

Debugging Software Architectures

Kyungsoo Im and John D. McGregor
School of Computing
Clemson University
Clemson, SC 29634



Outline

- Motivation
- Related Work
- Research Approach
- Summary & Future Work



Motivation

- An incorrect software architecture can lead to problems during development
- Architecture descriptions are becoming larger and more detailed -> more possibility of bugs
 - Ex) Avionics Display System: 21,000 lines
- Expensive to correct the software architecture in later stages
- Goal: aid the software architect in locating known defects in software architectures



Motivation

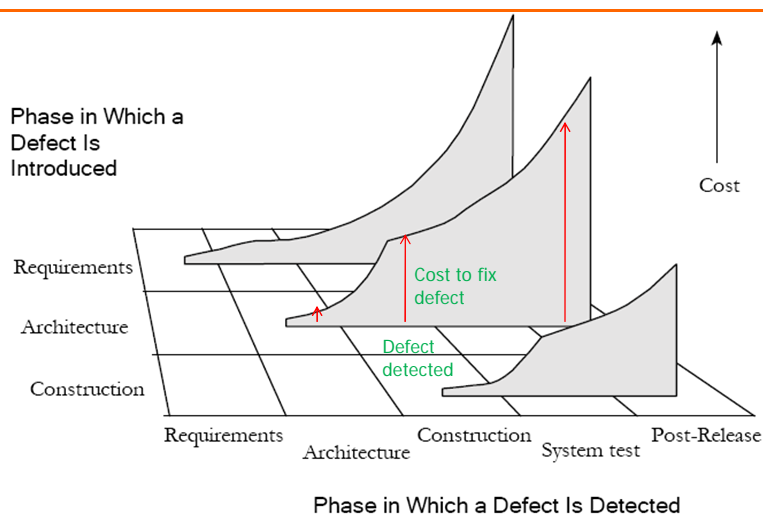


Image from *CodeComplete* 2nd edition by Steve McConnell



Related Work

- Much work exists on software architecture analysis
 - Most only point out existence of defect and not its location
- Debugging UML Designs
- Little work on debugging software architectures
 - Visualization of event traces, Monitoring of events, Simulation
 - No clear definition, process, or method to debugging software architectures



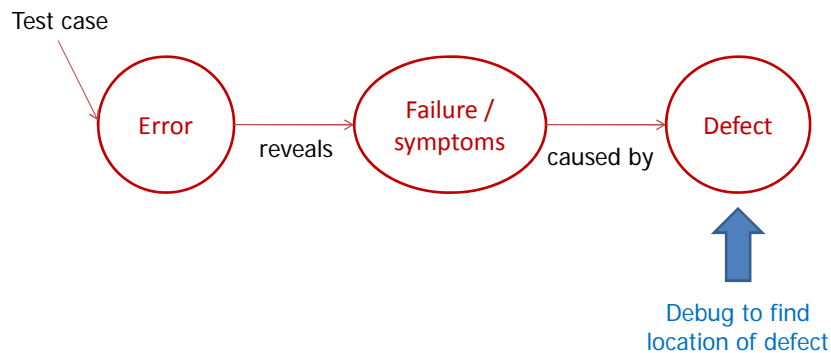
Research Approach

- Define debugging at the architectural level
- Develop a classification of architectural defects
- Develop techniques for tracing a defect to its cause through debugging



Definitions

- Mirror debugging at program level

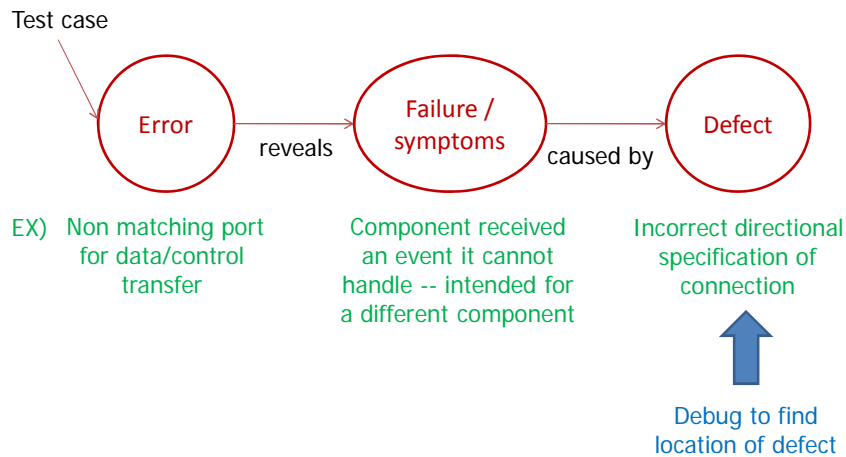


Definitions

- **Software Architectural Error** - Difference exist between actual and expected software architecture
- **Software Architectural Failure** - Inability of a software architecture to meet a functional or nonfunctional requirement
- **Software Architectural Defect** – Incorrect, incomplete, or inconsistent architectural specification, behavior, or design



