

TOWARDS PROBABILISTIC ASSESSMENT OF MODULARITY

Kevin Hoffman and Patrick Eugster, C.S. Department, Purdue University
ACoM 2008 @ OOPSLA

Questions of Interest

- **Q1)** Which program elements are the most depended upon?
- **Q2)** Which program elements are the most likely to change? (i.e. the most fragile?)
- **Q3)** Which program elements are the most likely to be affected by rippled effects given a set of changes?
- **Q4)** Which program elements were most influential during some execution of a test case?

Strategy for Answering

- Model software using Design Structure Matrices (DSMs)
- Assign weights to each cell as relevant to the question
- Formulate an *Assessment Graph* from the DSM
- Normalize data to produce a time-homogenous Markov chain
- Compute the stationary distribution of the Markov chain (à-la PageRank, EigenTrust)
- Overlay visualization onto DSM

DSM Generation

		1	2	3	4	5	6	7	8	9	10
generic_handlers	1	.									
exception_monitoring	2	X	.								
util	3	X		.							
util_handlers	4	X		X	.						
components_handlers	5	X				.	X				
components	6	X		X		X	.				
waf_handlers	7	X					X	.	X		
waf	8	X		X	X		X	X	.		
apps_handlers	9	X								.	X
apps	10	X		X	X		X		X	X	.

		A	B	C	D	E	F	G	H	I	J
		1	2	3	4	5	6	7	8	9	10
A	1										
B	2	X									
C	3										
D	4		X								
E	5				X						
F	6				X						
G	7				X	X					
H	8										
I	9							X			
J	10							X			

Weighted DSM Generation

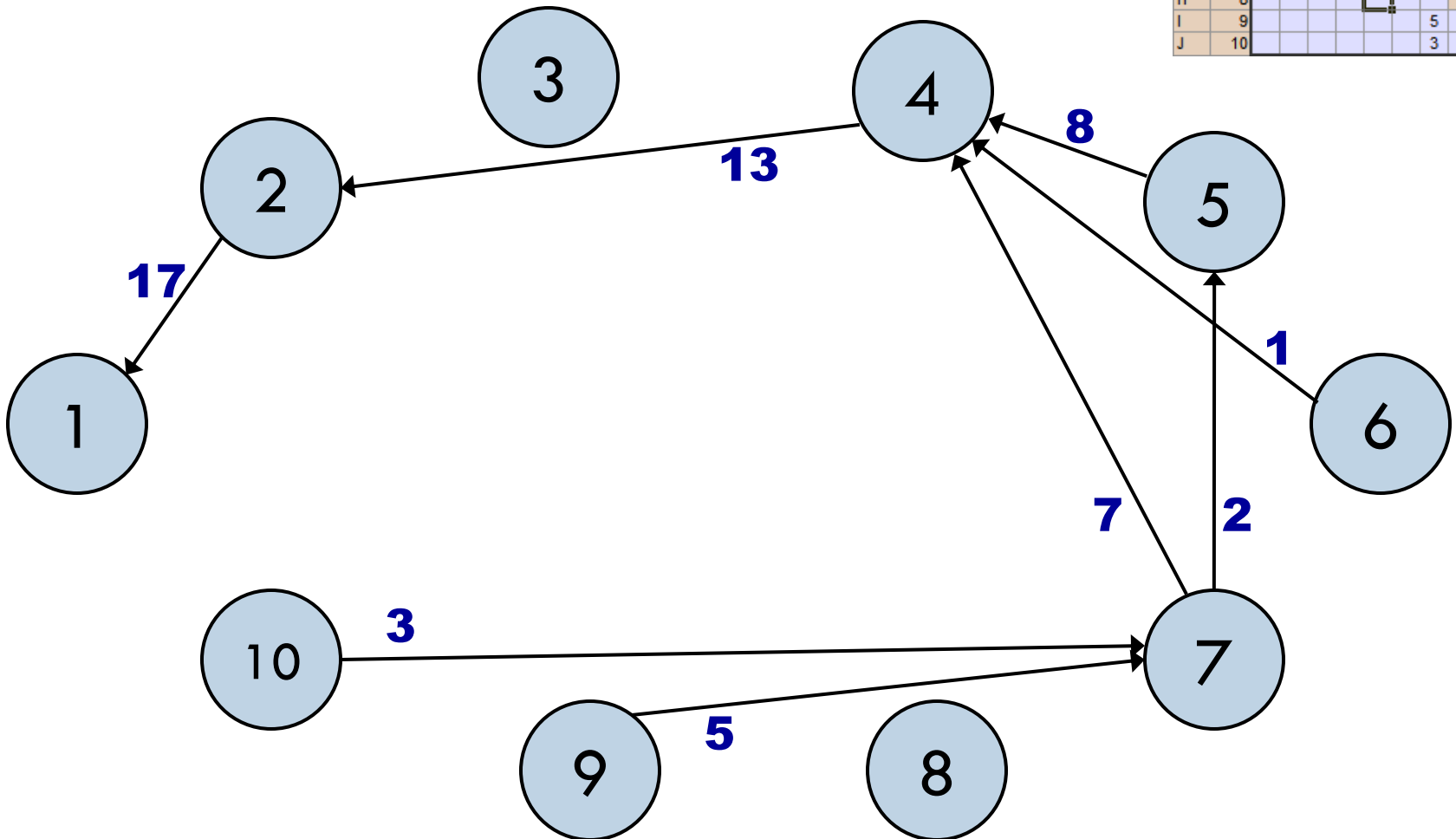
- Weights based on knowledge of other subsystems
- Weights based on actual use of other subsystems
- Weight assignment strategy governed by question

