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

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

Test case

Error  

reveals

Failure / symptoms

caused by

Defect

Debug to find location of defect

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

Test case reveals Failure / symptoms caused by Defect

- Error: Non matching port for data/control transfer
- Failure / symptoms: Component received an event it cannot handle -- intended for a different component
- Defect: Incorrect directional specification of connection

Debug to find location of defect

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

Size of buffer

t1 t1 t1 t1 t1 t1 t1...
t2 t2...

Over max buffer size

time
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

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

Model Checking
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

- 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

Error Trace

14: proc 1 (ActionType) line 46 "pan_in" (state 14) $1 \equiv (\text{action} \equiv \text{ENGINEERING})$

15: proc 1 (ActionType) line 42 "pan_in" (state 5).setAction \equiv \text{ON}

spin: trail ends after 15 steps

#processes 5:

18: proc 0 (init): line 38 [state 7]

19: proc 1 (ActionType) line 42 (state 6) (invalid end state)\
\text{current\_ON}

20: proc 2 (engine): line 30 (state 5)

21: proc 3 (crown): line 21 (state 5) (invalid end state)\
\text{SPEED\_CONTROLLING} \\ 0

22: proc 4 (crui\_off): line 33 (state 8) (invalid end state)\
\text{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?