Definitions



Architectural Defects

- Classification of software architectural defects
 - Helps understand possible types of defects
 - Depending on defect types, debugging methods will be different
- Defects can be found at 2 levels
 - Structural, Behavioral
- Defects regarding functional and non-functional requirements

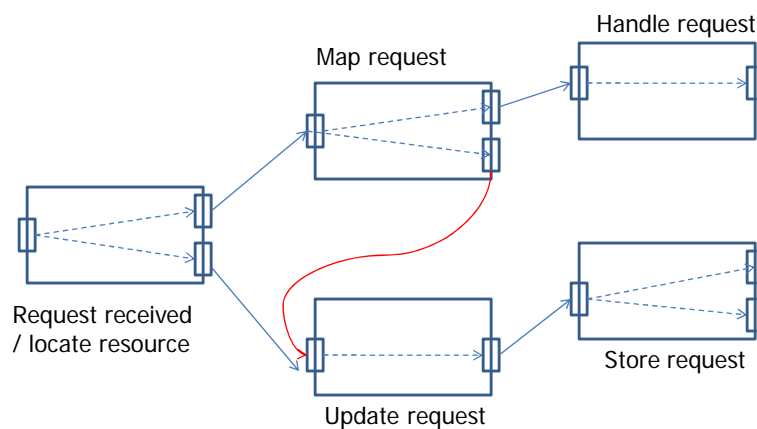


Structural Defects

- Syntactic defects
- Directional defects on connections, flows
- Missing or unintended connections or flows
- Data type mismatches
- Unused components
- Not matching the architectural pattern used
- Too much / too little modularity
- Failure to meet nonfunctional requirements (ex: modifiability)
-



Structural Defect Example



- Scenario cannot be fulfilled because of missing connection
- For structural defects, the failure usually defines region of interest to find defect



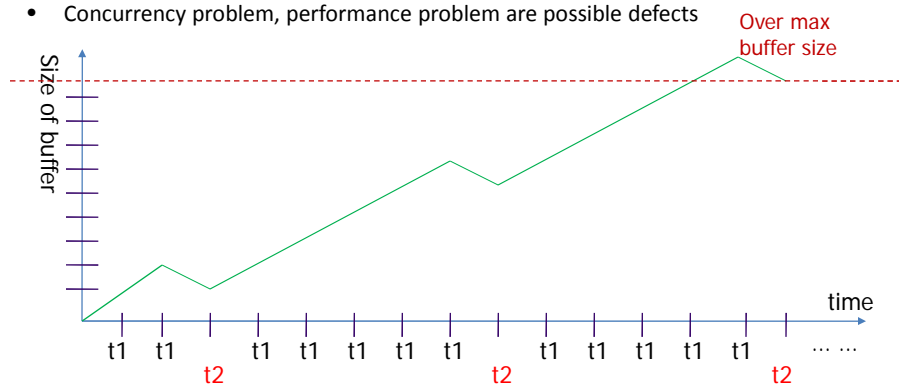
Behavioral Defects

- Receive unexpected event
- Expected event not sent
- Missing activity
- Extraneous activity
- Concurrency issues
- Execution on incorrect states
- Pre / Post conditions violations
- Failure to meet nonfunctional requirements (ex: performance)
-



Behavioral Defect Example

- Suppose there are two executing threads – one producing data (t1), one consuming data (t2)
- Assume t1 produces and t2 consumes the same size chunks
- Simulation shows the timestamps of the threads executing, where t2 lacks the speed of t1
- Concurrency problem, performance problem are possible defects