		1	2	3	4	5	6	7	8	9	10
generic_handlers	1	.									
exception_monitoring	2	1	.								
util	3	36	.								
util_handlers	4	2	1	.							
components_handlers	5	3			.	7					
components	6	94	70		2	.					
waf_handlers	7	1				1	.	6			
waf	8	27	2	1		64	1	.			
apps_handlers	9	6							.	13	
apps	10	113	253	1		843	149	5	.		

		A	B	C	D	E	F	G	H	I	J
		1	2	3	4	5	6	7	8	9	10
A	1										
B	2	17									
C	3										
D	4		13								
E	5				8						
F	6				1						
G	7				7	2					
H	8										
I	9								5		
J	10								3		

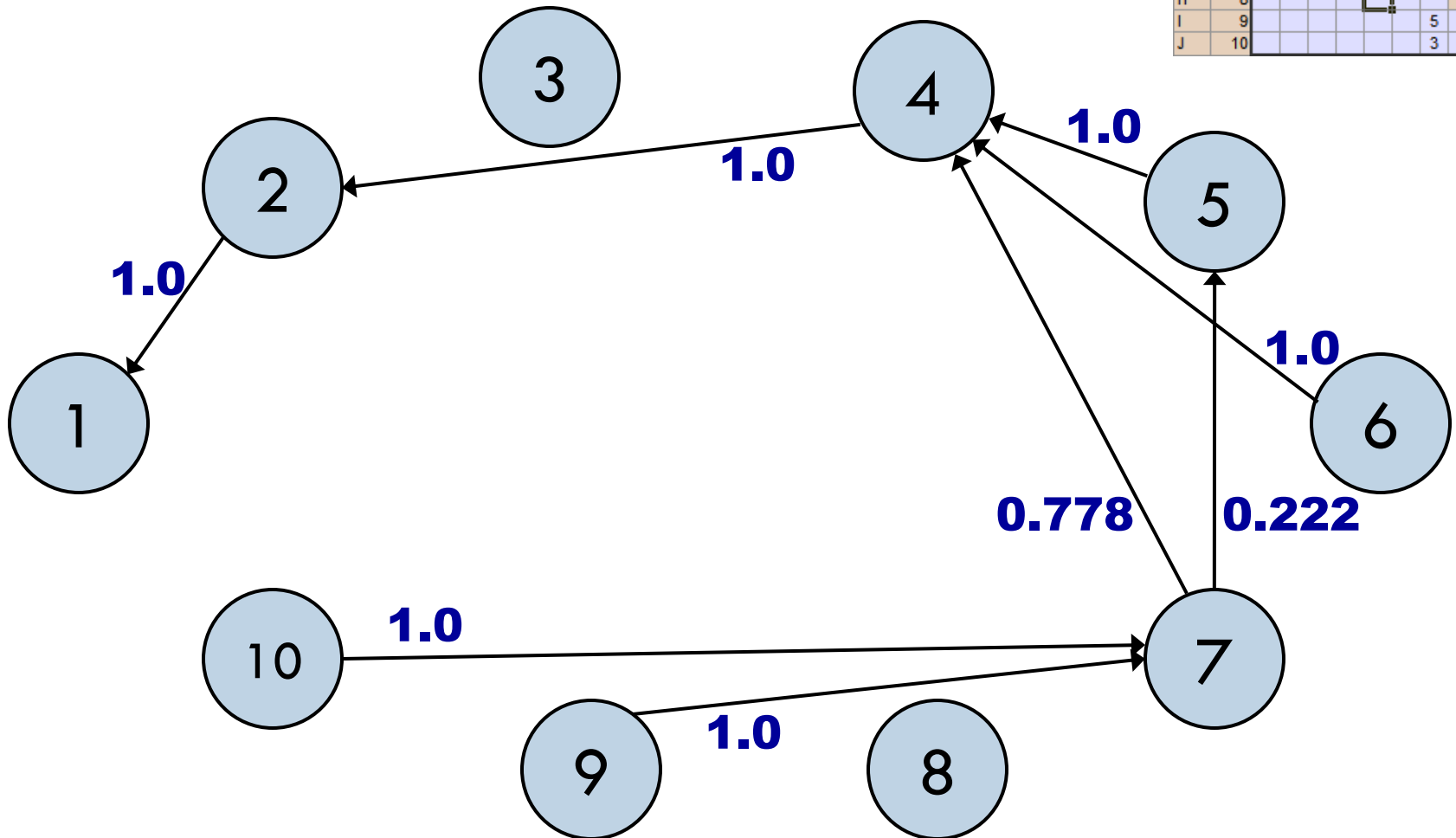
Form the Assessment Graph

	A	B	C	D	E	F	G	H	I	J
1										
2	17									
3										
4		13								
5				8						
6				1						
7				7	2					
8										
9								5		
10								3		



