Design Analysis: Design Structure Matrices

K.J. Sullivan, W.G. Griswold, Y. Cai, and B. Hallen. The Structure and Value of Modularity in Software Design. Foundations of Software Engineering, 2001.

Carliss Baldwin and Kim Clark. Design Rules: The Power of Modularity. MIT Press.

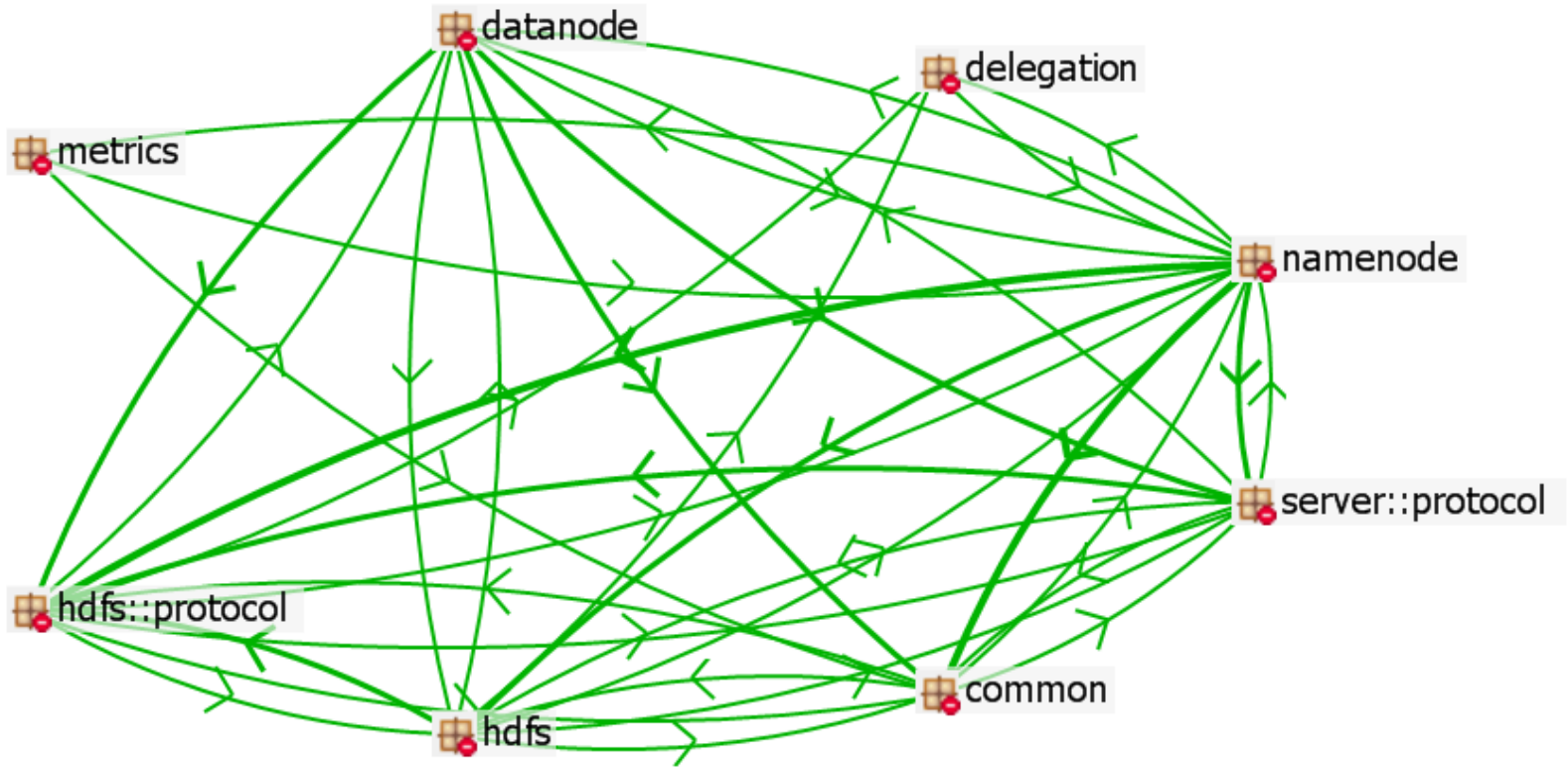
15-214: Principles of Software Construction

