CLEMSON
School of COMPUTING

Software testability measure for SAE Architecture Analysis and Design Language (AADL)

Hung Vo

Master thesis defense

July - 2012

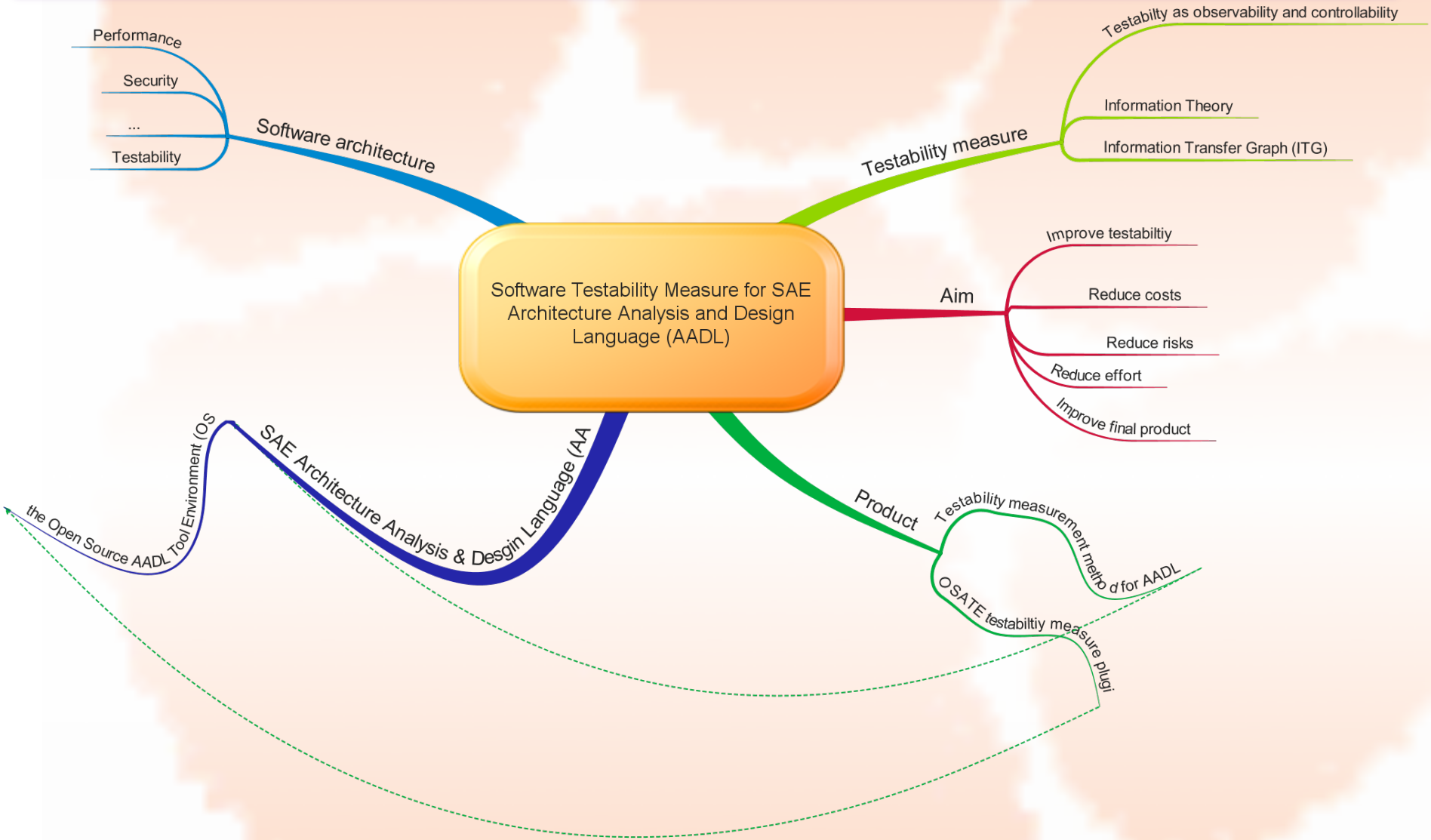
School of Computing, Clemson University

Contents

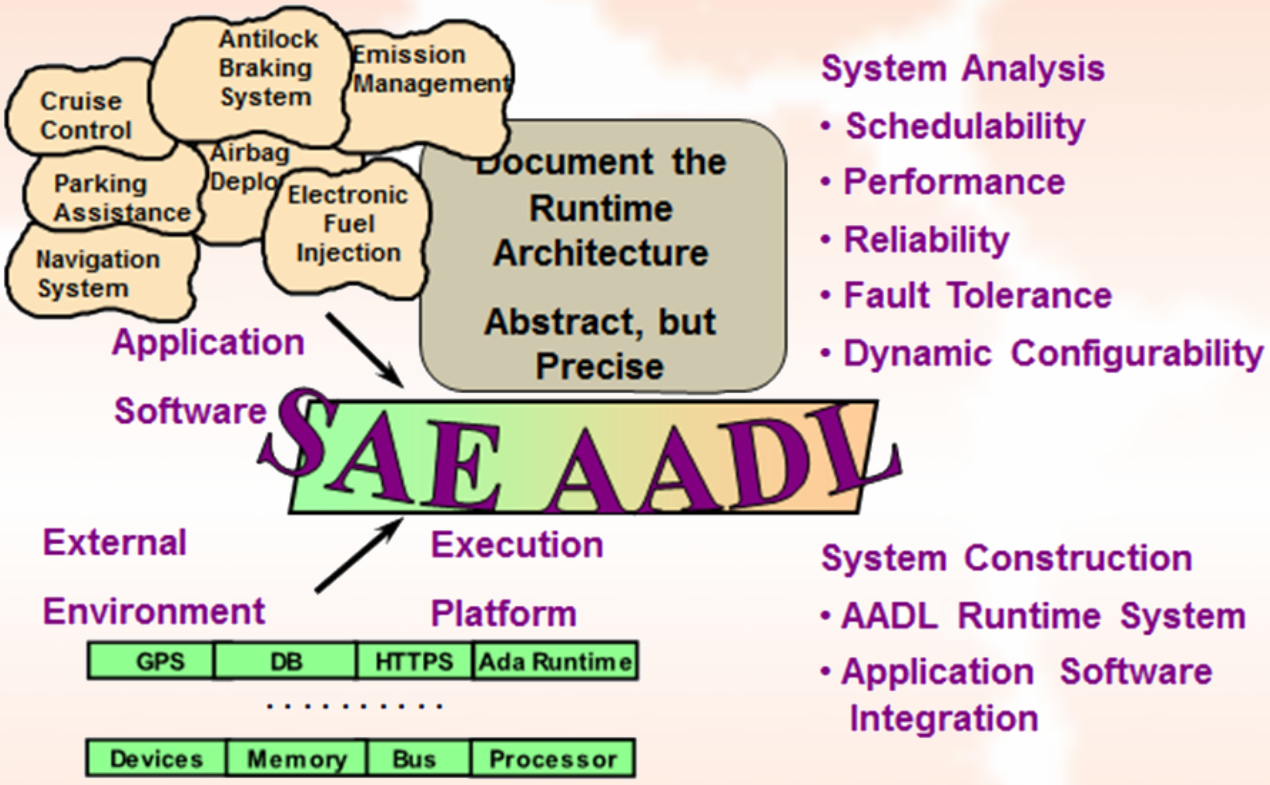
Acknowledgement

- I would like give special thanks to
  - My advisor: Dr. John Mcgregor
  - Dr. Murali Sitaraman
  - Dr. D.E. Stevenson
  - Dr. Mark Smotherman
  - The Vietnam Education Foundation
  - Ms. Barbara Ramirez
  - My friends in Clemson
  - My family

