An Analysis Of Modularity In Aspect Oriented Design

Cristina Videria Lopes and Sushil Krishna Bajracharya
(Authors)

Sushil K Bajracharya
sbajrach@ics.uci.edu

Department of Informatics
Donald Bren School of Information and Computer Sciences
University of California, Irvine

Outline

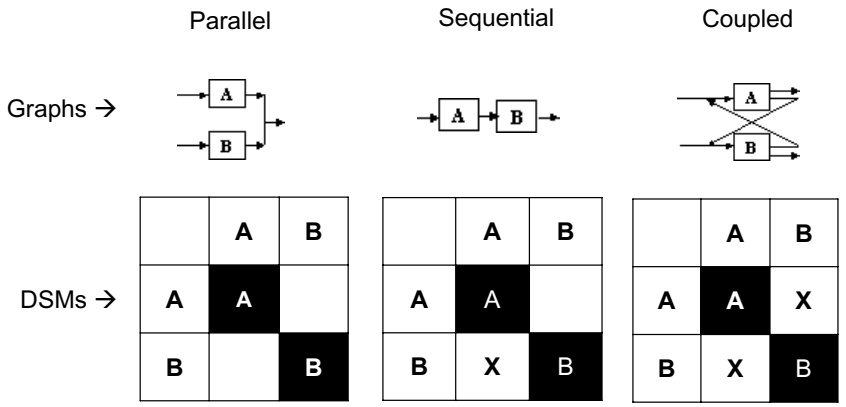
- Observing the effect of applying Aspect-oriented modularizations on the overall value of existing design
  - Tracing the evolution of a sample web-services application
  - Assessing the 'values' of five different design versions
- Using Net Options Value (NOV) [BC00] as a quantitative model for evaluating the value of design options
  - Modeling modular dependencies using Dependency Structure Matrices (DSMs)
  - Mapping design parameters in a software application to the parameters in the NOV framework
- Observations
- Open issues and future directions

Design Variants

Five different versions of design are studied as an existing application, ***StoreLocator,*** is modified to create a similar application ***WineryLocator***

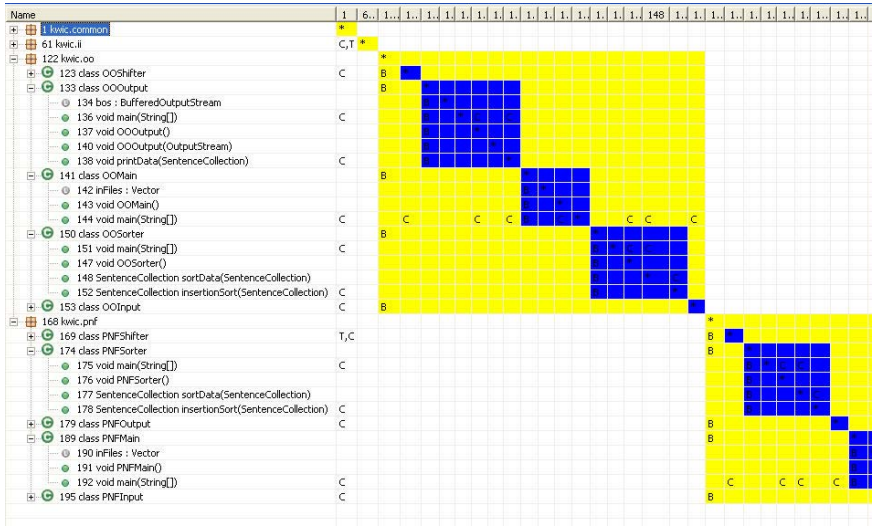
1. *StoreLocator* (*s1*)
   - Locates coffee stores in US, provides maps and directions
   - Uses MapPoint WebServices from Microsoft
2. *WineryLocator* (*w1*)
   - Locates wineries in California filtered by users' preferences, generates winery-tour, provides maps and directions
   - Uses MapPoint + a custom built web-service for locating wineries
3. *WineryLocator* (*w2*), adds a logging feature to *w1*
4. *WineryLocator* (*w3*), refactored version of *w2* with:
   - A simpler API
   - Interfaces from external services are hidden
5. *WineryLocator (w4),* Aspect-oriented modularization of w3