Jonathan Aldrich

Module Dependencies

- Ideal: changes restricted to within a module
  - Goal of information hiding
  - Need to know which decisions may change
- Reality: some changes are surprises
  - These may affect module *interfaces*
  - Changes *propagate* to dependent modules
- How bad is this propagation?
  - It depends on coupling

Design Dependencies in Hadoop



Credit: Rick Kazman

The Design Structure Matrix (DSM)

- A DSM is an alternative to a directed graph
- An N by N matrix
  - Each row is a module in the system
  - Columns also labeled with modules, in same order
- Dependencies are marked with X's
  - Marking row A, column B means A depends on B
  - The diagonal is ignored (self-dependence)
- Makes it easy to see common patterns

Credit: Rick Kazman

Design Structure Matrix (DSM) Example

	Model	GUI
Model	0	
GUI	X	0

Credit: Rick Kazman

Varying Degrees of Complexity: Fully Connected DSM

	A	B	C	D	E
A	O	X	X	X	X
B	X	O	X	X	X
C	X	X	O	X	X
D	X	X	X	O	X
E	X	X	X	X	O

Credit: Rick Kazman

Varying Degrees of Complexity: Fully Disconnected DSM

	A	B	C	D	E
A	O				
B		O			
C			O		
D				O	
E					O

Credit: Rick Kazman

Varying Degrees of Complexity: A Layered Architecture

	Kernel	Ring 1	Ring 2	Ring 3	Ring 4
Kernel	O				
Ring 1	X	O			
Ring 2	X	X	O		
Ring 3	X	X	X	O	
Ring 4	X	X	X	X	O

Credit: Rick Kazman

Varying Degrees of Complexity: A Strictly Layered Architecture

	OS	VM	I/O lib	Middle ware	App
OS	O				
VM	X	O			
I/O lib		X	O		
Middle ware			X	O	
App				X	O

Credit: Rick Kazman

Design Structure Matrices

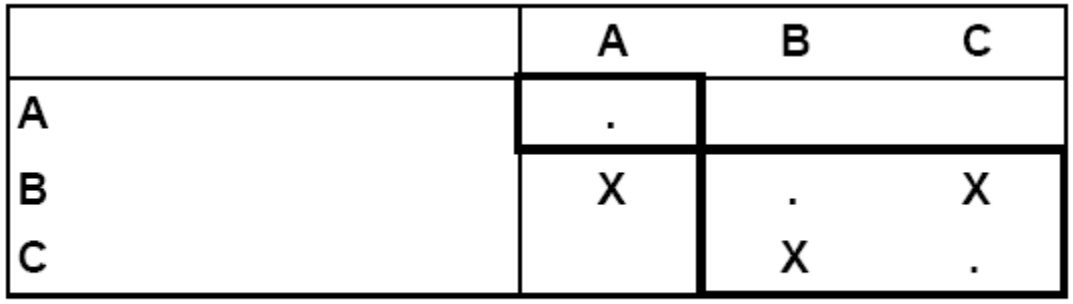
	A	B	C
A	.		
B	X	.	X
C		X	.

Figure 1: DSM for a design of three parameters.

More terminology

- B is *hierarchically dependent* on A
  - If you change A, you might have to change B as well
  - Suggests you should implement A first
- B and C are *interdependent*
- C and A are *independent*

Design Structure Matrices



	A	B	C
A	.		
B	X	.	X
C		X	.

Figure 2: DSM for a proto-modular design.

