

# Deriving Tests from Software Architectures

**Jeff Offutt**

Information & Software Engineering

George Mason University

Fairfax, VA USA

[www.ise.gmu.edu/faculty/ofut/](http://www.ise.gmu.edu/faculty/ofut/)

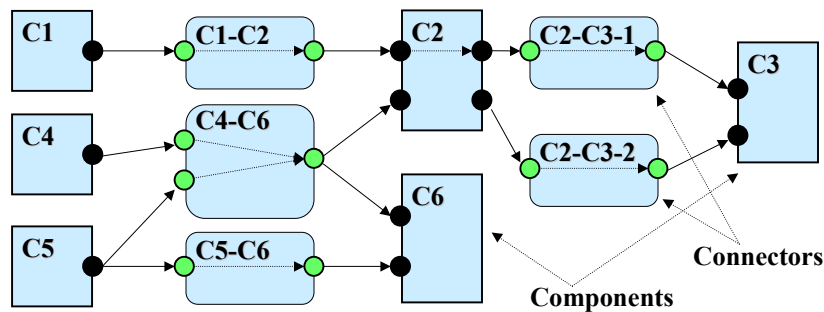
[ofut@ise.gmu.edu](mailto:ofut@ise.gmu.edu)

Joint research with: Zhenyi Jin, ITT Industries (PhD Dissertation)

*Supported by NSF and NIST.*

## Overview – Software Architecture

- High level of abstraction
- Components, connections, and configuration
- Focus on interactions
- Architecture Description Languages (ADLs)



## Motivation

- **Software Architecture Research**
  - System complexity increases
  - Better understanding and handling of larger systems
  - Reuse potential
- **Lack of Testing Techniques at Architecture Level**
  - Early faults
  - No formal definition of test criteria
  - No general techniques

(C) Copyright 2001, Offutt & Jin. All Rights Reserved.

3

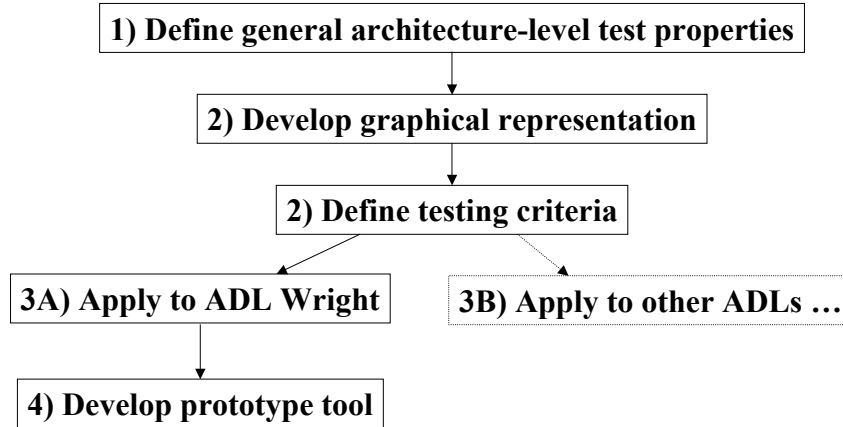
## Issues in Software Architecture-Based Testing

- **What general properties are important for testing at this level?**
- **Based on these general properties, what test requirements can be defined?**
- **Can general testing criteria be defined at this level?**
- **If so, what criteria?**

(C) Copyright 2001, Offutt & Jin. All Rights Reserved.

4

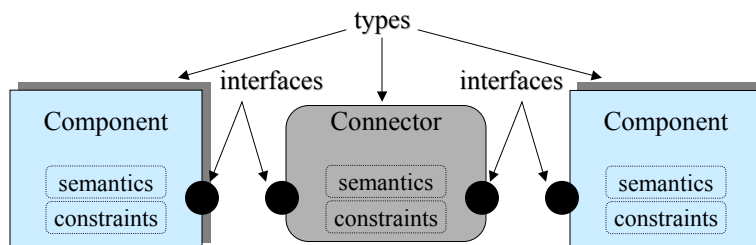
# Research Project Overview



(C) Copyright 2001, Offutt & Jin. All Rights Reserved.

5

# Testing Properties



(C) Copyright 2001, Offutt & Jin. All Rights Reserved.

6

## Testing Properties

- Component Internal Relation
  - *Component\_Internal\_Relation* ( $N_1.interf_1, N_1.interf_2$ )
- Component Internal Transfer Relation
  - *Component\_Internal\_Transfer\_Relation* ( $N.interf_1, N.interf_2$ )
- Component Internal Ordering Relation
  - *Component\_Internal\_Ordering\_Relation* ( $N.interf_1, N.interf_2$ )
- Connector Internal Relation
  - *Connector\_Internal\_Relation* ( $C.interf_1, C.interf_2$ )
- Connector Internal Transfer Relation
  - *Connector\_Internal\_Transfer\_Relation* ( $C.interf_1, C.interf_2$ )

(C) Copyright 2001, Offutt & Jin. All Rights Reserved.