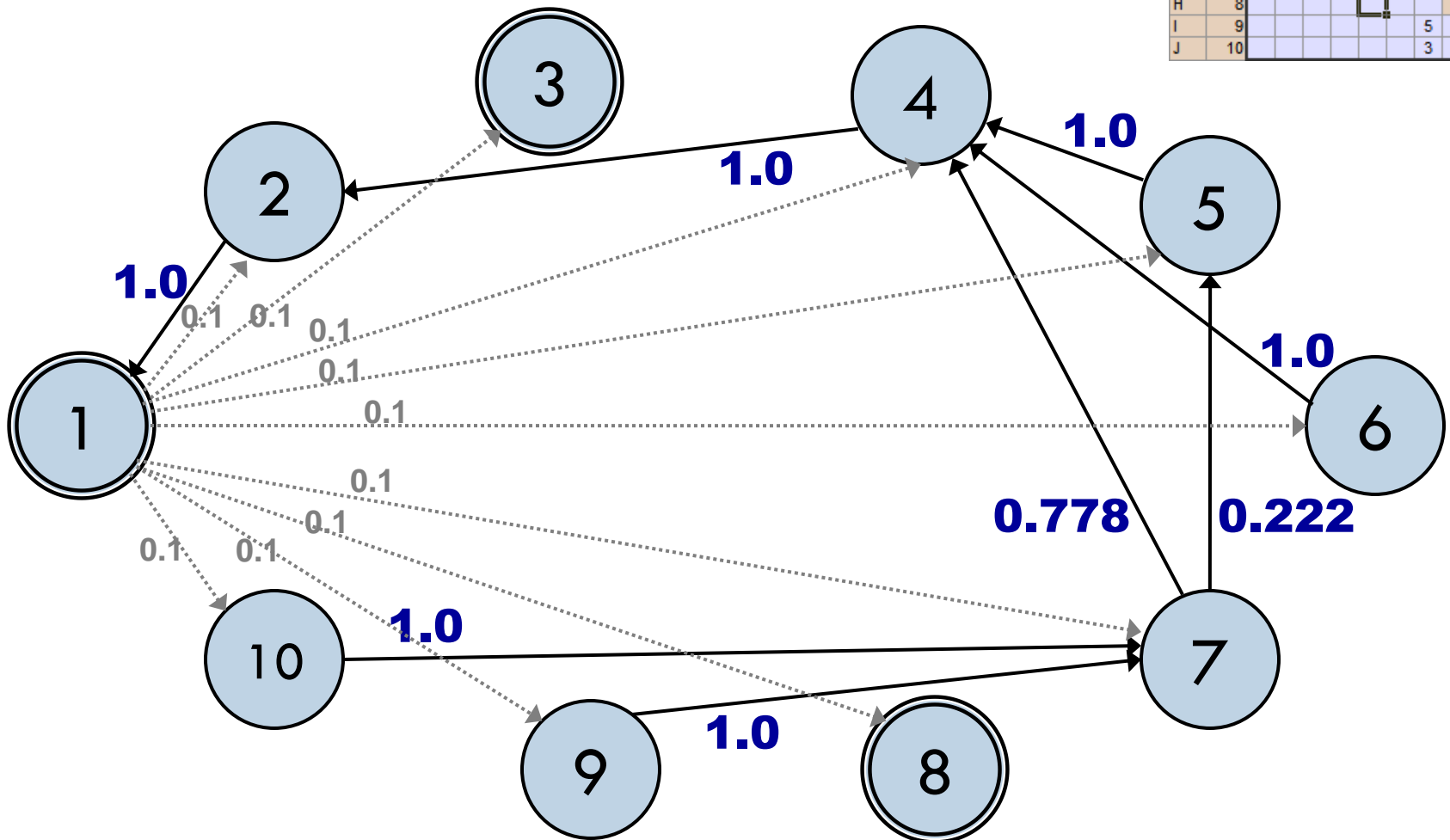
Normalize Edge Weights

	A	B	C	D	E	F	G	H	I	J
A	1									
B	2	17								
C	3									
D	4	13								
E	5			8						
F	6			1						
G	7			7	2					
H	8									
I	9							5		
J	10							3		



All Nodes Have > 0 out-Edges

	A	B	C	D	E	F	G	H	I	J
1	1	2	3	4	5	6	7	8	9	10
A	1									
B	2	17								
C	3									
D	4		13							
E	5			8						
F	6			1						
G	7			7	2					
H	8									
I	9							5		
J	10							3		



Metric Calculation

- Weighted edge matrix of Assessment Graph now forms a time-homogenous Markov chain
- Stationary distribution of the Markov chain provides the metric results
 - ▣ Calculated by finding the left eigenvector of the transition matrix of the Markov chain
 - ▣ Computed using Power Iteration algorithm
 - ▣ “Damping factor” to ensure convergence (guarantees aperiodicity of Markov chain)

Metric Calculation Example

		A	B	C	D	E	F	G	H	I	J
	1	2	3	4	5	6	7	8	9	10	
A	1										
B	2	17									
C	3										
D	4		13								
E	5				8						
F	6				1						
G	7				7	2					
H	8										
I	9								5		
J	10								3		

		A	B	C	D	E	F	G	H	I	J
	1	2	3	4	5	6	7	8	9	10	
A	1										
B	2	17									
C	3										
D	4		13								
E	5				8						
F	6				1						
G	7				7	2					
H	8										
I	9								5		
J	10								3		

Q1) Which program elements are the most depended upon?

- Calculate DSM weights based on coupling measures
- Form Assessment Graph from DSM and normalize
- Algorithm output defined as the...

Dependency Propagation Ranking

		A	B	C	D	E	F	G	H	I	J
		1	2	3	4	5	6	7	8	9	10
A	1										
B	2	17									
C	3										
D	4		13								
E	5				8						
F	6				1						
G	7				7	2					
H	8										
I	9							5			
J	10							3			

Q2) Which program elements are the most likely to change?

- Calculate DSM weights based on coupling measures
- Form Assessment Graph from DSM and normalize
- REVERSE DIRECTION of edges in graph
- Algorithm output defined as the...