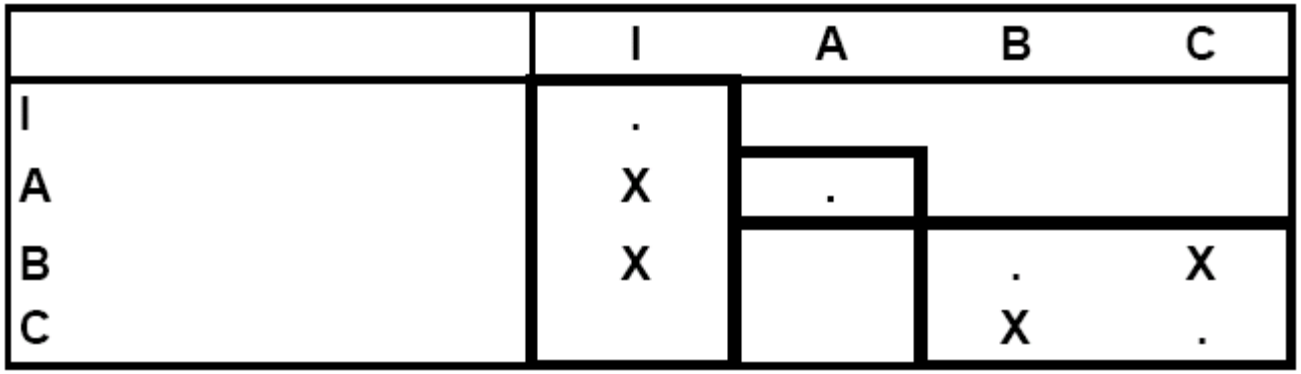
- Lines show clustering into proto-modules
  - Indicates several design decisions will be managed together
- True modules should be independent
  - i.e., no marks outside of its cluster
  - Not true here because B (in the B-C cluster) depends on A

Varying Degrees of Complexity: Layered Modules

	A	B	C	D	E
A	O				
B		O	X		
C	X	X	O		
D	X			O	
E	X	X		X	O

Credit: Rick Kazman

Design Structure Matrices

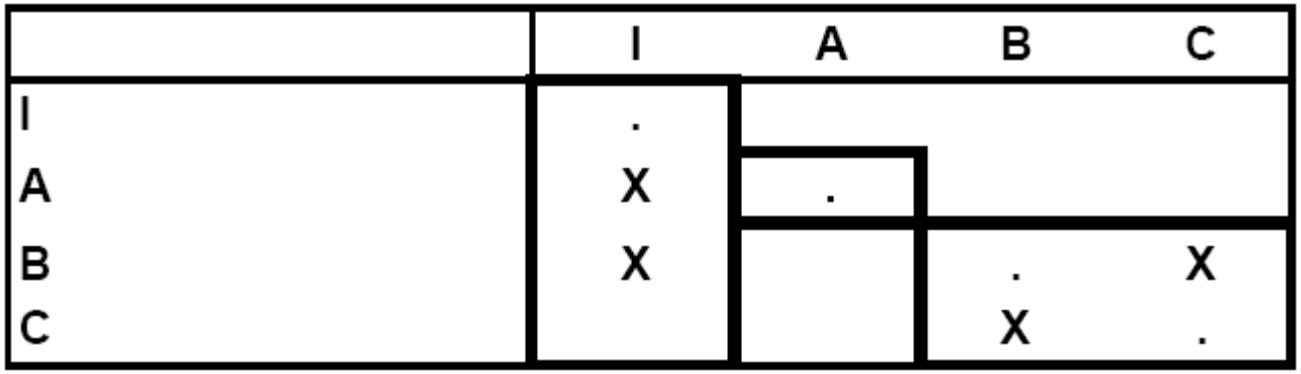


	I	A	B	C
I	.			
A	X	.		
B	X		.	X
C			X	.

Figure 3: DSM for a modular design obtained by splitting.

- Interface reifies the dependence as a separate entity
  - Instead of B depending on A, now A and B both depend on I
  - Serves to decouple A and B
  - Think of I as the interface of A

Value of Modularity



	I	A	B	C
I	.			
A	X	.		
B	X		.	X
C			X	.