7

## Testing Properties (2)

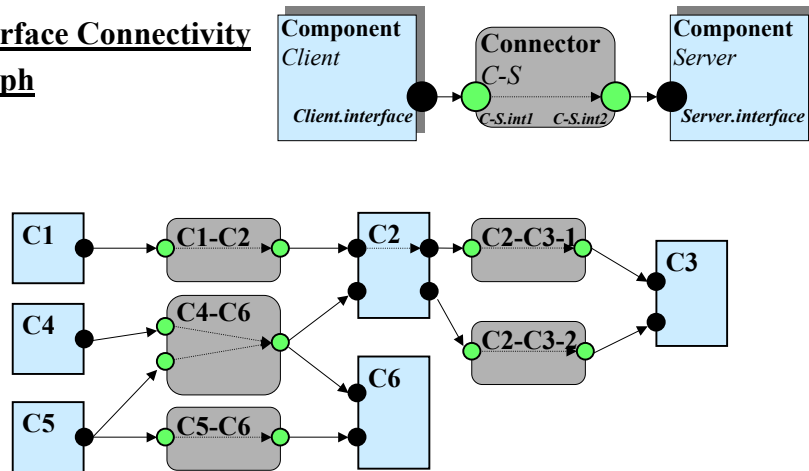
- Connector Internal Ordering Relation
  - *Connector\_Internal\_Ordering\_Relation* ( $C.interf_1, C.interf_2$ )
- Component and Connector Relation
  - *N\_C\_Relation* ( $N.interf_1, C.interf_1$ )
- Connector and Component Relation
  - *C\_N\_Relation* ( $C.interf_1, N.interf_1$ )
- Direct Component Relation
  - *Direct\_Component\_Relation* ( $N_1.interf_1, C_1.interf_1, C_1.interf_2, N_2.interf_1$ )
- Indirect Component Relation
  - *Indirect\_Component\_Relation* ( $N_1.interf_1, C_1.interf_1, C_1.interf_2,$   
 $N_2.interf_2, C_2.interf_1, C_2.interf_2, N_3.interf_1$ )

(C) Copyright 2001, Offutt & Jin. All Rights Reserved.

8

# Graphical Representation

## Interface Connectivity Graph



(C) Copyright 2001, Offutt & Jin. All Rights Reserved.

9

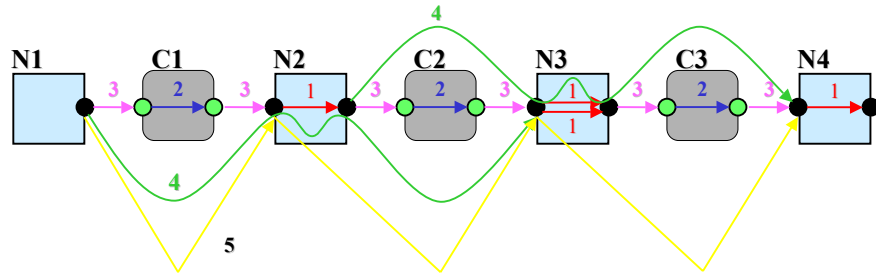
## Testing Criteria

1. Individual component interface coverage
2. Individual connector interface coverage
3. All direct component-to-component coverage
4. All indirect component-to-component coverage
5. All connected components coverage

(C) Copyright 2001, Offutt & Jin. All Rights Reserved.

10

## Testing Criteria Levels



Increasing levels of abstraction of concern

(C) Copyright 2001, Offutt & Jin. All Rights Reserved.

11

## Application to ADL Wright

Component Client

Port Service = ClientPullT

Computation = Service.open ; UseOrExit

where UseOrExit = UserService  $\square$  Exit

UserService = Service.request  $\rightarrow$  Service.result?y  $\rightarrow$  UseOrExit

Exit = Service.close  $\rightarrow$   $\$$

Component Server

Port Provide = ServerPushT

Computation = WaitForClient  $\square$  Exit  $\$$

where WaitForClient t =  $\square$  Provide.open  $\rightarrow$  Provide.receive-request  $\rightarrow$

Provide.result?y

$\rightarrow$  WaitForClient

Exit = Provide.close  $\rightarrow$   $\$$

Interface Type ClientPullT = open  $\rightarrow$  Operate  $\square$   $\$$

where Operate = request  $\rightarrow$  result?x  $\rightarrow$  Operate  $\square$  Close

Close = close  $\rightarrow$   $\$$

Interface Type ServerPushT = open  $\rightarrow$  Operate  $\square$   $\$$

where Operate = request  $\rightarrow$  resultx  $\rightarrow$  Operate  $\square$  Close

Close = close  $\rightarrow$   $\$$

(C) Copyright 2001, Offutt & Jin. All Rights Reserved.

