Vers and SE

Bart Massey

Portland State University

October 2002

The SE Lifecycle

- Requirements Gathering
- Requirements Specification
- Architectural Design
- Detailed Design
- Implementation + Unit Test + Debugging
- Integration + System Test
- Delivery + Acceptance Test

Requirements Specification

- Question: ***What do we want?***
- Format: Traditionally, numbered natural-language paragraphs
- Content: Must guide and verify design
- Common Defects
  - Inconsistency
  - Insufficient Specificity
  - Incompleteness
  - Incomprehensibility
  - Invalidity

Requirements Inspection

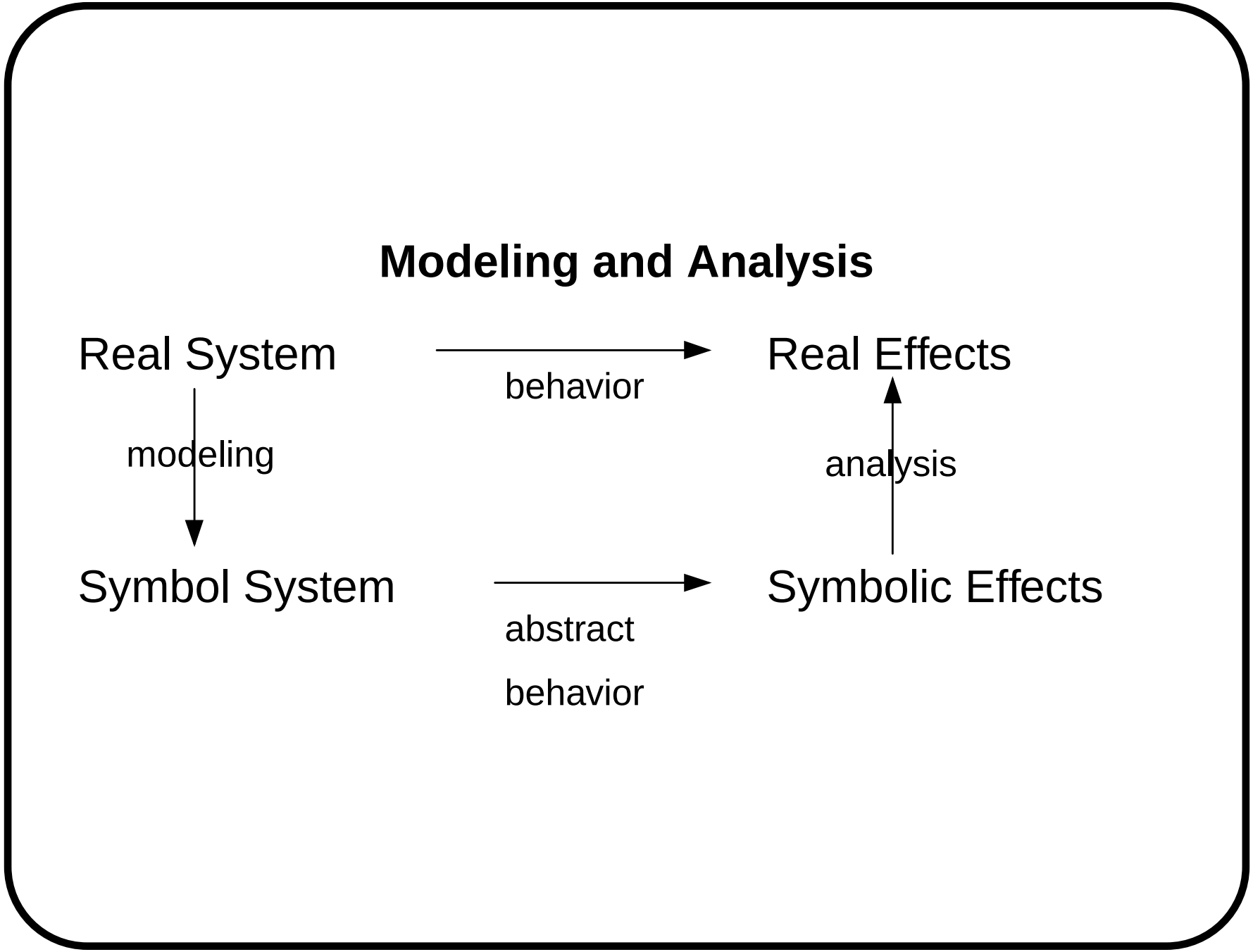
- Preparation: carefully read and annotate
- Inspection Roles: (team size: 4-6)
  - Reader, Moderator, Scribe
- Goal: *identify* issues
- Product
  - List of action items
  - Approve/reject