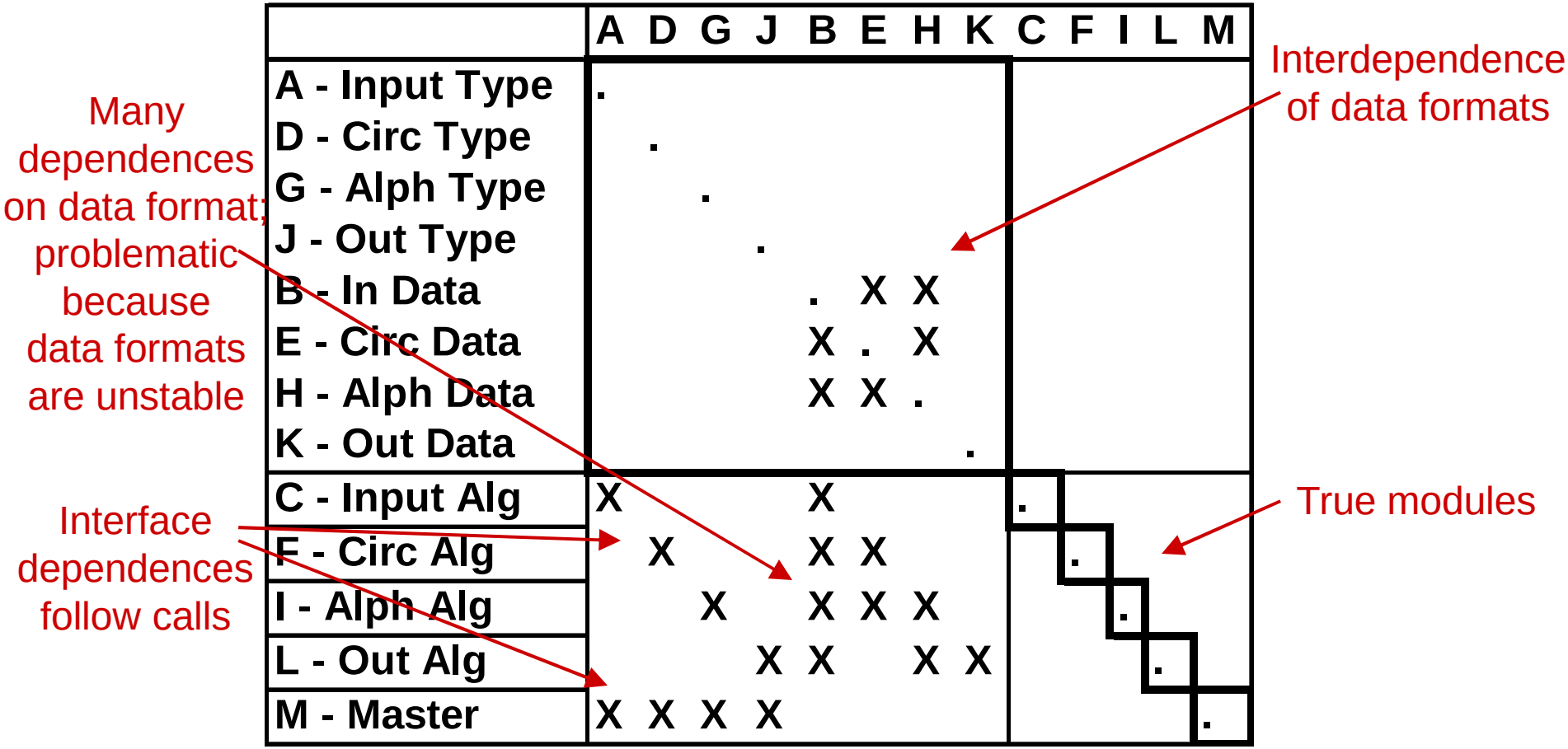
Figure 3: DSM for a modular design obtained by splitting.

- Information Hiding
  - If you can anticipate which design decisions are likely to change and hide them in a module, then evolving the system when these changes occur will cost less
  - Reduces maintenance cost and time to market
  - Frees resources to invest in quality, features

KWIC Design #1

	A	D	G	J	B	E	H	K	C	F	I	L	M
A - Input Type	.												
D - Circ Type		.											
G - Alph Type			.										
J - Out Type				.									
B - In Data					.								
E - Circ Data						.							
H - Alph Data							.						
K - Out Data								.					
C - Input Alg									.				
F - Circ Alg										.			
I - Alph Alg											.		
L - Out Alg												.	
M - Master													.

KWIC Design #1



	A	D	G	J	B	E	H	K	C	F	I	L	M
A - Input Type	.												
D - Circ Type		.											
G - Alph Type			.										
J - Out Type				.									
B - In Data					.	X	X						
E - Circ Data					X	.	X						
H - Alph Data					X	X	.						
K - Out Data								.					
C - Input Alg	X				X				.				
F - Circ Alg		X			X	X				.			
I - Alph Alg			X		X	X	X				.		
L - Out Alg				X	X		X	X				.	
M - Master	X	X	X	X									.