12

## Application to ADL Wright (2)

```
Connector C-S Connector
Role Client = ClientPullT
Role Server = ServerPushT
Glue = Client.open → Server.open → Glue □ Client.close → Server.close → Glue
      □ Server.result?x → Client.result!x → Glue
      □ §
Instances
c: Client
s: Server
cs: C-S Connector
Attachments:
c provides as cs.c
s provides as cs.S
end
```

(C) Copyright 2001, Offutt & Jin. All Rights Reserved.

13

## Behavior Graph (BG)

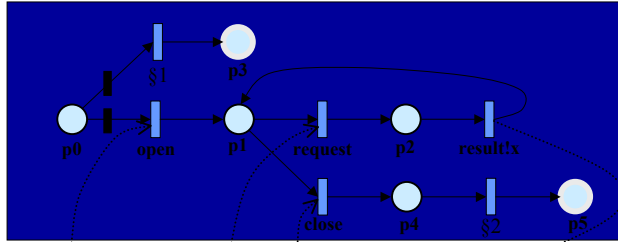
- The architectural description is modeled in a graphical representation called the behavior graph
- The BG uses a Petri net with extensions to model behavior
- It incorporates four kinds of paths:
  1. BG component behavior path (B-path)
  2. BG component connection path (C-path)
  3. BG interface interaction path (I-path)
  4. BG component indirect connection path (Indirect C-path)

(C) Copyright 2001, Offutt & Jin. All Rights Reserved.

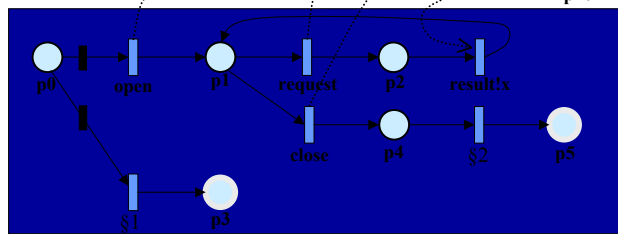
14

# Behavior Graph Example

Server Component



Client Component



(C) Copyright 2001, Offutt & Jin. All Rights Reserved.

15

# Mapping Wright to BG

Rule 1: Events  $e$  are translated as  $\text{Transf}[e]$

Rule 2: Events  $e?x$  are translated as  $\text{Transf}[e?x]$

Rule 3: Events  $e!x$  are translated as  $\text{Transf}[e!x]$

Rule 4: Process definitions  $P = e \rightarrow P$  are translated as  $\text{Transf}[P = e \rightarrow P]$

Rule 5: Event successful  $\S$  is translated as  $\text{Transf}[\S]$

Rule 6: Sequential Composition  $\circ$  :  $\text{Transf}[P_1 \rightarrow P_2] = \text{Transf}[P_1] \circ \text{Transf}[P_2]$

Rule 7: Non-deterministic (Internal Choice) Composition  $+$  :  $\text{Transf}[P_1 \square P_2] = \text{Transf}[P_1] + \text{Transf}[P_2]$

⋮

(C) Copyright 2001, Offutt & Jin. All Rights Reserved.

16

## Mapping Wright to BG (2)

Rule 8: Deterministic (External Choice) Composition  $\blacklozenge$ :  $\text{Transf}[P_1 \amalg P_2] = \text{Transf}[P_1] \blacklozenge \text{Transf}[P_2]$

Rule 9: Parallel Composition:  $\text{Transf}[P_1 \parallel P_2] = \text{Transf}[P_1] \cup \text{Transf}[P_2]$

Rule 10: Sequencing Composition ";":  $\text{Transf}[P_1 ; P_2] = \text{Transf}[P_1] \circ \text{Transf}[P_2]$

Rule 11: Naming Composition "where":  $\text{Transf}[f \rightarrow P \text{ where } P = P_1] = \text{Transf}[f] \circ$

Rule 12:  $\text{Transf}[\forall x: S \square P(x)] = \text{Transf}[P(x_1)] + \text{Transf}[P(x_2)] + \dots + \text{Transf}[P(x_n)]$