Introduction

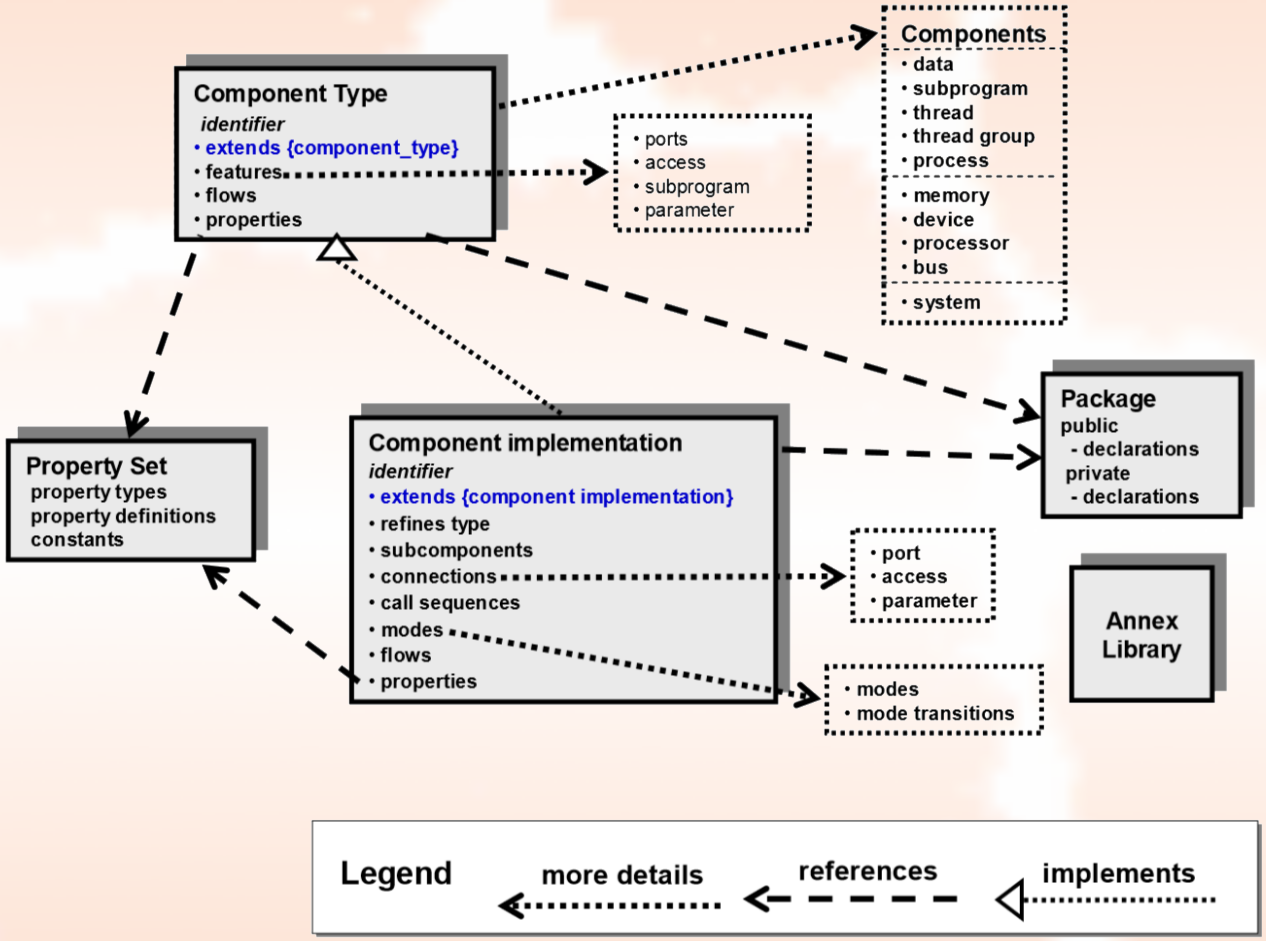


AADL

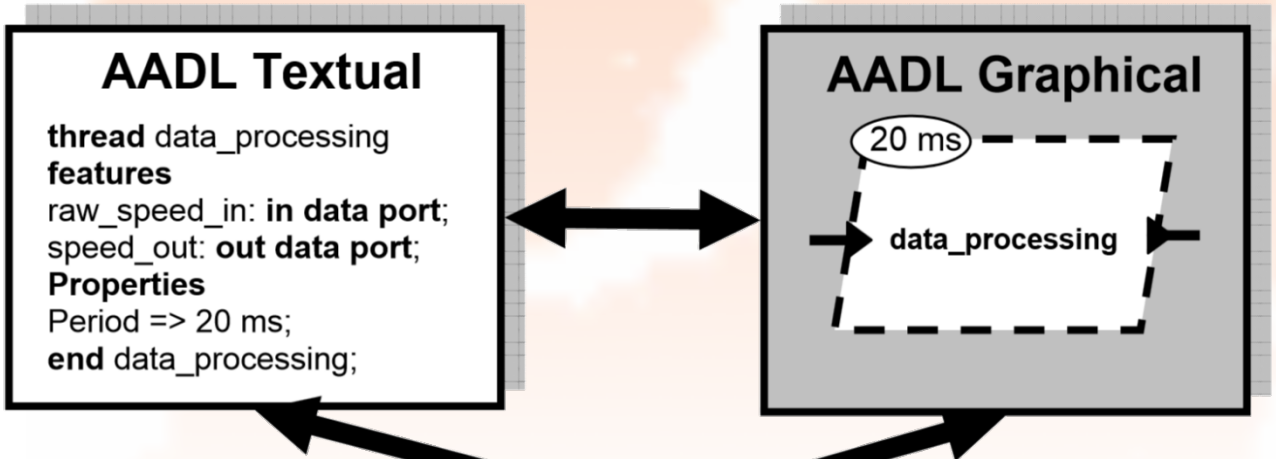


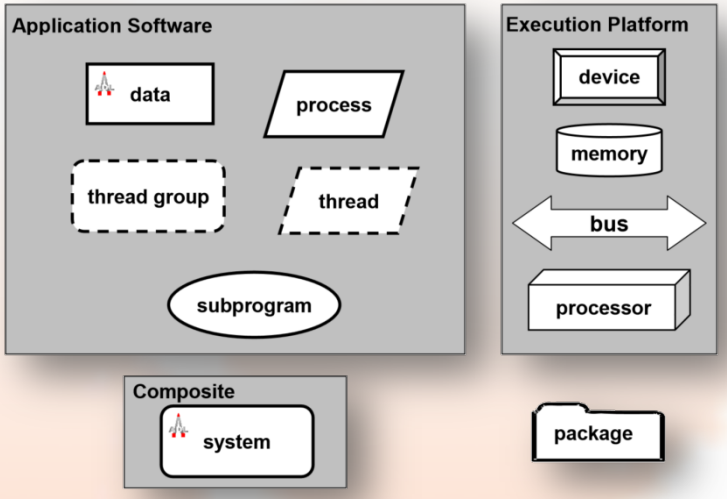
Model Based Development

AADL – basic elements

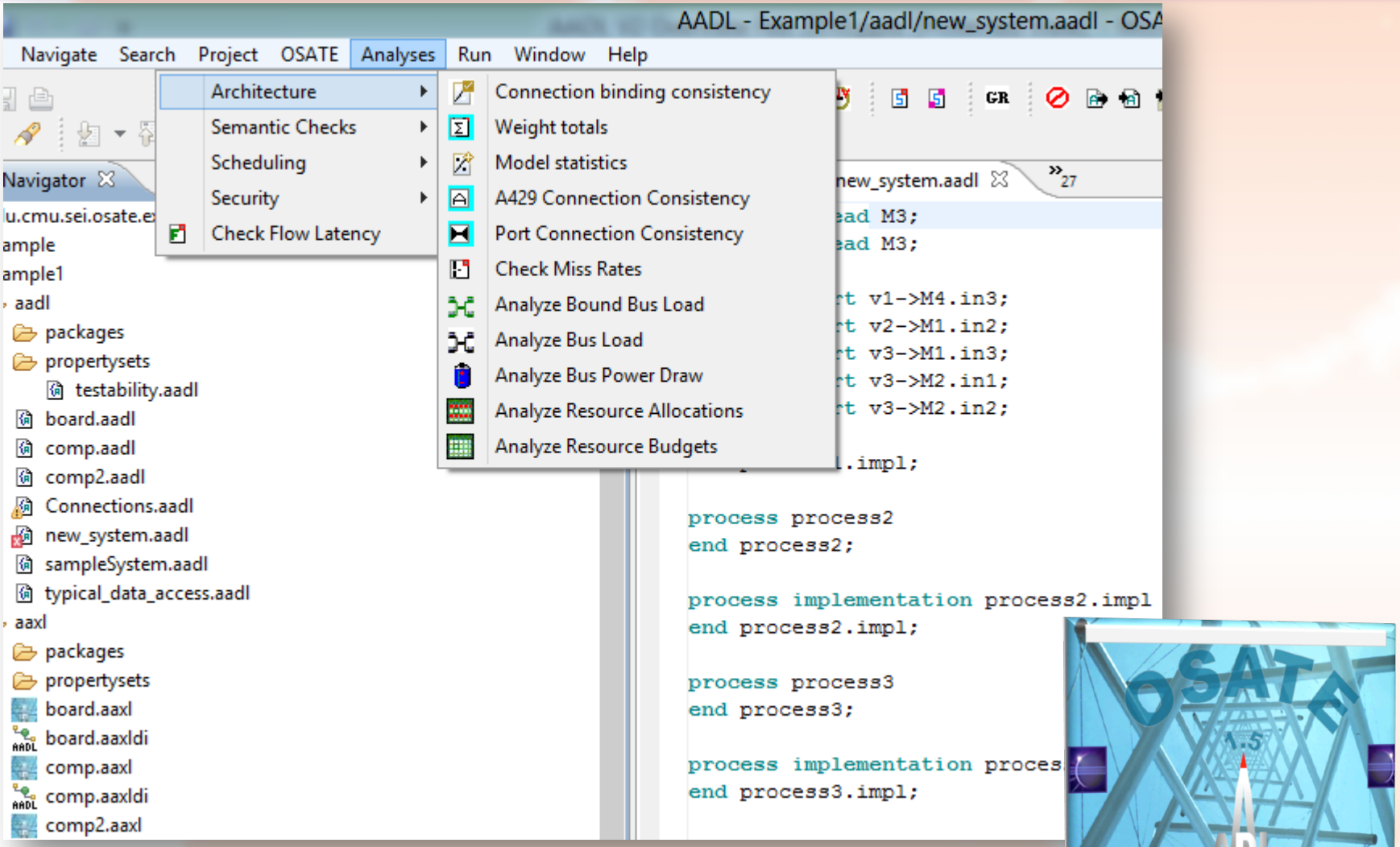


AADL - representation

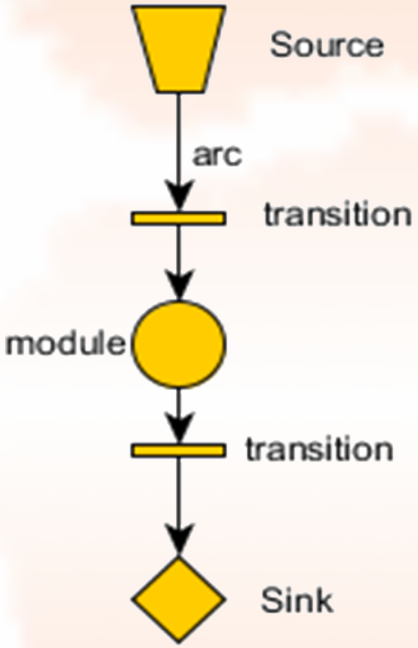




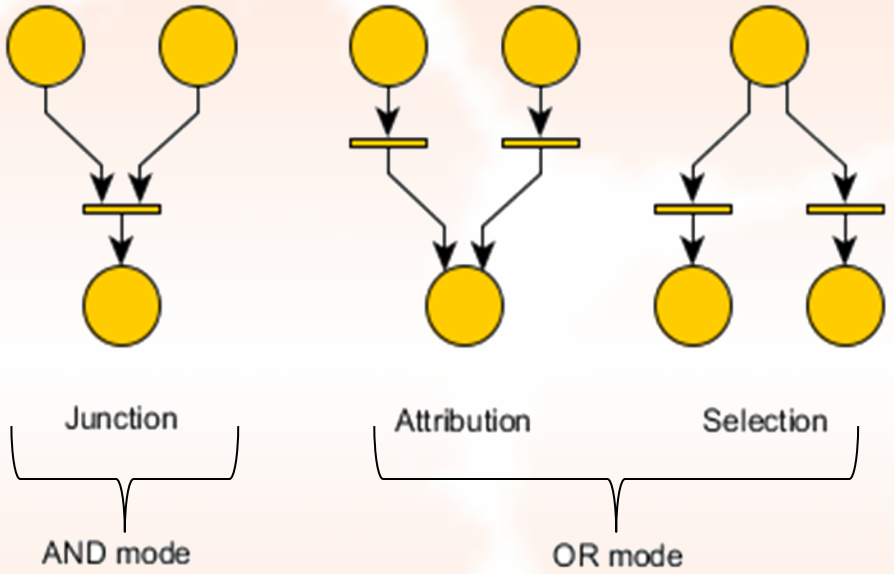
OSATE



Information transfer graph (ITG)

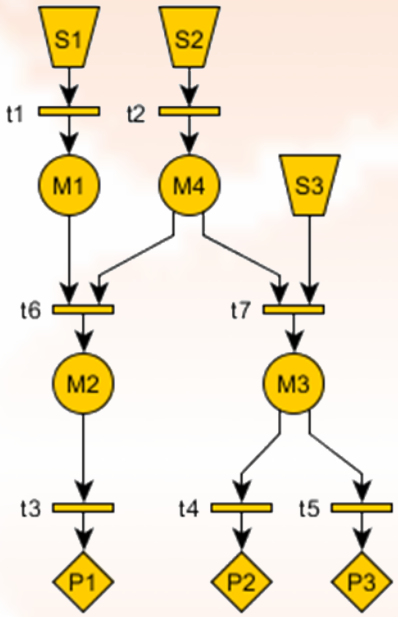


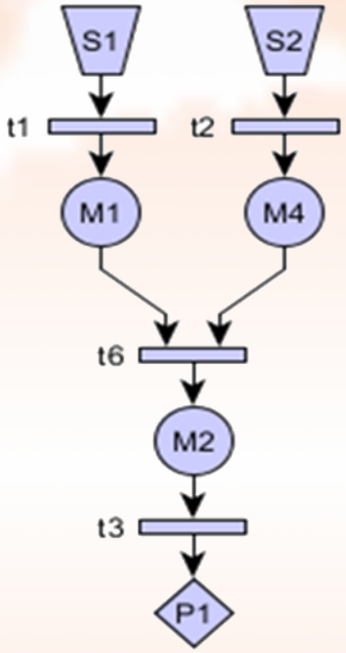
Simple ITG



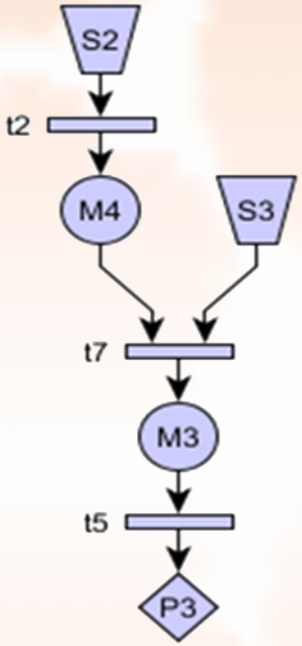
Transition modes

ITG - flows

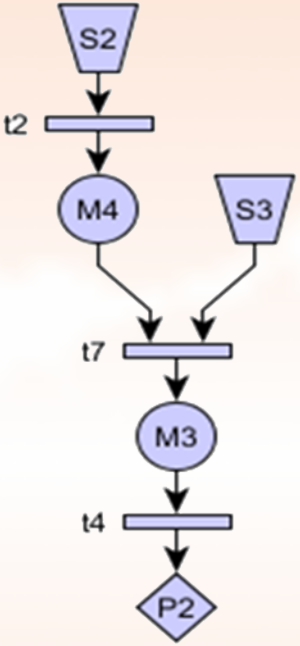




Flow 1

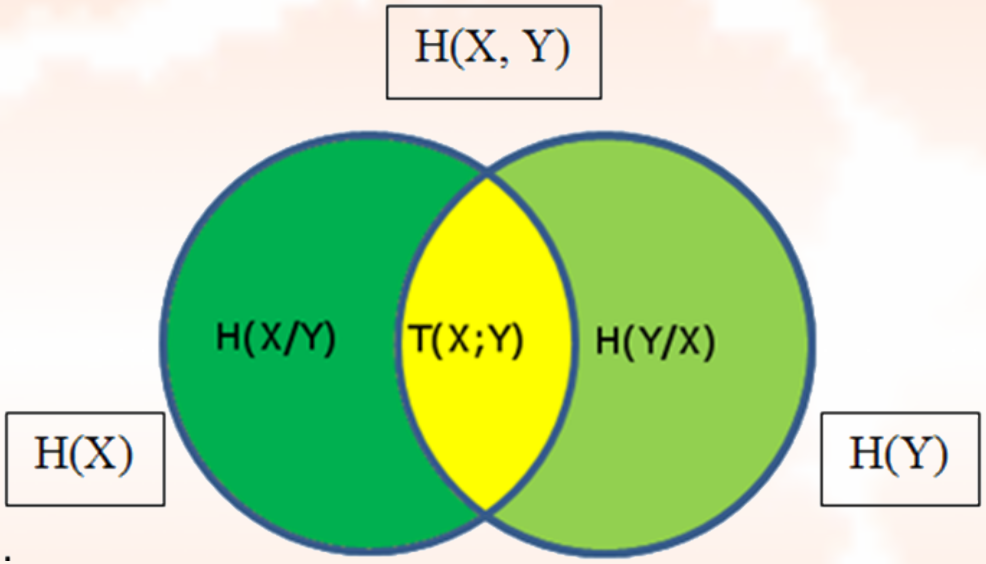


Flow 2



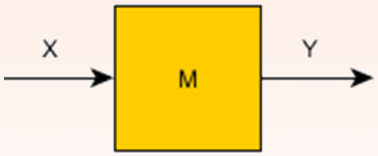
Flow 3

Information theory



Information channel

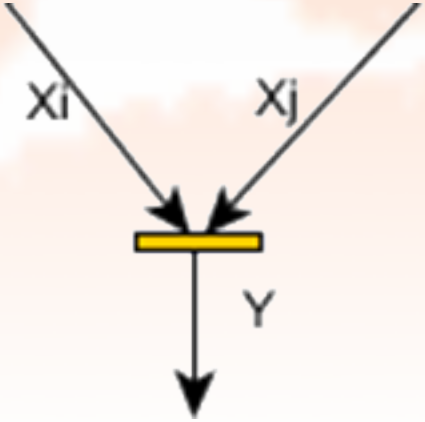
- Channel capacity C(X,Y)
  - is maximum of transinformation T(X,Y) over all possible dis tributions of X

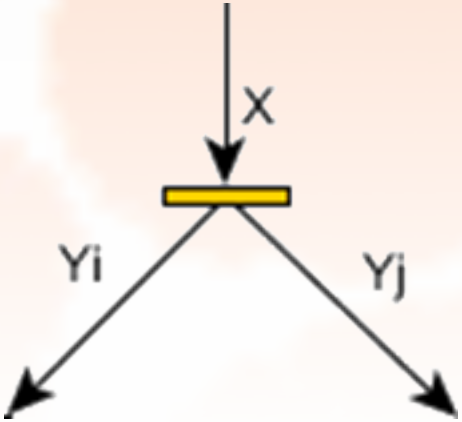


  - Module capacity C(M) = C(X,Y)

  - Line channel C(X), Y = X

Channel capacity – transition properties





$C(Y) = C(X)$

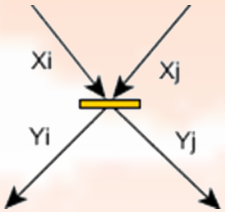
$C(Y) = \min [C(X_i,X_j), n_Y]$

$C(Y_i) = \min [C(X) , n_i]$

$C(Y_j) = \min [C(X), n_j]$

$C(Y_i,Y_j) = \min [C(X), n_i + n_j]$

Transinformation - properties



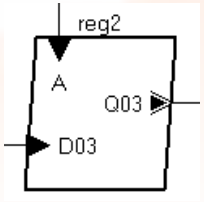
$$T(Z; Y{\downarrow}i, Y{\downarrow}j) \leq \min[T(Z; X{\downarrow}i, X{\downarrow}j), C(Y{\downarrow}i) + C(Y{\downarrow}j)]$$

To reduce complexity, upper bounded transinformation T' is introduced

$$T'(Z; Y{\downarrow}i, Y{\downarrow}j) = \min[T(Z; X{\downarrow}i, X{\downarrow}j), C(Y{\downarrow}i) + C(Y{\downarrow}j)]$$

T' can be solved using "min-cut, max-flow" algorithm.