KWIC Design #2

	N	A	D	G	J	O	P	B	C	E	F	H	I	K	L	M
N - Line Type	.															
A - Input Type		.														
D - Circ Type			.													
G - Alph Type				.												
J - Out Type					.											
O - Line Data						.										
P - Line Alg							.									
B - In Data								.								
C - Input Alg									.							
E - Circ Data										.						
F - Circ Alg											.					
H - Alph Data												.				
I - Alph Alg													.			
K - Out Data														.		
L - Out Alg															.	
M - Master																.

KWIC Design #2

	N	A	D	G	J	O	P	B	C	E	F	H	I	K	L	M
N - Line Type	.															
A - Input Type		.														
D - Circ Type			.													
G - Alph Type				.												
J - Out Type					.											
O - Line Data	X					.	X									
P - Line Alg	X					X	.									
B - In Data		X						.	X							
C - Input Alg	X	X						X	.							
E - Circ Data	X		X							.	X					
F - Circ Alg	X		X							X	.					
H - Alph Data				X								.	X			
I - Alph Alg			X	X								X	.			
K - Out Data					X									.	X	
L - Out Alg			X	X	X									X	.	
M - Master	X	X	X	X	X											.

Dependence on interfaces

True modules

Which Design is Better?

	A	D	G	J	B	E	H	K	C	F	I	L	M
A - Input Type	.												
D - Circ Type		.											
G - Alph Type			.										
J - Out Type				.									
B - In Data					.	X	X						
E - Circ Data					X	.	X						
H - Alph Data					X	X	.						
K - Out Data								.					
C - Input Alg	X				X				.				
F - Circ Alg		X			X	X				.			
I - Alph Alg			X		X	X	X				.		
L - Out Alg				X	X		X	X				.	
M - Master	X	X	X	X									.

	N	A	D	G	J	O	P	B	C	E	F	H	I	K	L	M
N - Line Type	.															
A - Input Type		.														
D - Circ Type			.													
G - Alph Type				.												
J - Out Type					.											
O - Line Data	X					.	X									
P - Line Alg	X					X	.									
B - In Data		X						.	X							
C - Input Alg	X	X						X	.							
E - Circ Data	X		X							.	X					
F - Circ Alg	X		X							X	.					
H - Alph Data				X								.	X			
I - Alph Alg			X	X								X	.			
K - Out Data					X									.	X	
L - Out Alg			X	X	X									X	.	
M - Master	X	X	X	X	X											.

Comparing the Designs

	A	D	G	J	B	E	H	K	C	F	I	L	M
A - Input Type	.												
D - Circ Type		.											
G - Alph Type			.										
J - Out Type				.									
B - In Data					.	X	X						
E - Circ Data					X	.	X						
H - Alph Data					X	X	.						
K - Out Data								.					
C - Input Alg	X				X				.				
F - Circ Alg		X			X	X				.			
I - Alph Alg			X		X	X	X				.		
L - Out Alg				X	X			X	X			.	
M - Master	X	X	X	X									.

	N	A	D	G	J	O	P	B	C	E	F	H	I	K	L	M
N - Line Type	.															
A - Input Type		.														
D - Circ Type			.													
G - Alph Type				.												
J - Out Type					.											
O - Line Data	X					.	X									
P - Line Alg	X					X	.									
B - In Data		X						.	X							
C - Input Alg	X	X						X	.							
E - Circ Data	X		X							.	X					
F - Circ Alg	X		X							X	.					
H - Alph Data				X								.	X			
I - Alph Alg			X	X								X	.			
K - Out Data					X									.	X	
L - Out Alg			X	X	X									X	.	
M - Master	X	X	X	X	X											.

- Both designs allow changes within modules
- However, in the first design the modules do not hide much
  - Many dependencies (2-4 per module) on data structures
  - Data structure dependencies are strong: they restrict algorithms used
  - Furthermore, data structures are likely to change
- The second design is much less constrained
  - Fewer dependencies (1-2 per module)
  - Interfaces are more abstract, do not restrict code as much
  - Interfaces are more stable in the face of likely changes
- Result: design 2 minimizes re-engineering in response to change

EDSMs: Considering Possible Changes

- Environment and Design Structure Matrices
  - Sullivan et al., ESEC/FSE 2001
- Add changes as *environmental parameters*
  - *Note: slightly more concrete than what Sullivan et al. propose*
  - Only partially controlled by designer
  - May affect each other
  - May affect design decisions in code
- What interfaces are affected?
  - Information hiding: interfaces should be stable
- What implementations are affected?
  - Information hiding hypothesis: should be local to a module

