ordering events

Daniel Jackson

For now, all rights reserved. Daniel Jackson, 2012.
a strategy

divide events into
› operations performed by user
› responses from system

for each operation
› figure out what it should do
› and what responses it should send

good? not initially: fragmentation

Tardis control panel
a better strategy

start by considering traces

what’s a trace?
› a sequence of events
› a history of behavior

behavior
› is set of all possible traces
› can actually define behavior fully like this
› for counter: <get!0, up, get!1, down, get!0>

but we’ll focus on partial description
› no event arguments
› regular language of traces
behavior as regular language

regular expression
› E*: zero or more E’s
› E+: one or more E’s
› [E]: optional E
› (E | F): E or F
› e: an event e

example
› Switch ::= (up down)*
› Account ::= (open (payin | withdraw)* close)
› Course ::= [prereg] (register | add) (drop | complete)

state machine
› equivalent way to express: can you drawn them?
multiple constraints

no need to do it in one!
  › can write multiple constraints
  › parallelism, or intersection of languages

example: requesting reminders and resetting passwords
  › SocialNetwork ::= 
     (join (post | reminder | reset)* leave) 
     || 
     (reminder reset)*

can split into roles
  › User ::= JoinLeave || Password
  › JoinLeave ::= (join (post | reminder | reset)* leave)
  › Password ::= (reminder reset)*
entity life histories

many instances of an entity
  › then separate language for each

example: library
  › User ::= (join (request | borrow | return | renew)* leave)
  › Book_i ::= ([request] borrow renew* return)*

can use to identify entities
  › in object model!
why are we doing this?

**identifying events**
- scopes the system (functionality, service life)
- exposes problems with undetectable events
- clarifies problem domain

**ordering events**
- exposes difficulties
- clarifies human protocols

**together**
- subset of events become operations in web service API
- orderings tell you what your actions must handle
- ActiveRecord::StateMachine supports directly
use cases

focus on user & interface
more than a trace but less than a grammar
where’s the full service life?
user stories

As a student, I want to purchase a parking pass so that I can drive to school.

Priority: Should
Estimate: 4

As a substitute teacher, I want to view a list of all students in my course, so that I can call on them by name.

limited to user
fragmented approach
only a small part
of a long story!
good way to
stimulate analysis of
problem scope,
preceding event
modeling
SDL

UML sequence diagram
(aka message sequence chart)

UML activity diagram