
Design Architecture for PL Testing

Tim Trew
Harumi Watanabe
Tomoji Kishi
Marlon Vieira

Elevator Statement

- Software complexity is increasing and PL variability increase that complexity. It is necessary to find ways for managing the testing process for those complex (PL) software.
 - Testing can became the bottleneck of software development.
 - Decision make in the architecture should be mapped onto PL implementation testing.
 - Architecture decision impact system testing, e.g.
 - Guaranteeing independence of features to reduce combination testing
 - Guaranteeing stability of features to reduce product instance retest obligations
-

Open Points

- What aspects of architecture should help on inspections and testing PL ?
 - How to design PL architecture in order to reduce the number of necessary test cases?
 - What kind of notation should be used to construct PL testable architecture ?
 - Importance of specific behavioral models (e.g., Petri nets) to help on describing PL architecture.
 - UML is a standard but it lacks some features for modeling specific aspects of a PL (e.g., multi threads and other variability mechanisms).
 - ADLs?
-

Good Practices

- Inspection (design and code) is a powerful tool for reaching PL quality
 - PL should not be implemented until we get to a point where we have test design or plan
 - Design for testability (e.g., components in sequence or in parallel depending on the application).
 - Assertions in architecture or design to improve testability (find problems)
 - 3 different aspects in testability:
 - Probes points in the right place – controllability, observability.
 - Granularity of probes points– probes more robust and coherent.
 - Design for plug and play. Integration test obligations on local test obligation.
-