Rule 13:  $\text{Transf}[\forall x: S \amalg P(x)] = \text{Transf}[P(x_1)] \blacklozenge \text{Transf}[P(x_2)] \blacklozenge \dots \blacklozenge \text{Transf}[P(x_n)]$ :

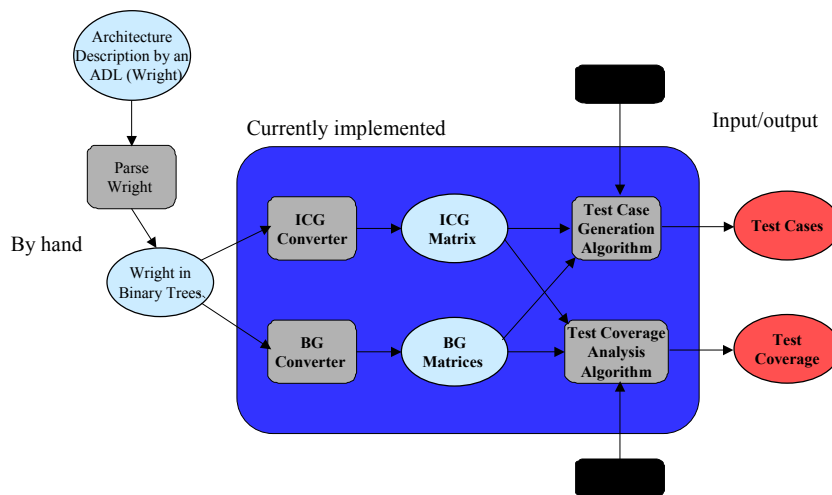
Rule 14:  $\text{Transf}[\forall x: S ; P(x)] = \text{Trans}[P(S)]$

Rule 15: State Variables

(C) Copyright 2001, Offutt & Jin. All Rights Reserved.

17

## Proof-of-Concept Tool



(C) Copyright 2001, Offutt & Jin. All Rights Reserved.

18

## Validation – Procedure

For each architecture description  $a$  and test adequacy criterion  $c$ :

1. Generate  $c$ -adequate test data set  $T(a, c)$
  2. Define fault set  $F(a)$  for  $a$
  3. For each  $f \in F(a)$  define the fault seeded architecture  $A(f)$  by seeding  $a$  with faults, yielding a fault-seeded architecture  $a(f)$  where each  $a(f) \in A(f)$
  4. For each  $t \in T(a, c)$ , if it detects some faults, increase  $\text{Num}(a, c)$  -- the number of faults detected by test data set  $T(a, c)$
  5. Determine the fault detection rate  $R(a, c)$ , for test adequacy criterion  $c$  with respect to architecture  $a$ , as:  $R(a, c) = \text{Num}(a, c) / |F(a, c)|$
- Determine the fault detection effectiveness  $E(a, c)$ , for test adequacy criterion  $c$  with respect to architecture  $a$ , as:  $E(a, c) = R(a, c) / |T(a, c)|$

(C) Copyright 2001, Offutt & Jin. All Rights Reserved.

19

## Validation – Subject Program

- **Industrial program furnished by first author's company**
- **Responsible for processing external messages**
- **8 software components (Java, Perl, C)**
- **3 data sources**
- **Connections through internet and LAN**

(C) Copyright 2001, Offutt & Jin. All Rights Reserved.

20

## Validation – Faults Seeded

- 16 types of faults
- Communication, messages, shared resources, ...
- Details in the paper

(C) Copyright 2001, Offutt & Jin. All Rights Reserved.

21

## Results: Faults Detected

	Architecture-based	Manual / Specification	Coupling-based
Number of test cases	24	21	14
Faults found	14	10	8
Faults not found	2	6	8
% faults found	87.5%	62.5%	50.0%
Test Effectiveness	59.3%	47.6%	57.1%

(C) Copyright 2001, Offutt & Jin. All Rights Reserved.

22

## Conclusions

- **A new general purpose software architecture-based testing technique**
- **Properties to be evaluated at the architectural level**
- **Formally defined general architecture-based testing criteria**
- **Testing criteria instantiated to a specific ADL, Wright**
- **Prototype tool and algorithm defined for the technique**
- **A Petri net based architecture modeling technique**

(C) Copyright 2001, Offutt & Jin. All Rights Reserved.

23

## Future Work

- **Application to other ADLs**
- **Mapping architecture names to implementation names**
- **Catalog of architecture mistakes**
  - Faults
  - Potential ramifications

**[www.ise.gmu.edu/faculty/ofut/](http://www.ise.gmu.edu/faculty/ofut/)**

(C) Copyright 2001, Offutt & Jin. All Rights Reserved.

24