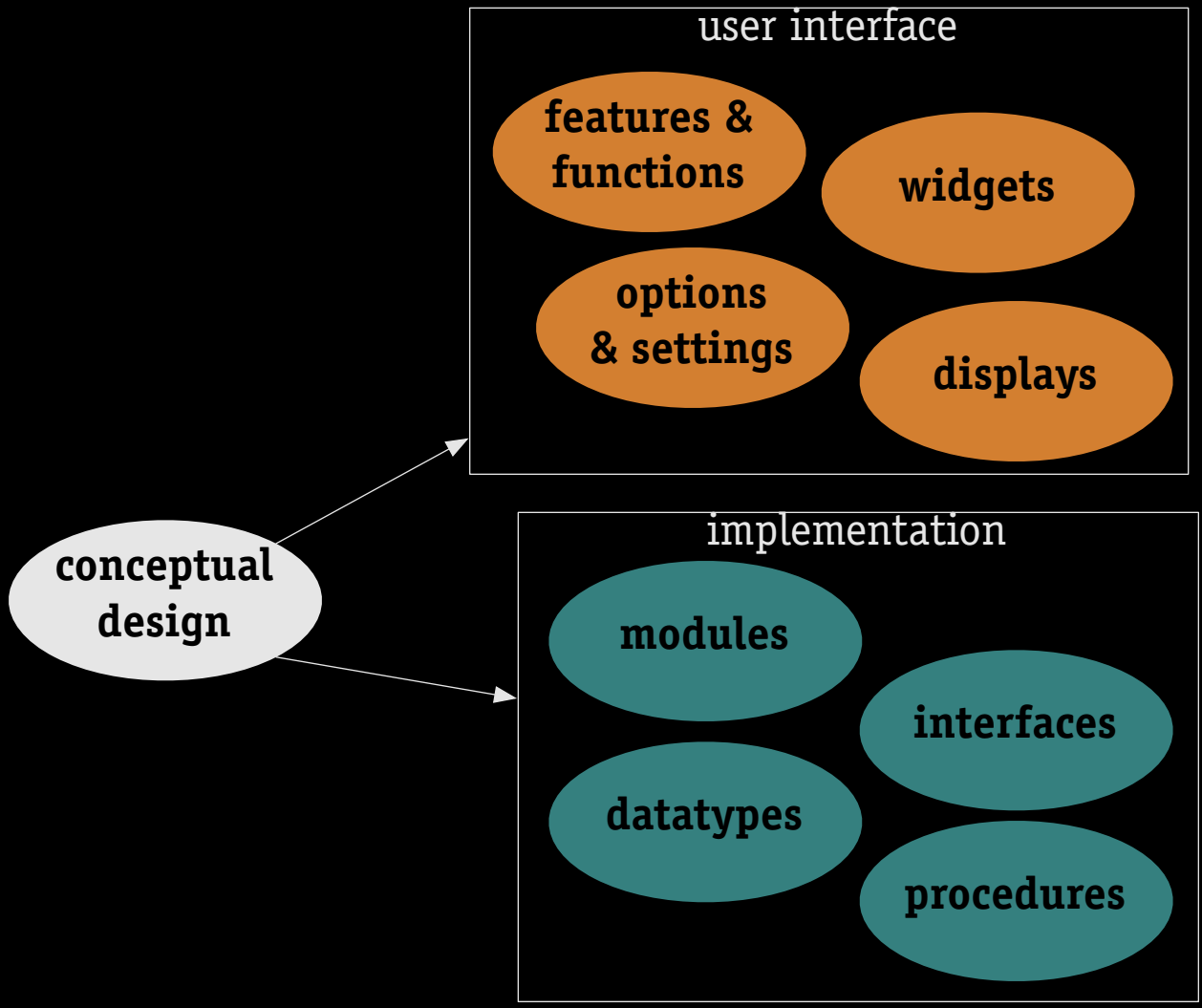
Additional Comments

Doing a remove

In the example I showed an *addOrder* method, but I didn't show the *removeOrder* method. If you want to do a remove, you would write it like the add method but set the customer to null.

```
Class Customer ...
 void removeOrder( Order arg ) {
  arg.setCustomer( null );
}
```

impact of conceptual design



ease of use
flexibility
robustness

clean interfaces
decoupling
extensibility
dependability