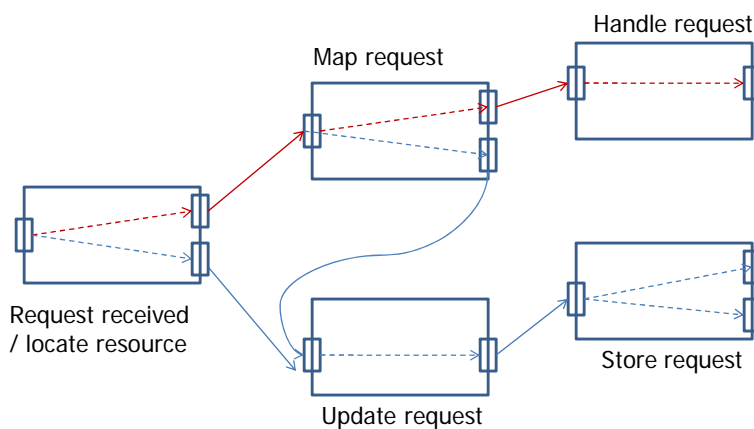


Debugging Approaches

- Our approach would be heterogeneous, including
 - Back trace a failed scenario
 - Localize defects through multiple runs of scenarios
 - Overlapping areas, divide and conquer
 - Apply software architecture slicing
 - Perform Simulation
 - Perform Model Checking
- In some cases, localizing a defect to a region in the architecture may be the correct result



Architecture Slicing



-----> ----->
Indicates the slice through the architecture



Simulation

The screenshot displays the ADeS simulation environment. The top menu bar includes File, Edit, Navigate, Search, Project, ADeS, Run, Window, and Help. The interface is divided into several panels:

- Navigator:** Shows a tree view of the simulation model with components like tm3, result, .trace, snapshot, .snp, .ades, and .project.
- Instances:** Lists the current instances, including testMode03_S_I_Instance.
- Instances Attributes:** A table showing attributes for the selected instance, such as Name, State, Actions list, and Properties.
- Event Manager:** A table showing the sequence of events, including their Id, Date, Priority, Type, Source, and Destination.
- Console:** Displays the execution log, showing thread and processor activities, actions, and events.

Attribute	Value
Name	testMode03_S_I_Instance.p.t2
State	AWAITING_DISPATCH
Actions list	None
Properties	
Period	40 ms
Deactivate_Deadline	5 ms

Id	Date	Priority	Type	Source	Destination
129	90 ms	2	CALL_BACK	evg_0_testMode03_S_I_Instance.p.t2	evg_0_testModel
146	90 ms	2	CALL_BACK	evg_2_testMode03_S_I_Instance.p.t2	evg_2_testModel
226	90 ms	2	CALL_BACK	evg_3_testMode03_S_I_Instance.p.t1	evg_3_testModel
142	90 ms	4	HYPERPERIOD_END	SOM Manager	SOM Manager
149	90 ms	5	TIME_OUT	timer testMode03_S_I_Instance.o.t2	timer testMode03_S_I_Instance.o.t2

```
[ 80 ms] thread 'S_I.p.t1' treats action compute(5 ms)
[ 80 ms] thread 'S_I.p.t1' has the state RUNNING
[ 85 ms] thread 'S_I.p.t1' treats event ACTION_ENDED sent by thread 'S_I.p.t1'
[ 85 ms] thread 'S_I.p.t1' has no actions list.
[ 85 ms] thread 'S_I.p.t1' has the state AWAITING_DISPATCH
[ 85 ms] processor 'S_I.proc' treats event ELEMENT_COMPUTATION_ENDED sent by processor 'S_I.proc'
```

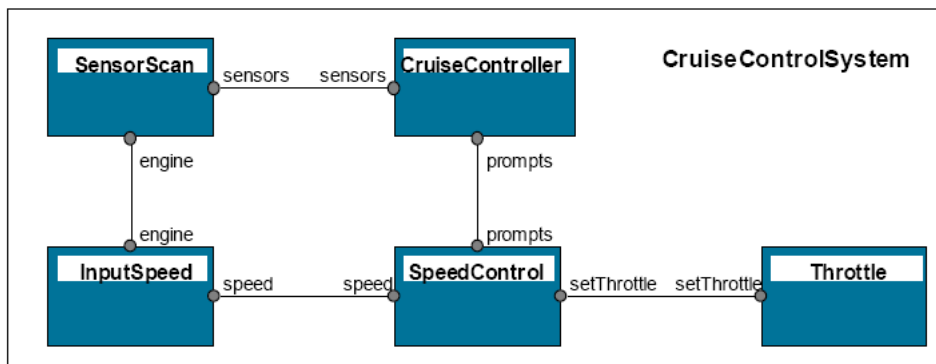


Model Checking

- Used frequently to find design flaws
 - Deadlock, starvation, unreachability, constraint violations
- Same flaws exist in software architectures
- Provides a counter example/error trace when flaw is found



Example of Deadlock



- Example and image from “Exposing the Skeleton in the Coordination Closet” by J. Kramer, J. Magee
- This example used by author in classes for 3 years before finding this flaw.