Testability as controllability and observability

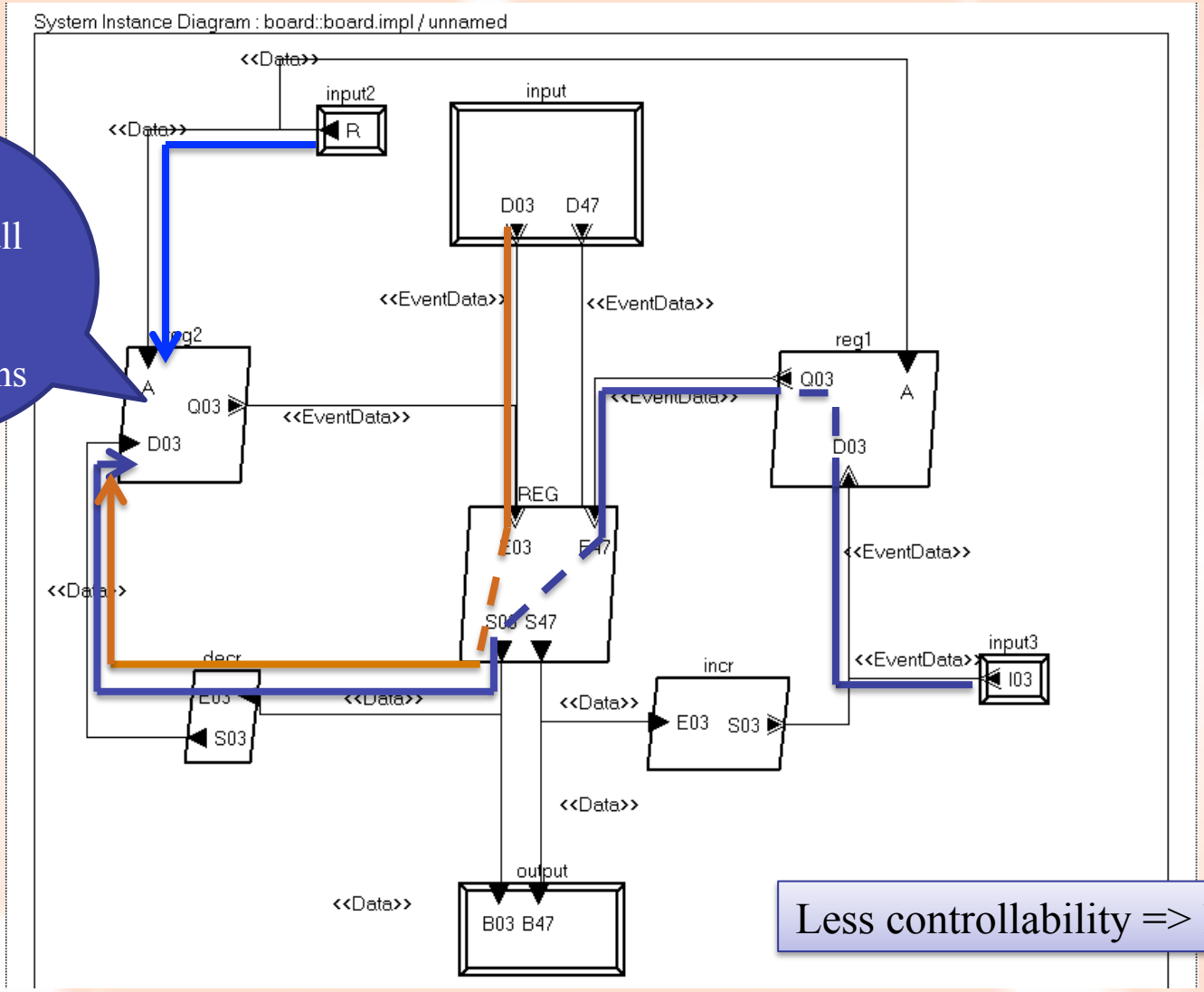


Total controllable

Total observable

Total testable

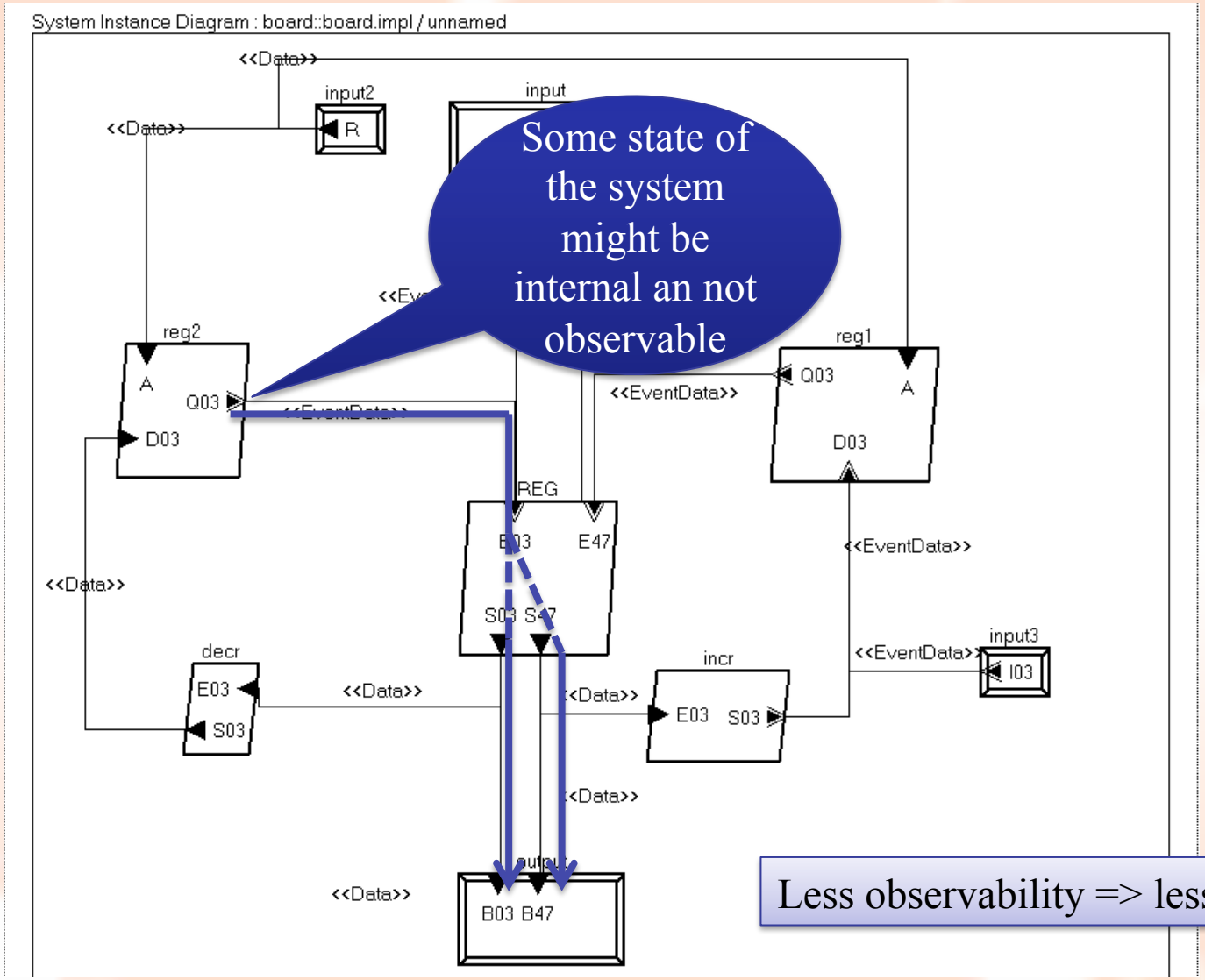
Testability as controllability and observability



Cannot guarantee all possible input combinations

Less controllability => less testability

Testability as controllability and observability



Less observability => less testability

Measure testability using ITG

Let F is any flow in the ITG, M is a module in flow F

X_M: inputs of module M

Y_M: outputs of module M

S_F: inputs of flow F

O_F: sinks of flow F

$$CO(M) = 2^{\uparrow} T(S_{\downarrow F}, X_{\downarrow M}) - C(X_{\downarrow M})$$

$$OB(M) = 2^{\uparrow} T(Y_{\downarrow M}, O_{\downarrow F}) - C(Y_{\downarrow M})$$