Dependency Structure Matrix

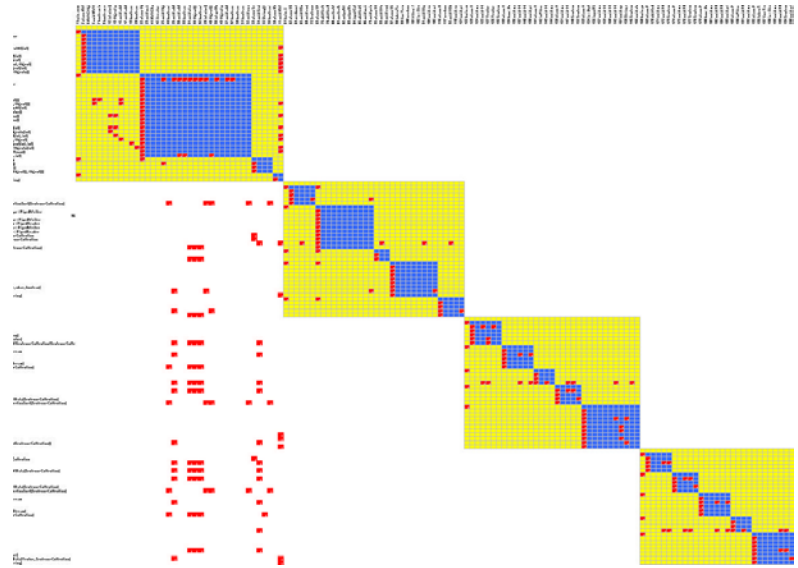


Three configurations of design parameters that characterize a system
(www.dsmweb.org)

DSM plugin for Eclipse



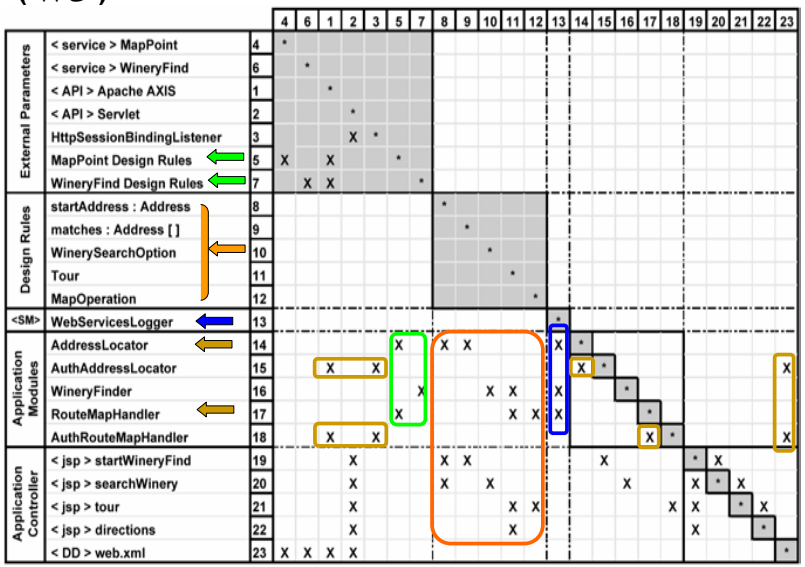
DSM worksheet



Categories of Design Parameters

- *Architecture* provides a framework that allows for both independence of structure and integration of function
  - External Parameters: Libraries, Frameworks, External Services
- Design Rules are decisions common to modules that are unlikely to change
  - Design rules constitute the interfaces to connect modules with each other (Application APIs)
- Module
  - strongly connected structural elements that are grouped together
  - increases the range of manageable complexity, allows concurrent work, and accommodates uncertainty
  - Application Modules: Perform application specific tasks
  - Subsidiary Modules: Contribute to subsidiary functionalities (non-functional requirements)
  - Application Controller: Modules that use the design rules as interface to access the functionalities provided by application modules