Impact Propagation Ranking

		A	B	C	D	E	F	G	H	I	J
		1	2	3	4	5	6	7	8	9	10
A	1	1									
B	2	17	1								
C	3			1							
D	4		13		1						
E	5				8	1					
F	6					1	1				
G	7				7	2	1				
H	8							1			
I	9								1	1	
J	10							3			1

Dependency Propagation Ranking

		A	B	C	D	E	F	G	H	I	J
		1	2	3	4	5	6	7	8	9	10
A	1	1									
B	2	17	1								
C	3			1							
D	4		13		1						
E	5				8	1					
F	6					1	1				
G	7				7	2	1				
H	8							1			
I	9								1	1	
J	10							3			1

Impact Propagation Ranking

Q3) Which program elements are the most likely to be affected by rippled effects given a set of changes?

- Calculate the *Impact Propagation Ranking* for two versions of a program
- Define the absolute value of the difference between these vectors to be the...

Impact Shift Ranking

Q4) Which program elements were most influential during some execution of a test case?

- Construct weighted DSM from elements in execution traces
 - ▣ Which parts of execution traces
 - ▣ How to abstract to the appropriate level of detail?
 - ▣ What should be used to formulate the weights?
- Form Assessment Graph from DSM and normalize
- Use output of algorithm to understand Q4

Normalization Choices

By Row (Local)

		A	B	C	D	E	F	G	H	I	J
		1	2	3	4	5	6	7	8	9	10
A	1										
B	2	17									
C	3										
D	4		13								
E	5				8						
F	6				1						
G	7				7	2					
H	8										
I	9							5			
J	10							3			

By Table (Global)

		A	B	C	D	E	F	G	H	I	J
		1	2	3	4	5	6	7	8	9	10
A	1										
B	2	17									
C	3										
D	4		13								
E	5				8						
F	6				1						
G	7				7	2					
H	8										
I	9							5			
J	10							3			

util_xmldocuments	1	generic_handler_EJPs	1
util_tracer	2	exception_monitor_interface	2
components_servicelocator	3	exception_monitor_impl	3
components_address	4	util_xmldocuments	4
components_creditcard	5	util_tracer	5
components_lineitem	6	util_tracer_handlers	6
components_asyncsender	7	components_uidgen_handlers	7
components_contactinfo	8	components_signon_handlers	8
components_catalog	9	components_servicelocator	9
components_purchaseorder	10	components_lineitem	10
components_cart	11	components_address	11
components_customer	12	components_creditcard	12
components_supplierpo	13	components_asyncsender	13
components_processmanager	14	components_contactinfo	14
components_signon	15	components_catalog	15
components_uidgen	16	components_supplierpo	16
components_mailer	17	components_purchaseorder	17
components_encodingfilter	18	components_customer	18
waf_controller	19	components_cart	19
waf_view	20	components_uidgen	20
apps_opc	21	components_signon	21
apps_admin	22	components_processmanager	22
apps_petstore	23	components_mailer	23
apps_supplier	24	components_encodingfilter	24
generic_abstract_handlers	25	waf_controller_handlers	25
generic_components_handlers	26	waf_controller	26
components_servicelocator_handlers	27	waf_view	27
components_asyncsender_handlers	28	apps_admin_handlers	28
components_catalog_handlers	29	apps_petstore_handlers	29
components_purchaseorder_handlers	30	apps_supplier_handlers	30
components_cart_handlers	31	apps_opc	31
components_supplierpo_handlers	32	apps_admin	32
components_processmanager_handlers	33	apps_petstore	33
components_signon_handlers	34	apps_supplier	34
components_uidgen_handlers	35		
components_mailer_handlers	36		
util_xmldocuments_handlers	37		
waf_controller_handlers	38		
waf_view_handlers	39		
apps_petstore_handlers	40		
apps_supplier_handlers	41		
apps_admin_handlers	42		
apps_opc_handlers	43		
exception_monitor_interface	44		
exception_monitor_impl	45		

Future Directions

- Refine metric definitions
 - Appropriate normalization strategies
- Other sources for weights
 - Logical coupling
 - Data flow analysis
 - Execution trace data
- Apply to several large bodies of software
- Integrated tool chain for fast visualization

Send comments and questions to kjhoffma@cs.purdue.edu