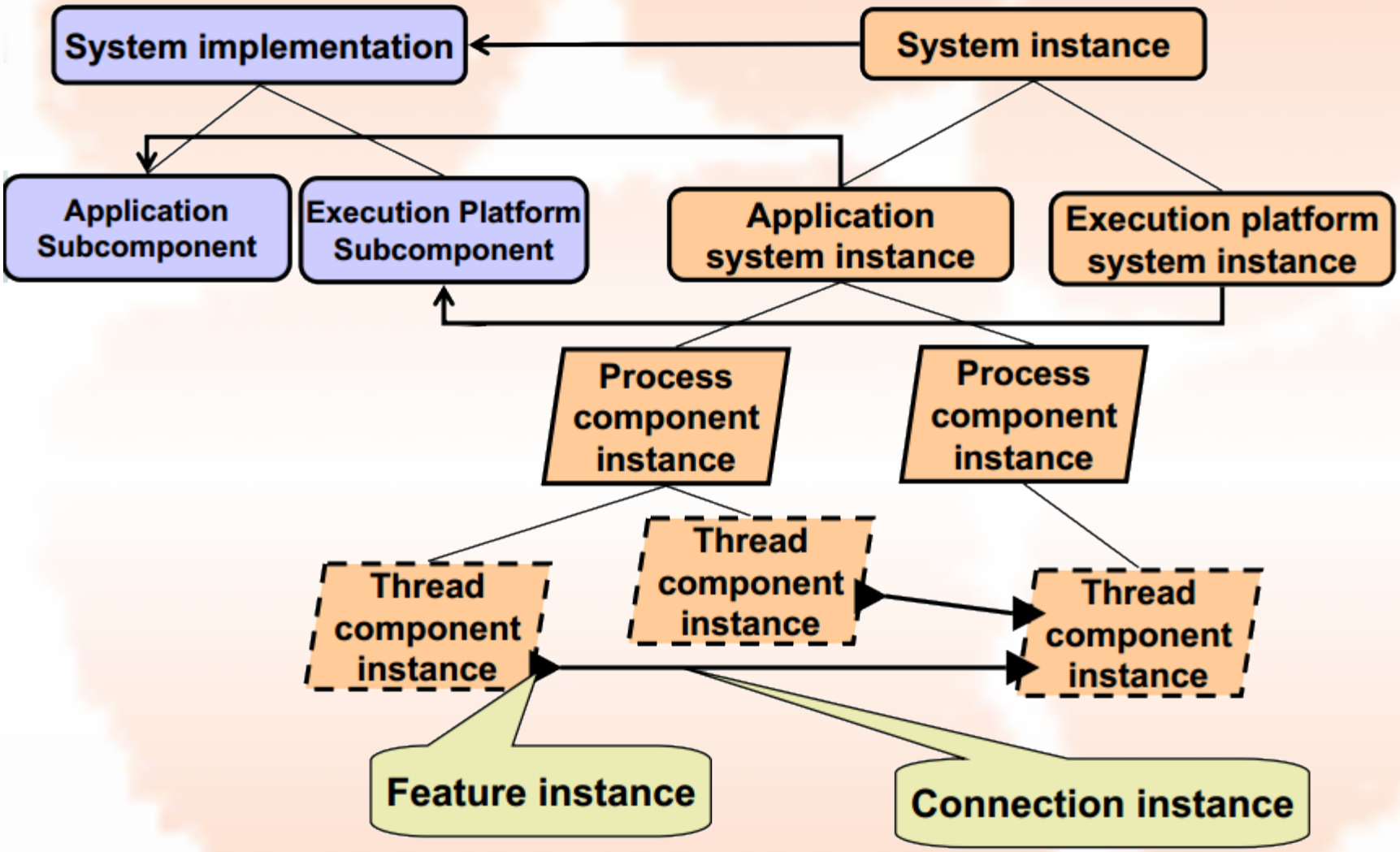
$$TE(M) = CO(M) \times OB(M)$$

$$0 \leq CO(M), OB(M), TE(M) \leq 1$$

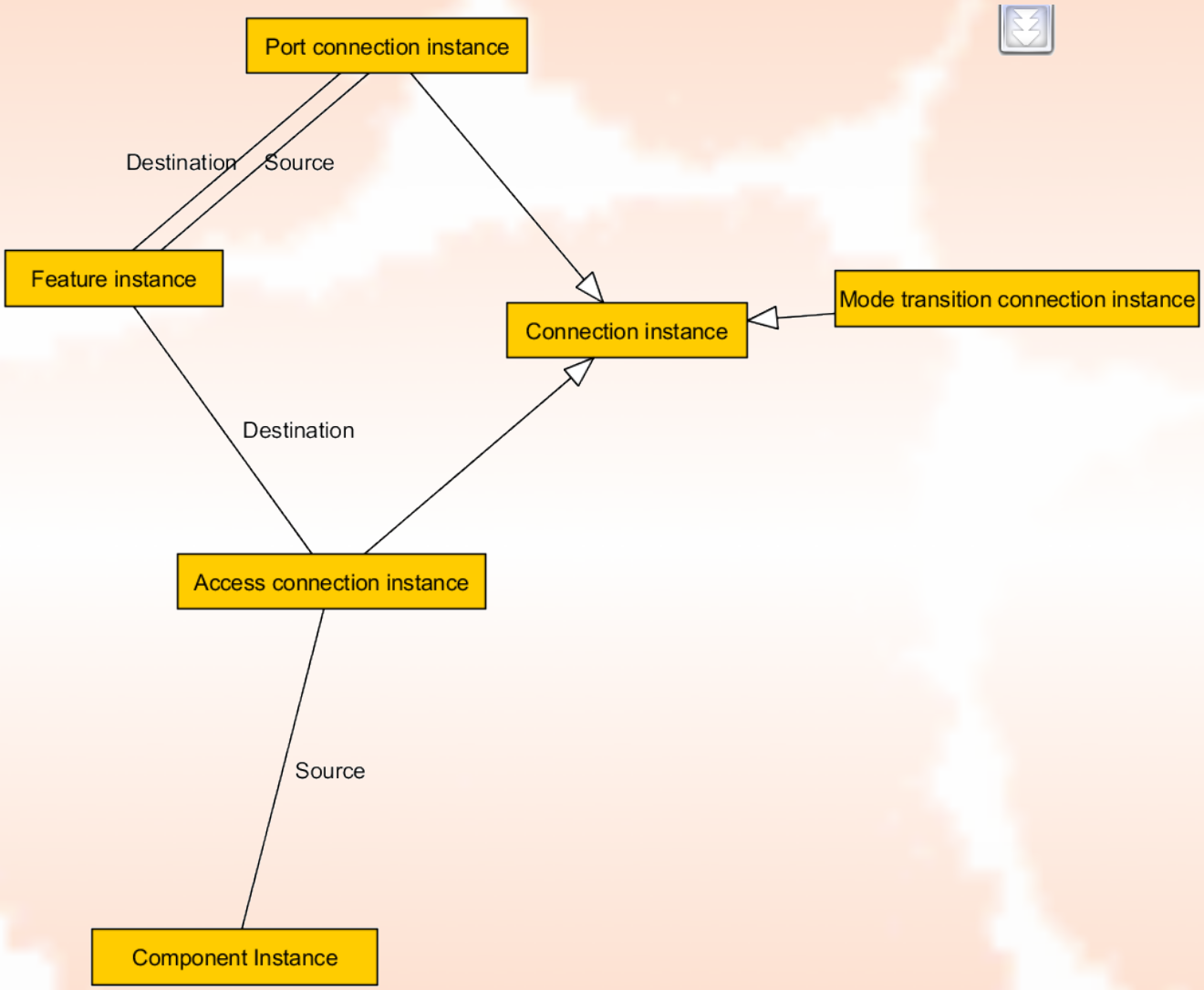
Theory Exposition

- AADL instance model
  - AADL instance connection
- Convert AADL instance model to ITG
- Calculate upper bounded testability
  - Calculate controllability
  - Calculate observability
  - Calculate channel capacity

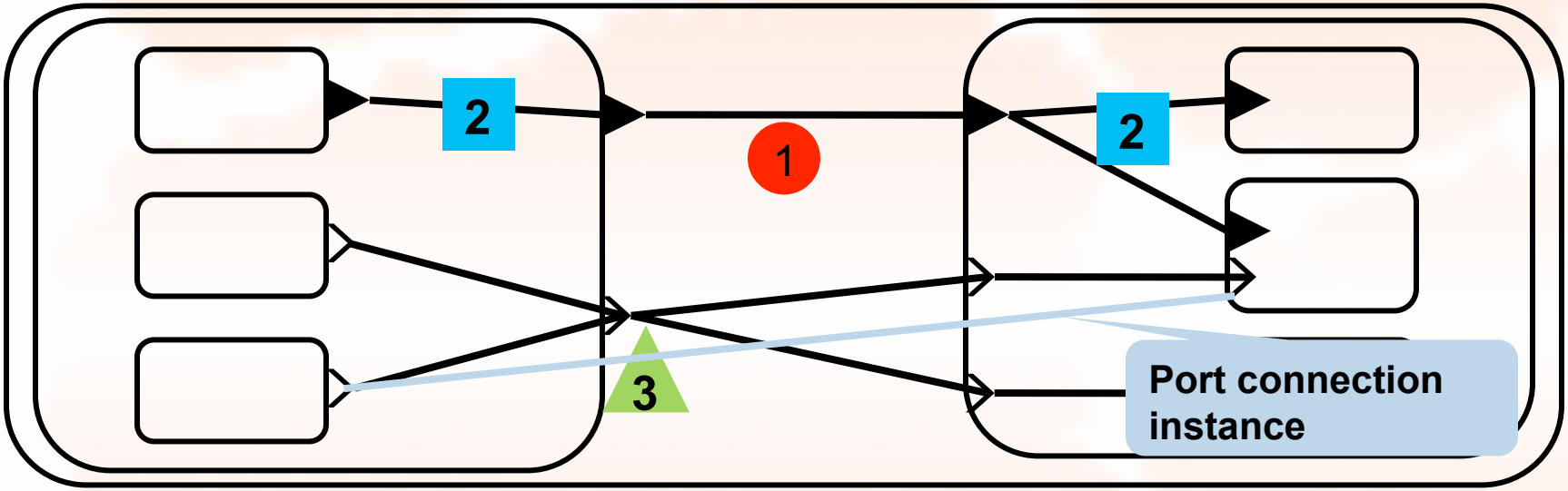
AADL instance model



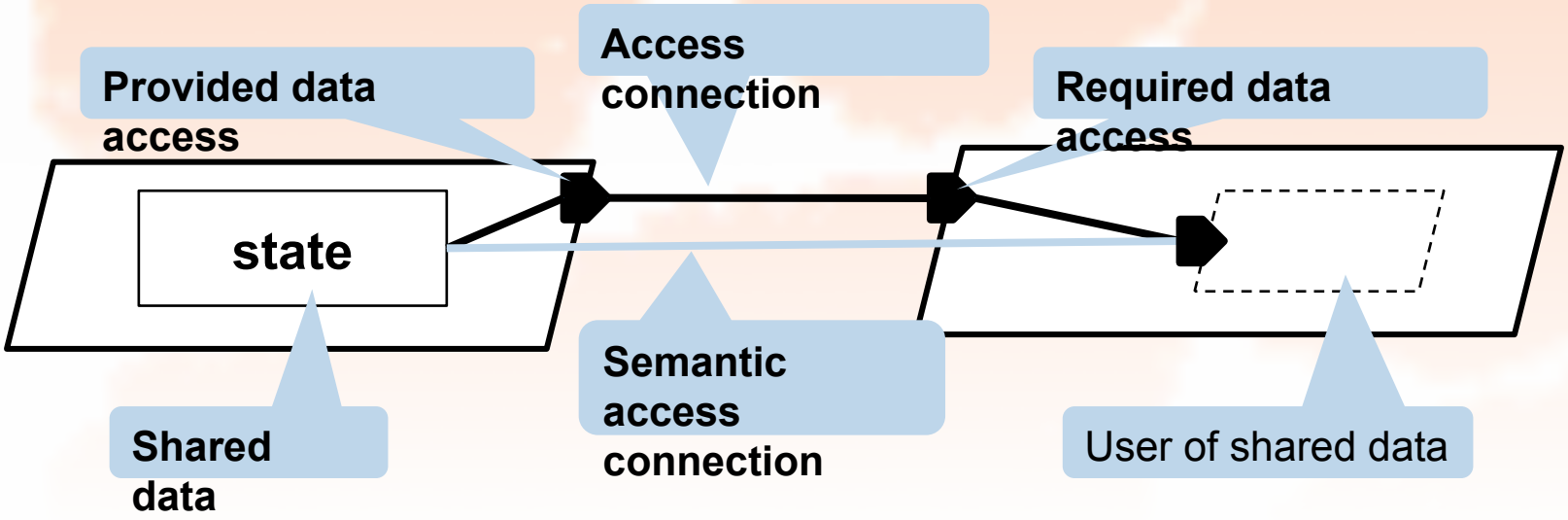
AADL instance model – Connection instance



AADL instance model – port connection instance



AADL instance model – data access connection

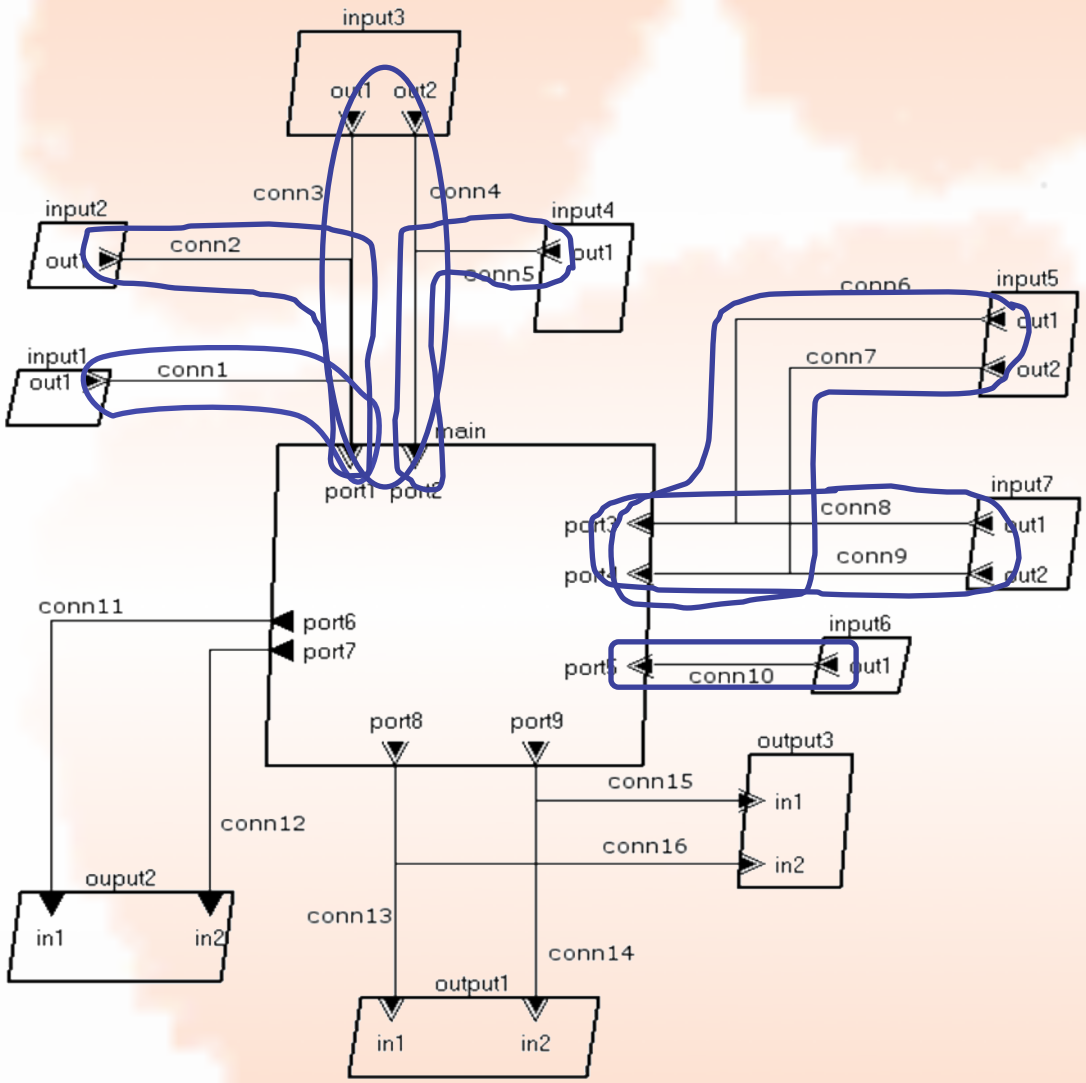



```
Provided_Access : access enumeration (read_only, write_only, read_write, by_method) => read_write
applies to (data, bus);
Required_Access : access enumeration (read_only, write_only, read_write, by_method) => read_write
applies to (data, bus);
```

Treat data access connection instance as port connection instance:

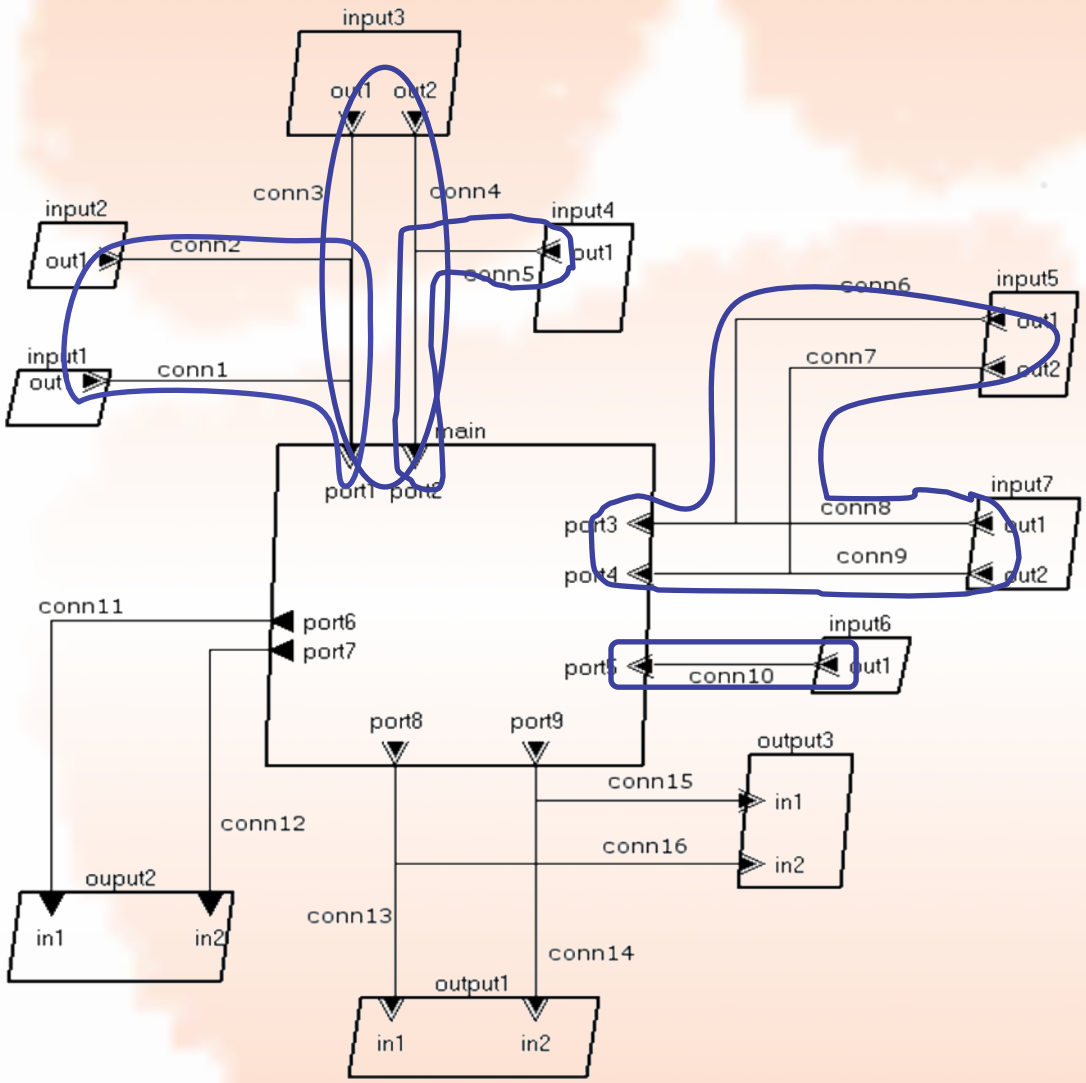
- Consider source component instance of a data access connection as feature instance of a port connection instance
- Direction of the connection depends on type of access (readable or writable)

Convert AADL instance model to ITG



- Input elements
- IE1 = {conn1}
- IE2 = {conn2}
- IE3 = {conn3, conn4}
- IE4 = {conn5}
- IE5 ={conn6,conn7}
- IE6 ={conn8, conn9}
- IE7 ={conn10}

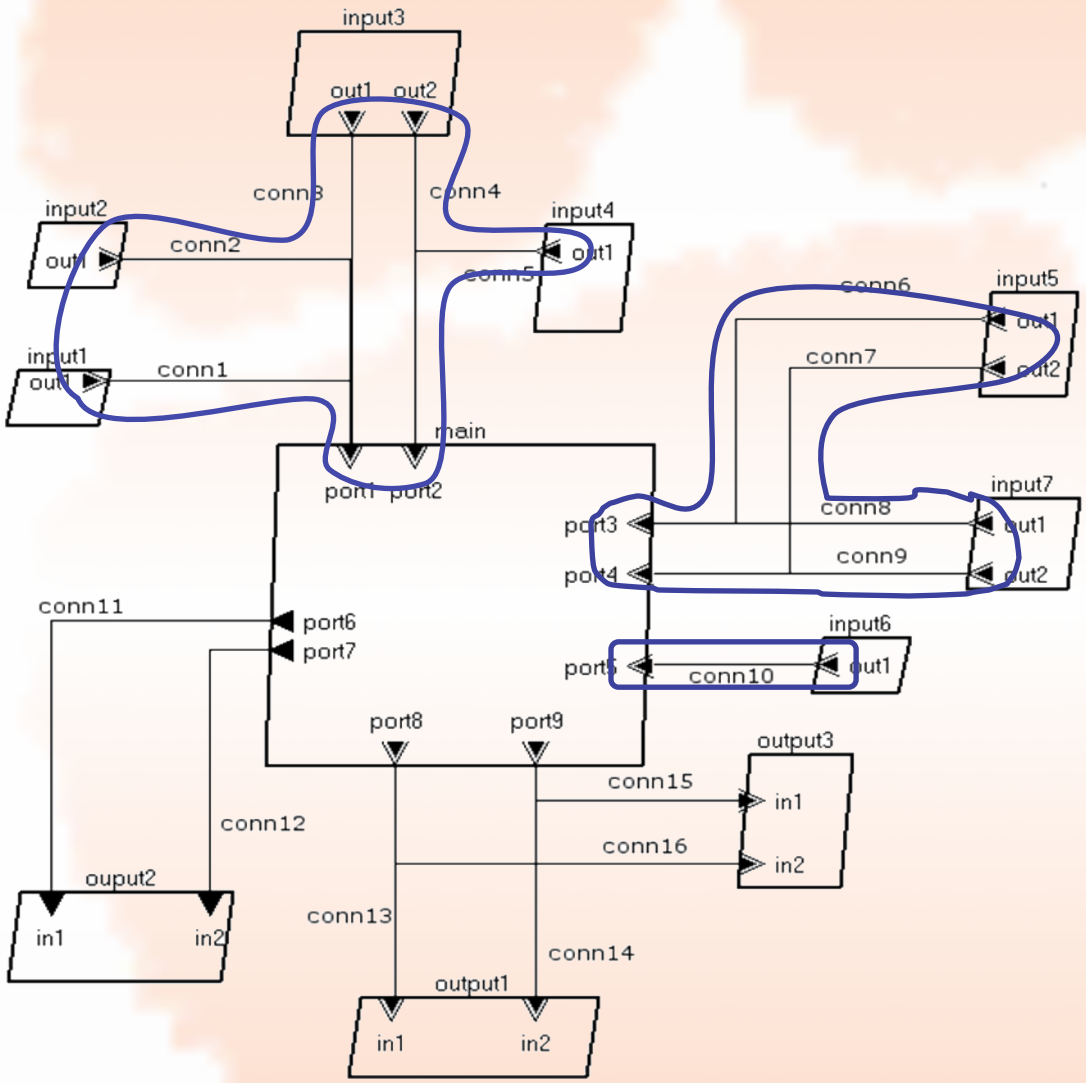
Convert AADL instance model to ITG



- Input elements
  - IE1 = {conn1}
  - IE2 = {conn2}
  - IE3 = {conn3, conn4}
  - IE4 = {conn5}
  - IE5 ={conn6,conn7}
  - IE6 ={conn8, conn9}
  - IE7 ={conn10}

- Input groups
  - IG1 = {IE1, IE2}
  - IG2 = {IE3}
  - IG3 = {IE4}
  - IG4 = {IE5, IE6}
  - IG5 = {IE7}

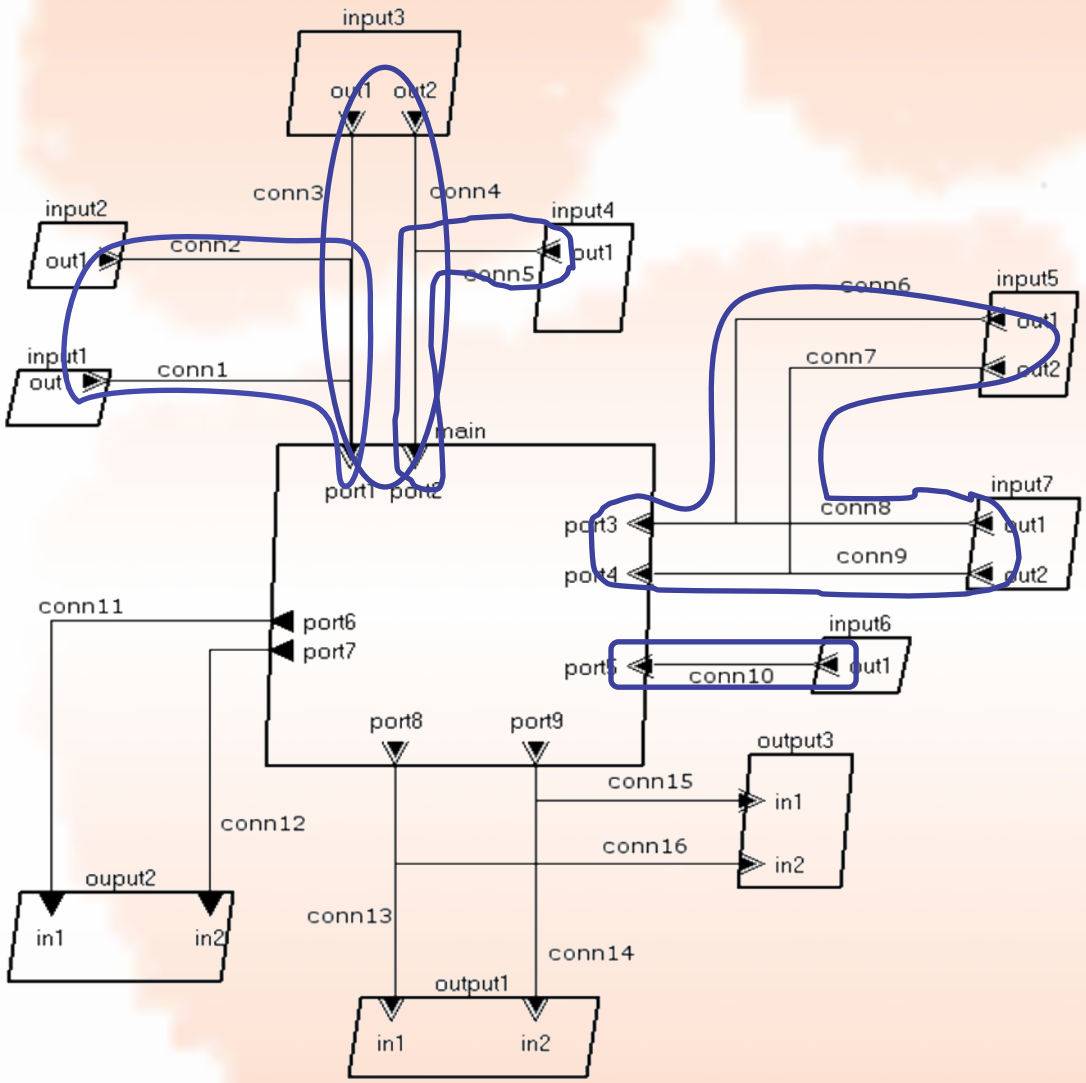
Convert AADL instance model to ITG



- Input groups
  - IG1 = {IE1, IE2}
  - IG2 = {IE3}
  - IG3 = {IE4}
  - IG4 = {IE5, IE6}
  - IG5 = {IE7}