Winery Locator with new Design Rules (w3)



Net Options Value

- A mathematical model for evaluating design options based on the economic theory of real options
  - Design is a value seeking process
  - Making changes in design is an experiment with uncertain outcome
  - In a modular system, the total Value of the System is the sum of the NOVs of each module

NOV of i^{th} module = max_{ki} [**Expected value** from *k*-experiments - **Cost** incurred in *k*-experiments - **Visibility cost** to replace i^{th} module]

  - max_{ki} is the maximum value out of all experiments for the i_{th} module
  - **Cost** is the redesign cost for each of the *k* number of experiments
  - **Visibility Cost** is the cost incurred in making changes in all other modules that depend on the i_{th} module

Net Options Value

- A mathematical model for evaluating design options based on the economic theory of real options
  - Design is a value seeking process
  - Making changes in design is an experiment with uncertain outcome
  - In a modular system, the total Value of the System is the sum of the NOVs of each module

NOV of i^{th} module = max_{ki} [**Expected value** from *k*-experiments - **Cost** incurred in *k*-experiments - **Visibility cost** to replace i^{th} module]

Benefit Investment

$$NOV_i = max_{ki}\{\sigma_i n_i^{1/2} Q(k_i) - C_i(n_i)k_i - Z_i\}$$

Degree of *uncertainty* inherent in a module

Heuristics for σ

- *Environmental Factors govern the value of* σ [sul+01]
  - Users, Data, Deployed Environment
- Technical Potential (σ_i) of modules in *WineryLocator*

$$\sigma_i = f(e_i, p_i) = (e_i + 1)\ X\ p_i$$

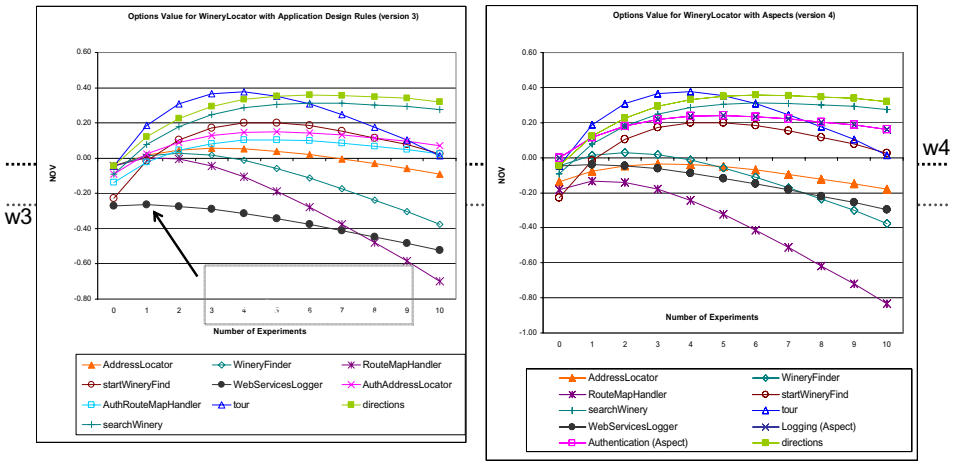
- *e* = number of *external parameters* that a module depends on
- *p* = module's relevance to the *end users*
  - p_i = 2, for Application Controller modules
  - p_i = 1, for Functional Modules, Subsidiary Modules and Design Rules
  - p_i = 0, for the deployment descriptor

NOV results for each design

Design	ID	NOV	I_c %	I_s %	I_{w1} %
StoreLocator	s	0.72	NA	NA	NA
WineryLocator	w1	1.38	91.41	91.42	NA
WineryLocator with Logging	w2	1.41	2.18	95.6	2.18
WineryLocator with design rules for application	w3	1.59	12.59	120.2	15.05
WineryLocator with Aspects	w4	1.76	24.55	143.6	27.28