Requirements and Design

- Req. Spec. and Arch. Design are tightly interlaced
- Separation is a good thing
- This part of process less well-understood than others
- Relationship impacts req. spec. methodology, tools

What is Z?

- Formal logic and set theory
- Finite and integer sorts
- Models world
- Specifies constraints


Modeling and Analysis
Real System
behavior
Real Effects
modeling
analysis
Symbol System
abstract
behavior
Symbolic Effects

Z Paragraphs

Z description is natural-language +

- Entities
- Axioms
- Schemata

Z Definitions

Set existence

$$[FISH]$$

$$COIN ::= penny \mid nickle \mid dime \mid quarter$$

Functions and constraints

$$least\_change : 0 \,..\, 99 \rightarrow bag\ COIN$$

$$\forall n : 0 \,..\, 99 \bullet elems(least\_change\ n) \leq 9$$

Z Schema

Like records: indicate state change

VendingMachine

$$change : bag\ COIN$$

$$elems(change) \leq 1000$$

Change in Z

Schema can show state change

Sell

$$\Delta \mathit{VendingMachine}$$

$$\mathit{payment}?, \mathit{price}? : \mathbb{N}$$

$$\mathit{change}' = \mathit{change}\ \mathit{less}\ \mathit{least\_change}(\mathit{payment}? - \mathit{price}?)$$

$$\mathit{payment}? \geq \mathit{price}?$$

Why Z For Vers?

- Consistency
- Completeness
- Precision
- Accuracy
- Terseness
- Verifiability

The Vers Environment

$$[USER, DIR, FILE, FILENAME]$$

$$| \; fs : DIR \times FILENAME \pfun FILE$$

$$| \; cwd : USER \fun DIR$$

Internals

$$VERSION == \mathbb{N}_1$$

$$LOCK\_STATE ::= locked\_by \langle\langle USER \rangle\rangle \mid unlocked$$

+ Repository

The Repository

Delta

$$version : VERSION$$
$$file : FILE$$

$$HISTORY == \mathrm{seq}\, Delta$$

Record

$$lock\_state : LOCK\_STATE$$
$$history : HISTORY$$

Repository

$$repository : FILENAME \pfun Record$$

Most Recent Version

$$mrv : Record \rightarrow VERSION$$

$$\forall r : Record \bullet$$
$$mrv\ r = (r.history\ 1).version$$

Creating A New Record

$$new\_record : USER \times FILE \longrightarrow Record$$

$$\begin{aligned}
&\forall u : USER;\ f : FILE;\ r : Record;\ d : Delta \bullet \\
&\quad r = new_record\,(u, f) \land \\
&\quad r.lock_state = unlocked \land \\
&\quad r.history = \langle d \rangle \land \\
&\quad d.version = 1 \land \\
&\quad d.file = f
\end{aligned}$$

Changing The Repository

InitRepository

$$Repository'$$

$$repository' = \varnothing$$

New

$$\Delta Repository$$

$$filename? : FILENAME$$

$$user? : USER$$

$$repository' = repository \cup \{filename? \mapsto new\_record(user?, fs(cwd\ user?, filename?))\}$$

$$filename? \notin \mathrm{dom}(repository)$$

Conclusions

- Important to inspect *Vers* reqs
- Z formalization helps with reqs
- Reqs are hard