- Input hypergroups
  - IHG1 = {IG1, IG2, IG3}
  - IHG2 = {IG4}
  - IHG3 = {IG5}

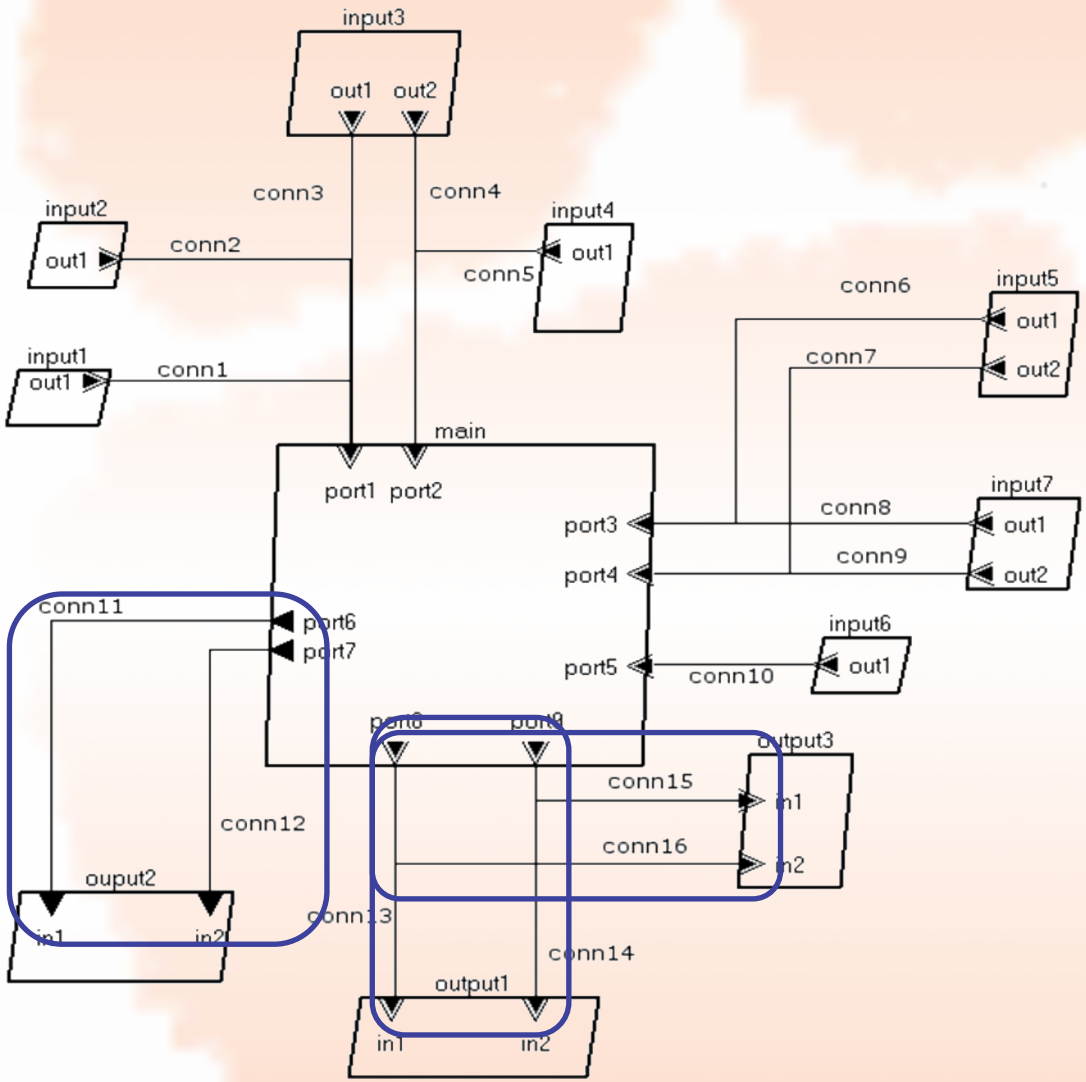
Convert AADL instance model to ITG



- Input hypergroups
- IHG1 = {IG1, IG2, IG3}
- IHG2 = {IG4}
- IHG3 = {IG5}

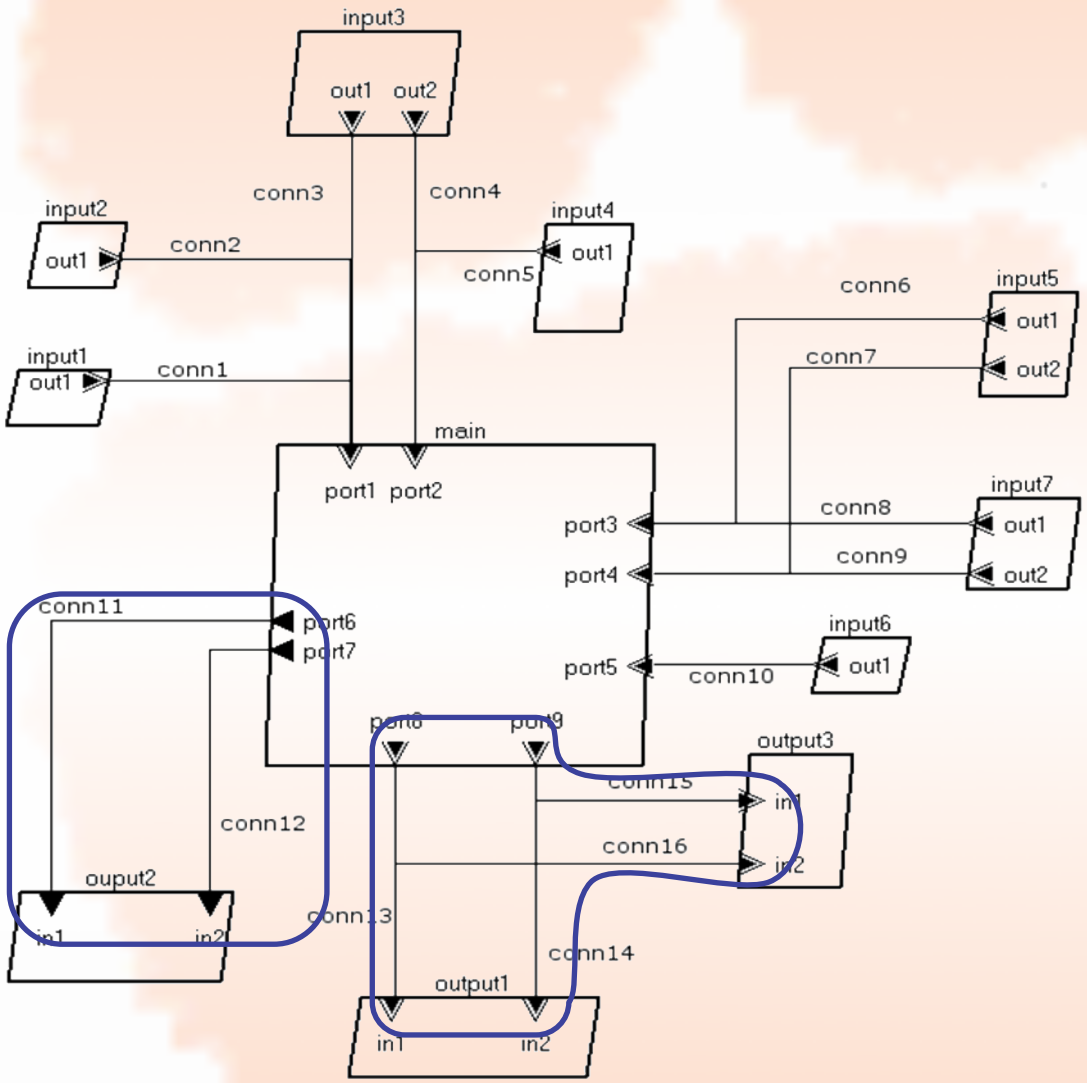
- Input supergroups
- IHG1:
  - ISG1 = {IG1, IG3}
  - ISG2 = {IG2}
- IHG2:
  - ISG3 = {IG4}
- IHG3:
  - ISG4 = {IG5}

Convert AADL instance model to ITG



- Output Elements
- OE1={conn11, conn12}
- OE2={conn13, conn14}
- OE3 = {conn15, conn16}

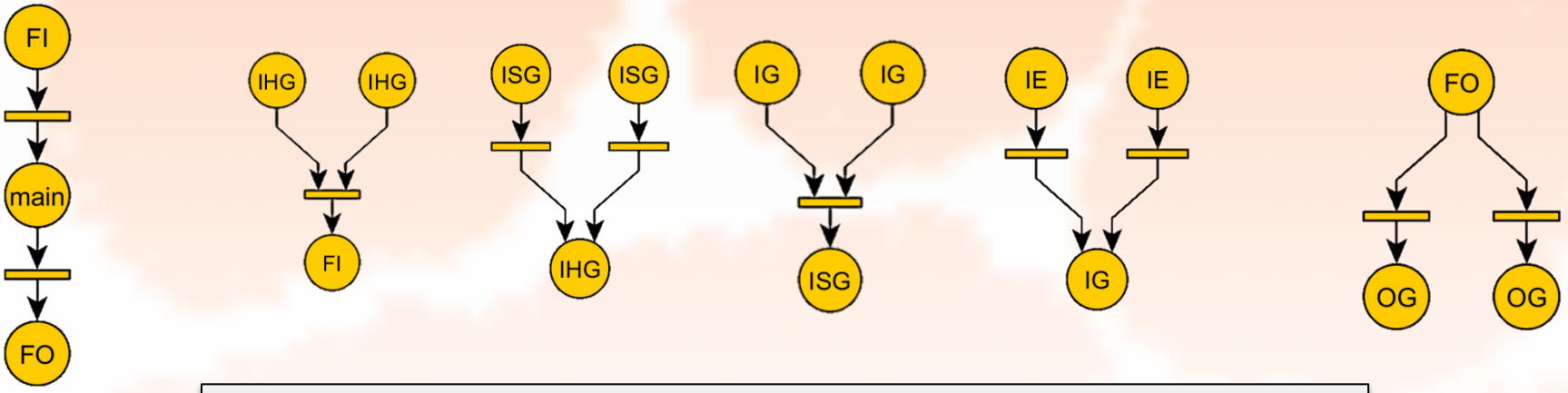
Convert AADL instance model to ITG



- Output Elements
  - OE1={conn11, conn12}
  - OE2={conn13, conn14}
  - OE3 = {conn15, conn16}

- Output groups
  - OG1 = {OE1}
  - OG2 = {OE2, OE3}

Convert AADL instance model to ITG



Functional input analysis

FI = IHG1 * IHG2 * IHG3

= (ISG1 + ISG2) * ISG3 * ISG4

= ((IG1 * IG3) + IG2) * IG4 * IG5

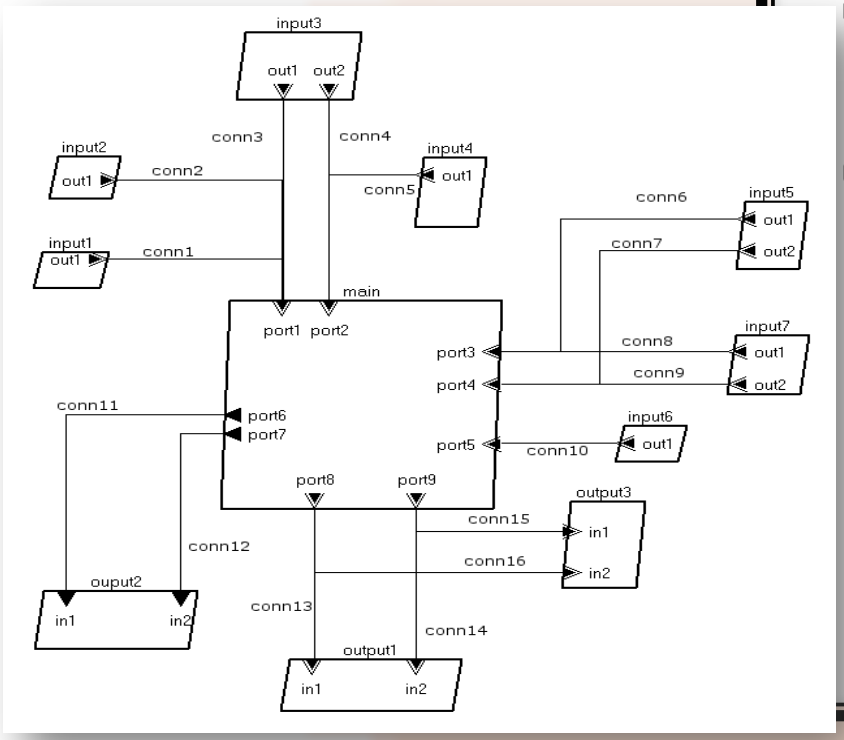
= ((IE1 + IE2) * IE4) + IE3) * (IE5 + IE6) * IE7

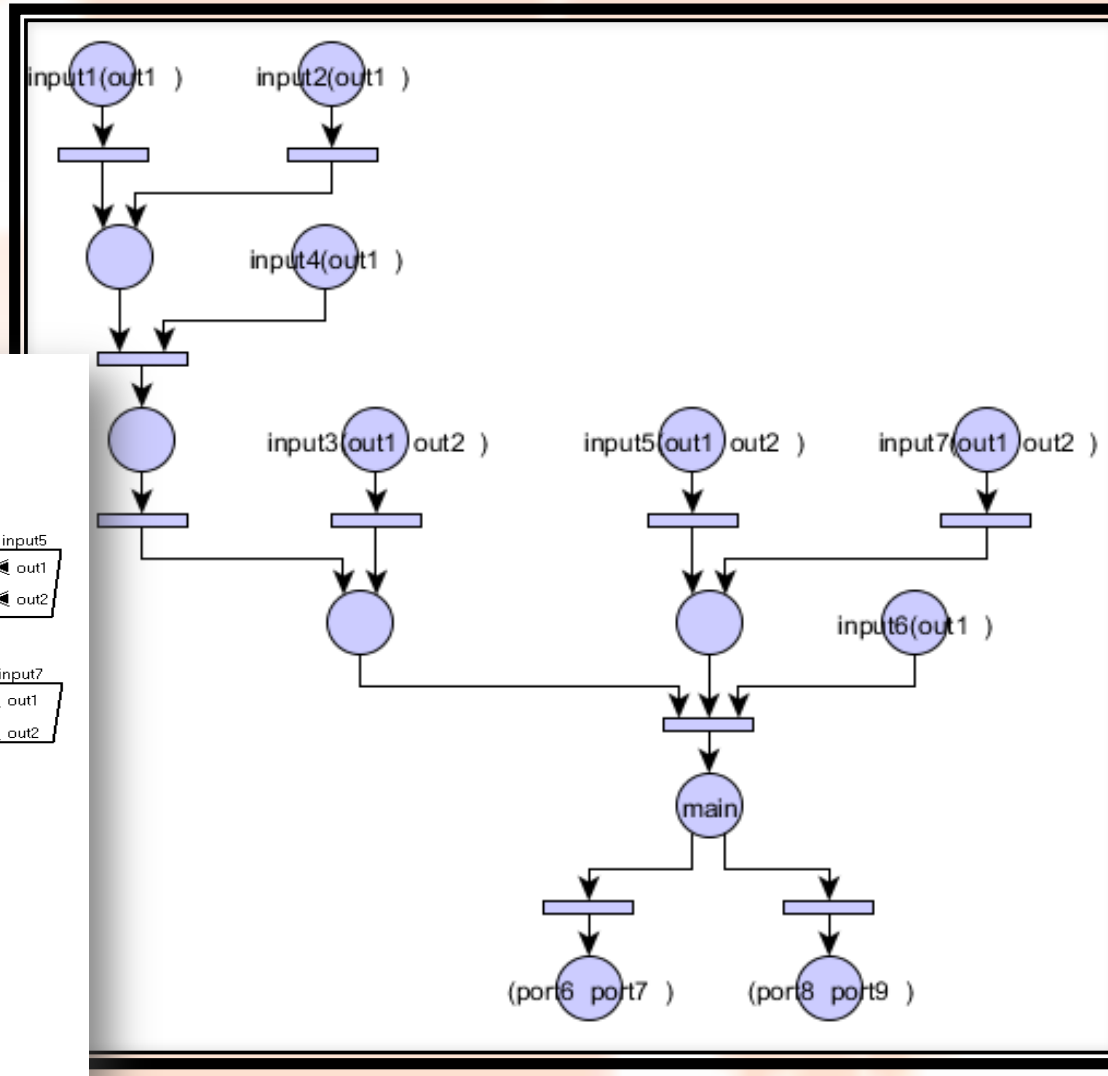
Functional output analysis

FO = OG1 + OG2

Convert AADL instance model to ITG

Final specific model for “main”