Example of Deadlock

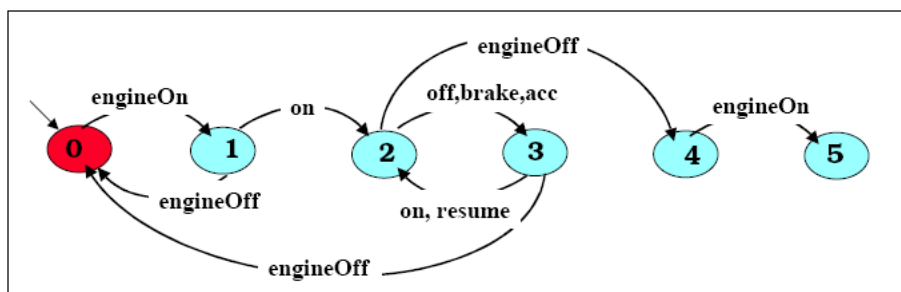


Image from “Exposing the Skeleton in the Coordination Closet” by J. Kramer, J. Magee

- Deadlock exists in the software architecture in the event of engineOn, cruiseControlOn, engineOff, and then engineOn again.
 - Reason: SpeedControl never disabled, leading to a deadlock state.



Model Checking with JSpin

```

File Edit Spin Converg Options Setting Output SpinSpide Help LTL formula
Open Check Random Interactive Trail Weak fairness Safety Verify Stop Translate Clear Load SpinSpic
(SPEEDCONTROLLING == false) -> current?ON -> CRUISING
22 }
23
24
25 proctype engon()
26 {
27   current?ENGINEON -> ENGINEERUNNING = true;
28 }
29
30
31 proctype cruoff()
32 {
33   current?OFF -> CRUISING = false;
34 }
35
36
37 proctype ActionType()
38 {
39   end:
40   do
41     :: (action == NONE) -> action = ENGINEON -> cur
42     :: (action == ENGINEON) -> action = ON -> curre
43     :: (action == ON) -> action = OFF; current!OFF;
44     :: (action == OFF) -> action = ENGINEOFF;

```

```

pan: invalid end state (at depth 14)
pan: wrote EX-Cruise3.pml.trail
(Spin Version 4.3.0 -- 22 June 2007)
Warning: Search not completed
+ Partial Order Reduction
Full statespace search for:
  never claim -
(none specified)
  assertion violations +
  cycle checks -
(disabled by -DSAFETY)
  invalid end states +
State-vector 40 byte, depth reached
16, ... errors: 1 ...
  11 states, stored
  0 states, matched
  11 transitions (=
stored+matched)
  4 atomic steps
hash conflicts: 0 (resolved)
2.302 memory usage (Mbyte)

```

```

c:\jspin\jspin-examples\pan -m2000 -X ... done!
bin\spin.exe -a -v EX-Cruise3.pml ... done!
bin\spin.exe -a EX-Cruise3.pml ... done!
c:\mingw\bin\gcc.exe -DSAFETY -o pan pan.c ... done!
C:\jspin\jspin-examples\pan -m2000 -X ... done!

```



Error Trace

```

14:   proc 1 (ActionType) line 40 "pan_in" (state 14)      [((action==ENGINEON))]
15:   proc 1 (ActionType) line 42 "pan_in" (state 5)      [action = ON]

spin: trail ends after 15 steps
#processes 5:
15:   proc 0 (:init:) line 58 (state 7)
      -end-
15:   proc 1 (ActionType) line 42 (state 6) (invalid end state)
      current!ON
15:   proc 2 (engon) line 30 (state 5)
      -end-
15:   proc 3 (cruon) line 21 (state 5) (invalid end state)
      {(SPEEDCONTROLLING==0)}
15:   proc 4 (cruoff) line 33 (state 3) (invalid end state)
      current?OFF

```



Summary & Future Work

- Outline of approach to debugging software architectures
- Overview of what is debugging at software architecture level & how to achieve it
- Debugging an architectural defect dependent on type of defect
- Extend debugging for all architectural defects
 - Especially, how to debug a quality attribute
- Implement the debugging techniques to be used as tools by the architect



Questions?