I_c = Cumulative increase in value
I_s = Net increase in value with respect to 's'
I_{w1} = Net increase in value with respect to 'w1'

Effect of Aspect Oriented Modularization



Before Aspects - Design Version 'w3'

After Aspects - Design Version 'w4'

WineryLocator with Aspects (w4)

			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
External Parameters	< service > MapPoint	1	*																						
	< service > WineryFind	2		*																					
	< API > Apache AXIS	3			*																				
	< API > Servlet	4				*																			
	HttpSessionBindingListener	5					X	*																	
	MapPoint Design Rules	6	X		X				*																
	WineryFind Design Rules	7		X	X					*															
Design Rules	startAddress : Address	8									*														
	matches : Address [ ]	9										*													
	WinerySearchOption	10											*												
	Tour	11												*											
	MapOperation	12													*										
Application Modules	AddressLocator	13						X		X	X					*									
	WineryFinder	14							X				X	X			*								
	RouteMapHandler	15						X					X	X				*							
Application Controller	< jsp > startWineryFind	16				X				X	X				X				*	X					
	< jsp > searchWinery	17				X				X		X				X			X	*	X				
	< jsp > tour	18				X							X	X			X		X	X	*				
	< jsp > directions	19				X							X						X			*			
	< DD > web.xml	20	X	X	X	X																	*		
Subsidiary Modules	WebServicesLogger	21																						*	
	< Aspect > Logging	22						X	X														X	*	
	< Aspect > Authentication	23			X		X									X		X				X			*

Winery Locator with new Design Rules (w3)

			4	6	1	2	3	5	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
External Parameters	< service > MapPoint	4	*																						
	< service > WineryFind	6		*																					
	< API > Apache AXIS	1			*																				
	< API > Servlet	2				*																			
	HttpSessionBindingListener	3					X	*																	
	MapPoint Design Rules	5	X		X				*																
	WineryFind Design Rules	7		X	X					*															
Design Rules	startAddress : Address	8									*														
	matches : Address [ ]	9										*													
	WinerySearchOption	10											*												
	Tour	11												*											
	MapOperation	12													*										
<SM>	WebServicesLogger	13													*										
Application Modules	AddressLocator	14						X		X	X				X	*									
	AuthAddressLocator	15				X		X							X	*									X
	WineryFinder	16							X			X	X		X		*								
	RouteMapHandler	17						X					X	X	X			*							
	AuthRouteMapHandler	18				X		X											X	*					X
Application Controller	< jsp > startWineryFind	19				X				X	X						X				*	X			
	< jsp > searchWinery	20				X				X		X						X			X	*	X		
	< jsp > tour	21				X							X	X						X	X	*	X		
	< jsp > directions	22				X							X								X		*		
	< DD > web.xml	23	X	X	X	X																			*

Conclusion

- NOV can be used as a design evaluation framework
  - Applies equally well in design with Aspects
- Aspects added value to an existing design in our study

- *Open Issues / Future Directions*
  - Rich representation of dependencies
    - Classifying the dependencies
      - method call Vs pointcuts
    - Representing various forms of dependencies
      - Concern Model, non-technical dependencies
  - Applying NOV to larger designs with more plausible and empirically validated heuristics
  - Making sense of the numeric value of NOV
    - What is the 'value' of software design in 'capital markets' ?
      - software interpretation of such economic terms?

Reference

- [BC00] C. Y. Baldwin and K. B. Clark. Design Rules vol I, The Power of Modularity. MIT Press, 2000
  - Standard reference on NOV
  - Uses NOV on System Design (IBM 360 family, UNIX)
- [sul+01] K. J. Sullivan, W. G. Griswold, Y. Cai, and B. Hallen. The structure and value of modularity in software design. ACM SIGSOFT 2001
  - Application studied: KWIC (KeyWords In Context)
  - Simple (and first) heuristics for calculating the Technical Potential
    - Introduced Environmental Parameters