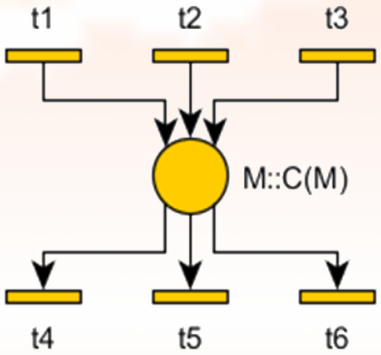
Calculate upper bounded testability

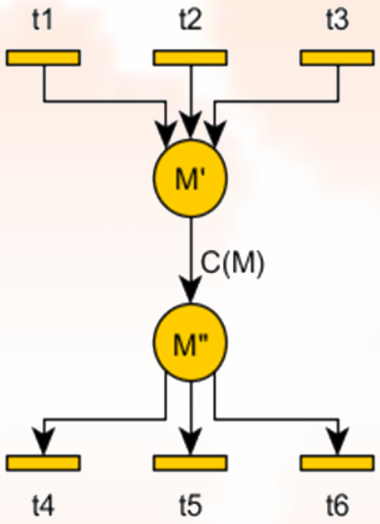
$$CO(M) = 2^{\uparrow} T'(S{\downarrow}F, X{\downarrow}M) - C(X{\downarrow}M)$$

$$OB(M) = 2^{\uparrow} T'(Y{\downarrow}M, O{\downarrow}F) - C(Y{\downarrow}M)$$

$$TE(M) = CO(M) \times OB(M)$$

Calculate upper bounded testability - transform





$$C(X \downarrow M) = C(X \downarrow M')$$

$$C(Y \downarrow M) = C(M)$$

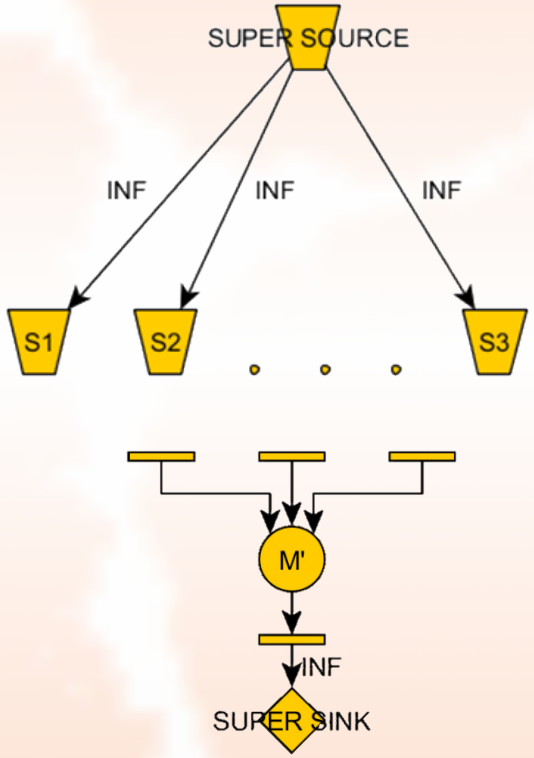
Calculate upper bounded testability

Calculate controllability

Add super source SS, connect SS to all sources

Add fiction edge u_r from M' to super sink SO

Find max flow from SS to SO



$T\uparrow(S\downarrow F, X\downarrow M) = T\uparrow(SS, X\downarrow M\uparrow) = \text{flow}($

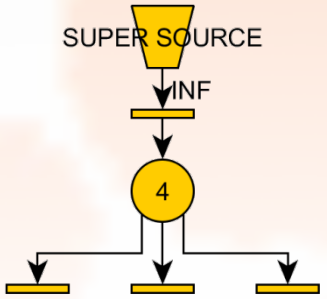
Calculate upper bounded testability

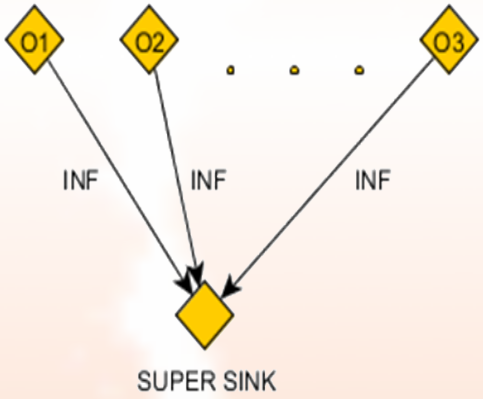
Calculate observability

Add fiction edge v_r from SS to M'

Connect all flow sinks to SO

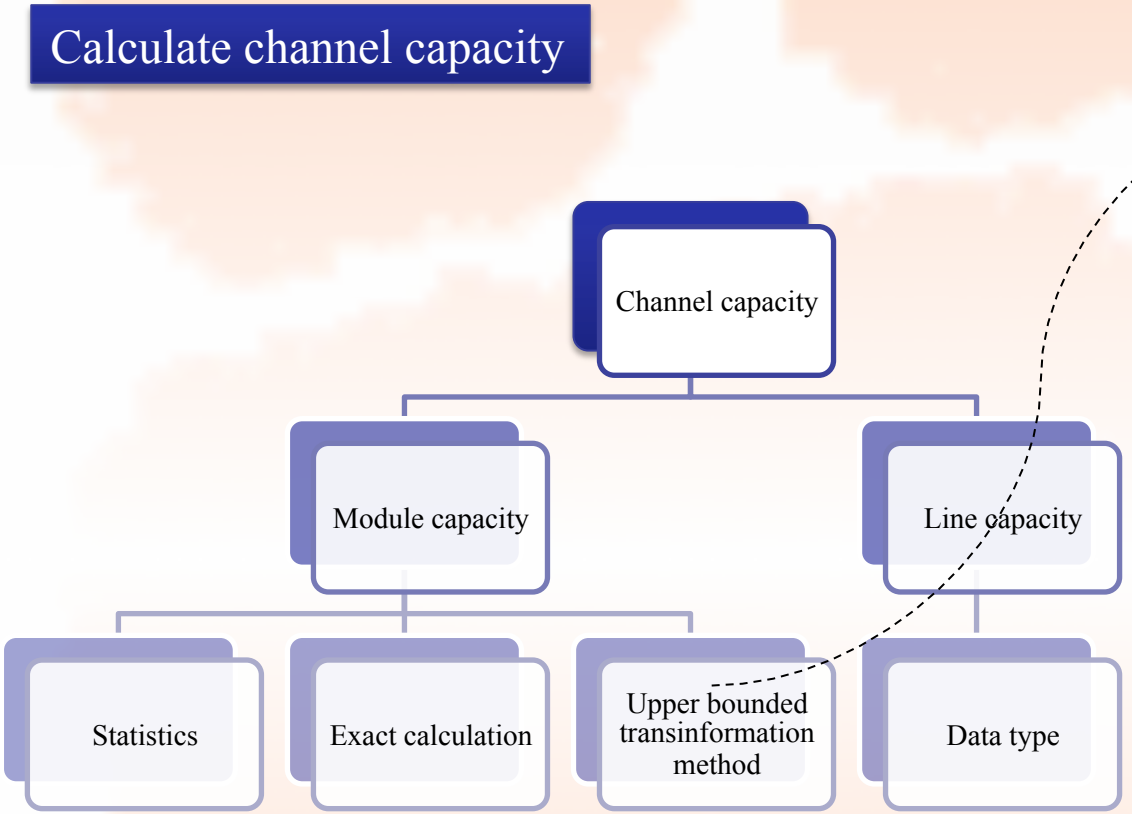
Find max flow from SS to SO

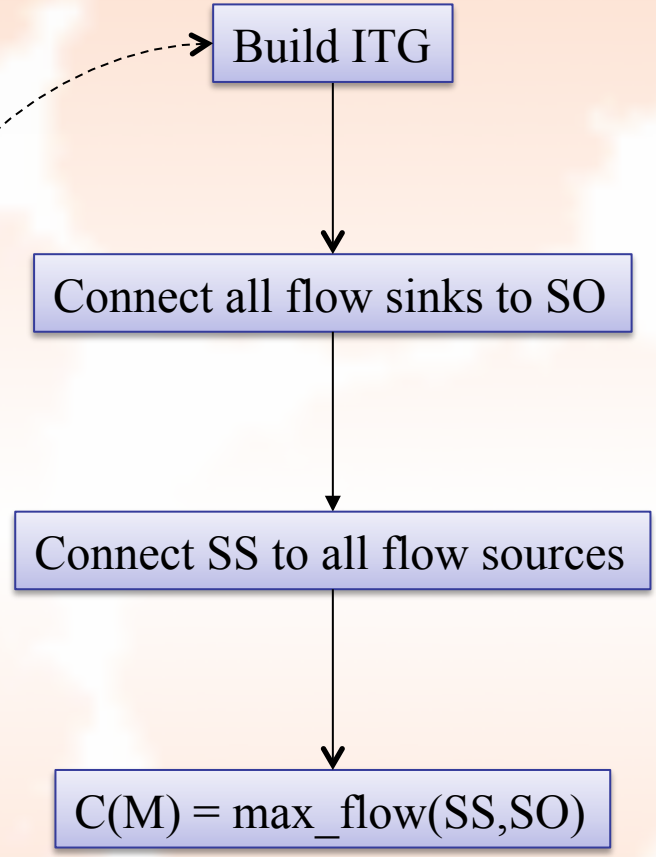




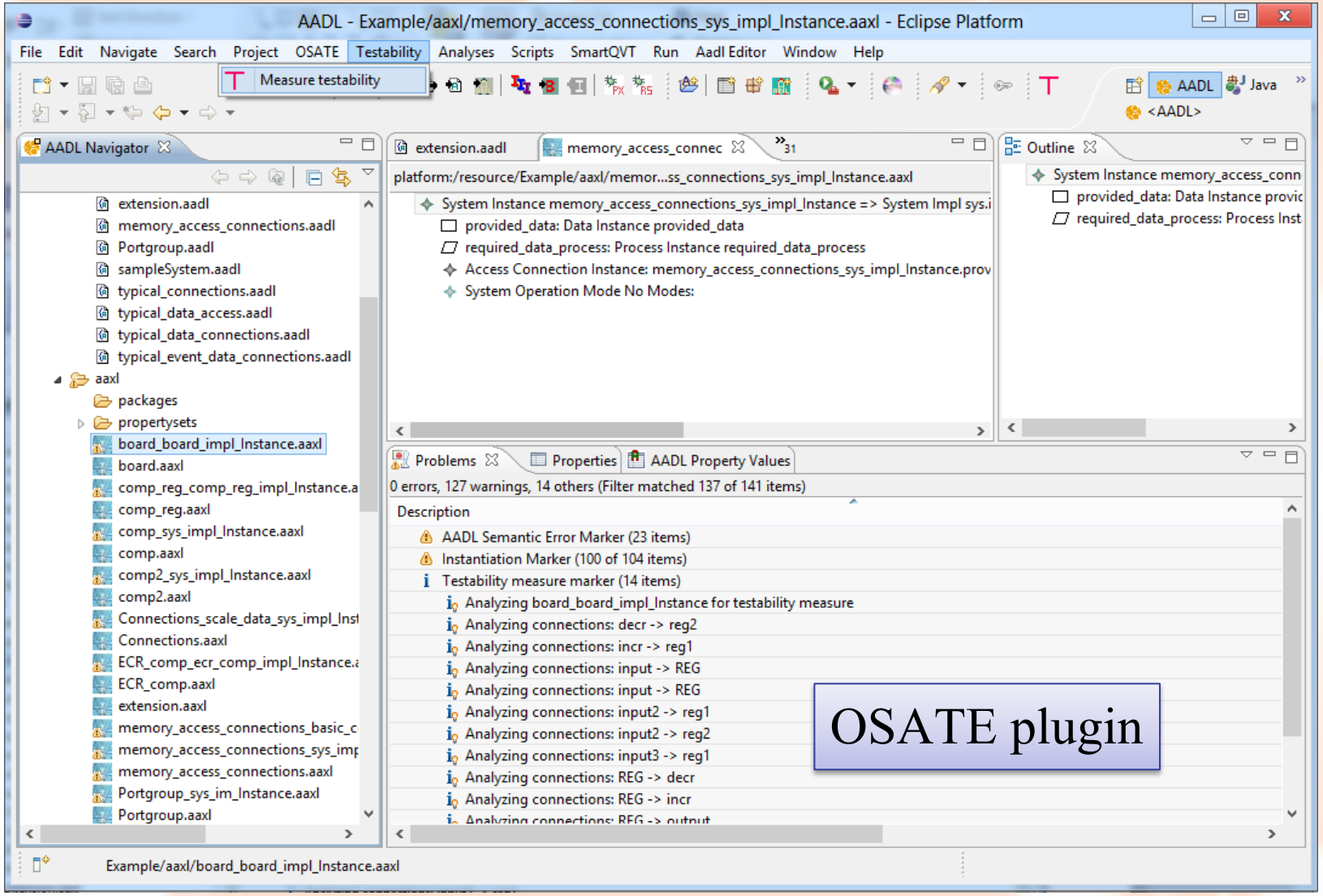
$TI'(Y{\downarrow}M, O{\downarrow}F) = TI'(Y{\downarrow}M', SO) = \text{flow}(v{\downarrow}r)$