Effect of Change – Design #1

	V	W	X	Y	Z	A	D	G	J	B	E	H	K	C	F	I	L	M
V - Input Fmt	.																	
W - Store Mem		.																
X - Compress			.															
Y - Shift Store				.														
Z - Amortize					.													
A - Input Type						.												
D - Circ Type							.											
G - Alph Type								.										
J - Out Type									.									
B - In Data										.	X	X						
E - Circ Data										X	.	X						
H - Alph Data										X	X	.						
K - Out Data													.					
C - Input Alg						X				X				.				
F - Circ Alg							X			X	X				.			
I - Alph Alg								X		X	X	X				.		
L - Out Alg									X	X		X	X				.	
M - Master						X	X	X	X									.

Effect of Change – Design #1

	V	W	X	Y	Z	A	D	G	J	B	E	H	K	C	F	I	L	M
V - Input Fmt	.																	
W - Store Mem		.																
X - Compress			.															
Y - Shift Store				.														
Z - Amortize					.													
A - Input Type						.												
D - Circ Type							.											
G - Alph Type								.										
J - Out Type									.									
B - In Data		X	X							.	X	X						
E - Circ Data				X						X	.	X						
H - Alph Data				X	X					X	X	.						
K - Out Data													.					
C - Input Alg	X	X	X			X				X				.				
F - Circ Alg		X	X	X			X			X	X				.			
I - Alph Alg		X	X	X	X			X		X	X	X				.		
L - Out Alg		X	X	X	X				X	X		X	X				.	
M - Master					X	X	X	X	X									.

Unstable data interfaces depend on changes

Algorithms depend on data

Effect of Change – Design #2

	V	W	X	Y	Z	N	A	D	G	J	O	P	B	C	E	F	H	I	K	L	M
V - Input Fmt	.																				
W - Store Mem		.																			
X - Compress			.																		
Y - Shift Store				.																	
Z - Amortize					.																
N - Line Type						.															
A - Input Type							.														
D - Circ Type								.													
G - Alph Type									.												
J - Out Type										.											
O - Line Data						X					.	X									
P - Line Alg						X					X	.									
B - In Data							X						.	X							
C - Input Alg						X	X						X	.							
E - Circ Data						X		X							.	X					
F - Circ Alg						X		X							X	.					
H - Alph Data									X								.	X			
I - Alph Alg								X	X								X	.			
K - Out Data										X									.	X	
L - Out Alg								X	X	X									X	.	
M - Master						X	X	X	X	X											.

Effect of Change – Design #2

Interfaces are stable

Effect of change is localized

	V	W	X	Y	Z	N	A	D	G	J	O	P	B	C	E	F	H	I	K	L	M
V - Input Fmt	.																				
W - Store Mem		.																			
X - Compress			.																		
Y - Shift Store				.																	
Z - Amortize					.																
N - Line Type						.															
A - Input Type							.														
D - Circ Type								.													
G - Alph Type									.												
J - Out Type										.											
O - Line Data		X	X			X					.	X									
P - Line Alg		X	X			X					X	.									
B - In Data	X						X						.	X							
C - Input Alg	X					X	X						X	.							
E - Circ Data				X		X		X							.	X					
F - Circ Alg				X		X		X							X	.					
H - Alph Data					X				X								.	X			
I - Alph Alg					X			X	X								X	.			
K - Out Data										X									.	X	
L - Out Alg								X	X	X									X	.	
M - Master						X	X	X	X	X											.

Comparison

- Design 1 hides information better
  - Interfaces are unaffected by likely change scenarios
  - Changes required to implement likely change scenarios are local

Summary

- DSMs are a structured way of thinking about the value of design
  - Are design decisions isolated to a module, or do they affect several modules?
  - How do modules depend on interfaces?
  - On which parts of the system can I experiment independently?
  - How much value is there in the experiments?
    - Technical potential of the module
- EDSMs incorporate change scenarios
  - How are interfaces and code affected by change?
- More to explore
  - Baldwin and Clark – discuss value of modularity
  - Sullivan and Griswold – apply B&C to S/W, introduce EDSMs
  - Lattix LDM tool – derives DSMs from code