Calculate upper bounded testability

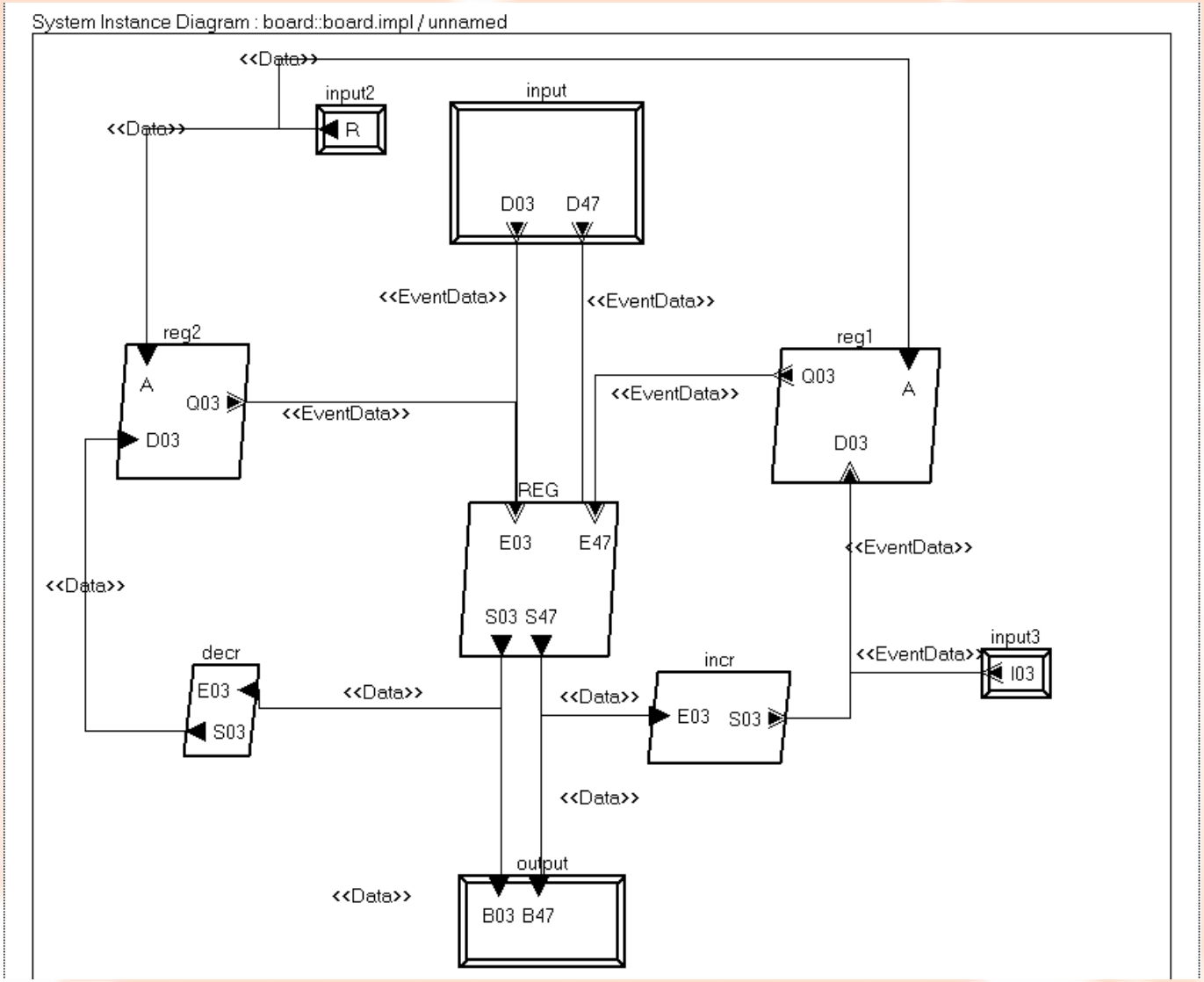




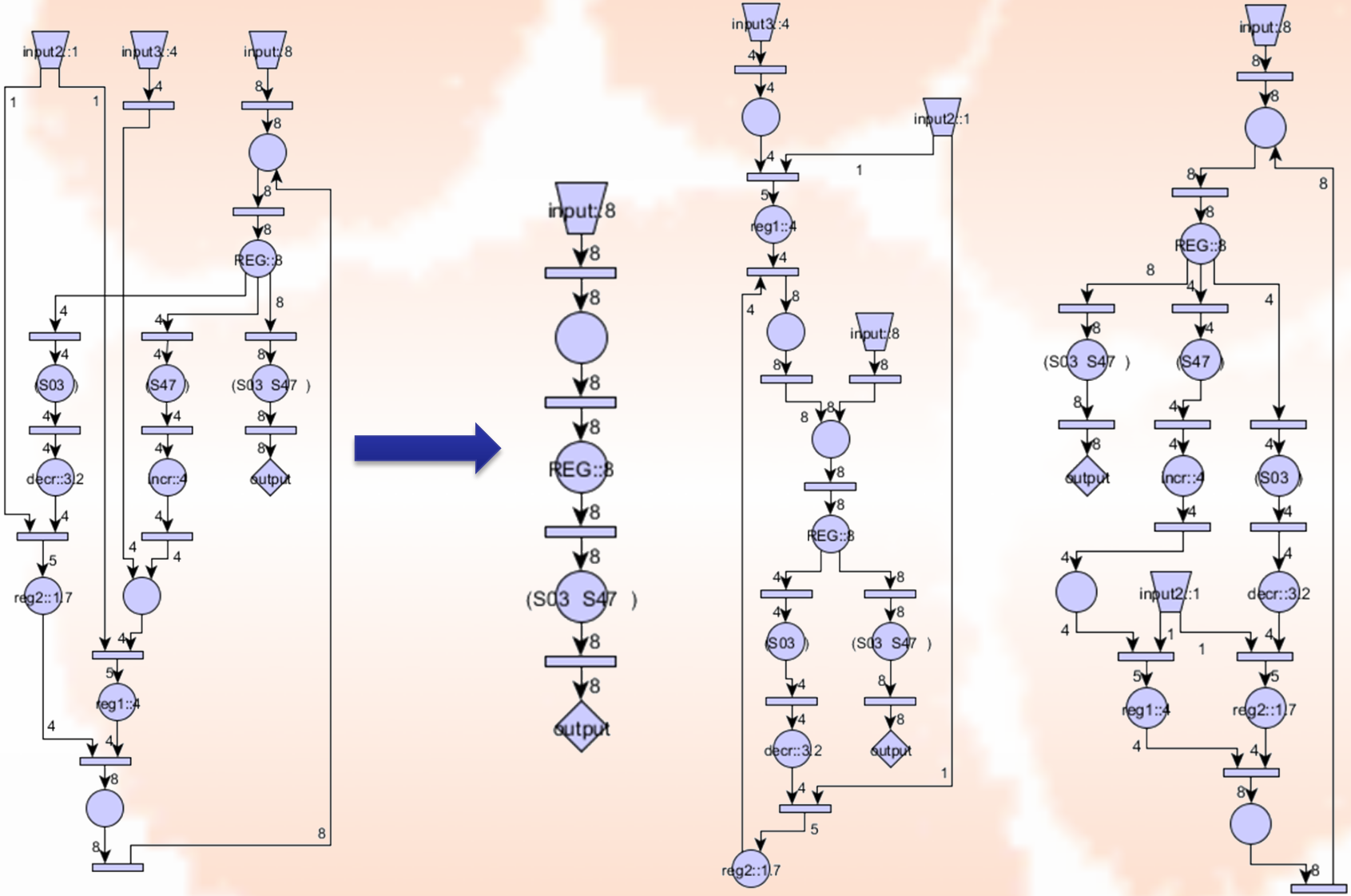
Experiment results



Experiment results



Experiment results



Experiment results

Flow 1 information:

Controllabilities:	Observabilities:	Testabilities:
CO(REG) = 1.00000	OB(REG) = 1.00000	TE(REG) = 1.00000

Flow 2 information:

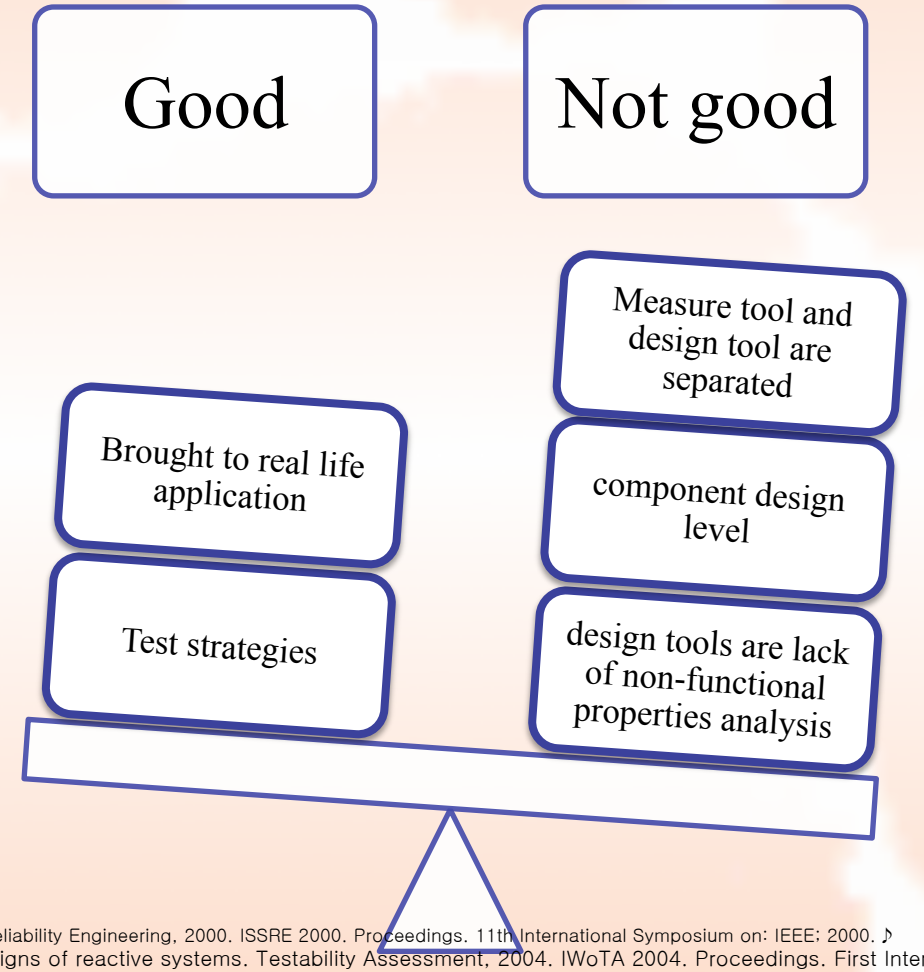
Controllabilities:	Observabilities:	Testabilities:
CO(reg1) = 1.00000	OB(reg1) = 1.00000	TE(reg1) = 1.00000
CO(REG) = 1.00000	OB(REG) = 1.00000	TE(REG) = 1.00000
CO(decr) = 1.00000	OB(decr) = 0.35355	TE(decr) = 0.35355
CO(reg2) = 0.57435	OB(reg2) = 1.00000	TE(reg2) = 0.57435

Flow 3 information:

Controllabilities:	Observabilities:	Testabilities:
CO(REG) = 1.00000	OB(REG) = 1.00000	TE(REG) = 1.00000
CO(incr) = 1.00000	OB(incr) = 1.00000	TE(incr) = 1.00000
CO(reg1) = 1.00000	OB(reg1) = 1.00000	TE(reg1) = 1.00000
CO(decr) = 1.00000	OB(decr) = 0.35355	TE(decr) = 0.35355
CO(reg2) = 0.57435	OB(reg2) = 1.00000	TE(reg2) = 0.57435

Related work

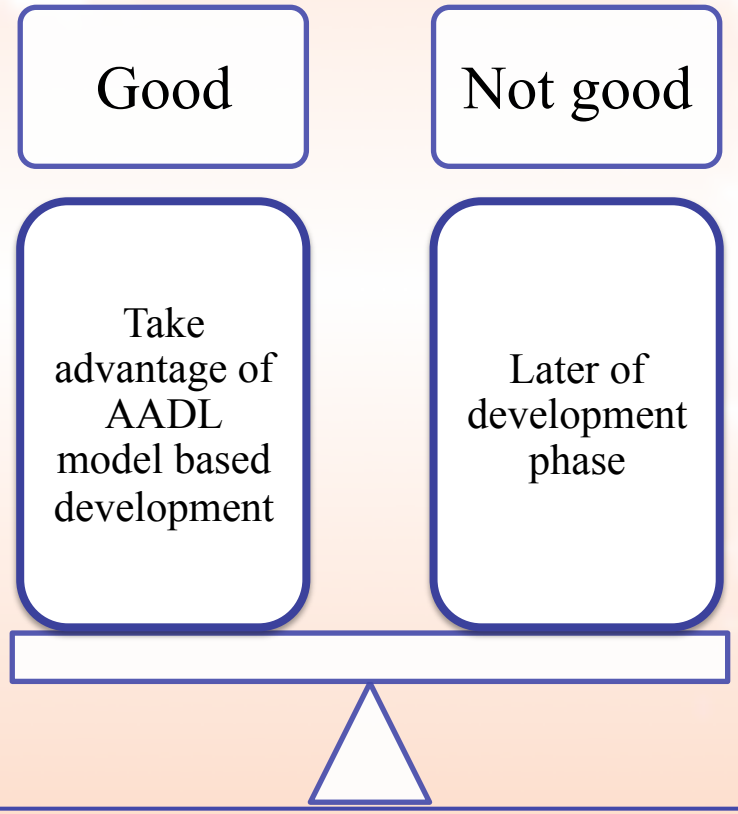
Testability analysis for data flow software (1-5)



(1) Analyzing testability on data flow designs. Software Reliability Engineering, 2000. ISSRE 2000. Proceedings. 11th International Symposium on: IEEE; 2000. ♪
(2) Automatic testability analysis for data-flow designs of reactive systems. Testability Assessment, 2004. IWoTA 2004. Proceedings. First Inter♪
♪
national Workshop on: IEEE; 2004. ♪
(3) Do H, Delaunay M, Robach C. MaC: A Testability Analysis Tool for Reactive Real-Time Systems. ERCIM news 2004;58:26.♪
(4) An Overview of the Architecture Analysis & Design Language (AADL) Error Model Annex. AADL Workshop; 2005. ♪
(5) Binh NT, Delaunay M, Robach C. Testability Analysis of Data-Flow Software. 2005. ♪

Related work

- Framework for model based testing AADL models (6)
- Test case generating based on AADL system compone nt modes (7)



(7) Research of embedded software testing method based on AADL modes. Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on: IEEE; 2011.
(6) A Model-Based Testing for AADL Model of Embedded Software. Quality Software, 2009. QSIC'09. 9th International Conference on: IEEE; 2009.

Summary and future work

Summary

- Develop a testability measure method for AADL models
- Develop a plugin for OSATE that automates the testability measurement.

Summary and future work

Future work

- Deal with control flow
- Extend for execution platform
- Utilize the plugin

Question

Thank you!