1st International Workshop on **Software Architecture Metrics** (SAM2014)

Welcome Address

Organizer Introduction

- Heiko Koziolek, ABB Corporate Research, DE
- Robert L. Nord, Software Engineering Institute, US
- Ipek Ozkaya, Software Engineering Institute, US
- Paris Avgeriou, University of Groningen, NL

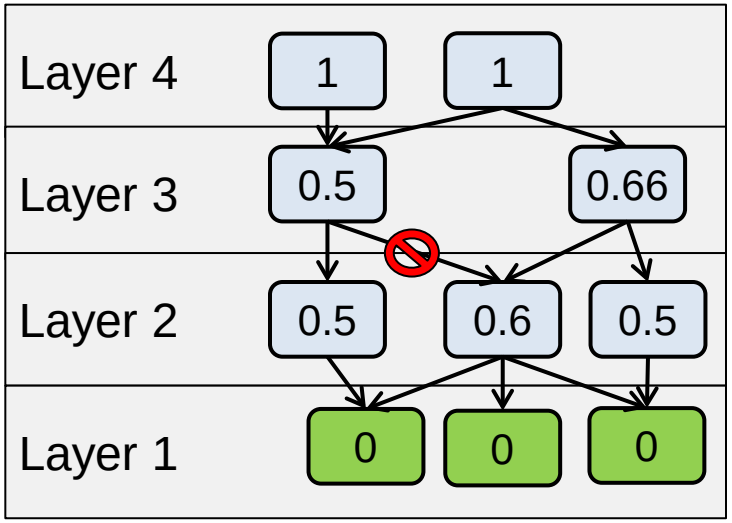
Software Architecture Metrics

- Software Architecture:
  „Fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution” [IEEE42010, 2012]
- Software Quality Metric:
  „A function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which software possesses a given attribute that affects its quality” [IEEE1061, 1998]
- Software Architecture Metric:
  „A software quality metric concerning software architecture. A software architecture metric quantifies architecture quality, value and cost” [Us :-)]

Artifacts providing Input for Software Architecture Metrics

- Architecture artifacts
  - Informal architectural documentation
  - Architecture models and views
    - Component / Connector
    - Knowledge / Decision
- Intermediate artifacts
  - Source code, Byte code
  - Prototypes (e.g., measure architecture qualities)
- Trace links between architecture and intermediate artifacts (e.g., requirements, code, test cases)
- …

Example: Module Interaction Stabilility



- Characterizes software according to the principle of Maximization of Stand-Alone Extensibility
- Promotes the use of stable modules in lower layers

Instability of a module

Modules m depends on

$$\mathcal{I}(m) = \frac{|fanout(m)|}{|fanin(m)| + |fanout(m)|}$$

Modules that depend on m

$$SD(m) = \{m_i \in fanout(m) \mid \mathcal{I}(m) > \mathcal{I}(m_i) \;\&\; \mathcal{L}(m) \geq \mathcal{L}(m_i)\}$$

Set of stable dependencies to lower layers

$$MISI(m) = \frac{|SD(m)|}{|fanout(m)|}$$

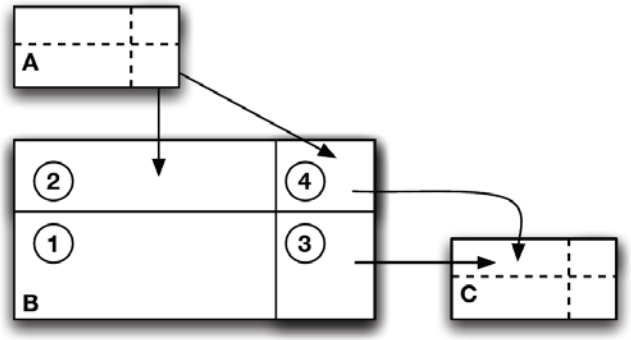
$$= 1 \text{ when } fanout(m) = \emptyset,$$

$$MISI(\mathcal{S}) = \frac{1}{M} \sum_{i=1}^{M} MISI(m_i).$$

For all modules

S. Sarkar, G. M. Rama, and A. C. Kak, *"API-based and information-theoretic metrics for measuring the quality of software modularization,"* IEEE Trans. Softw. Eng., vol. 33, pp. 14–32, January 2007.

Example: Dependency Profiles

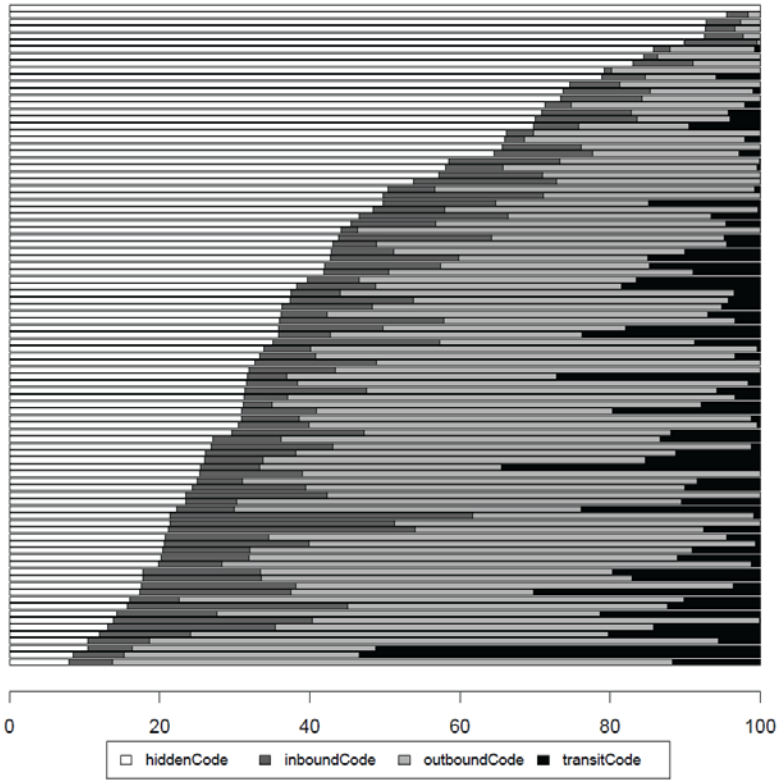


1 = hidden modules

2 = inbound modules

3 = outbound modules

4 = transit modules



Dependency Profiles
for 95 industrial and open source systems

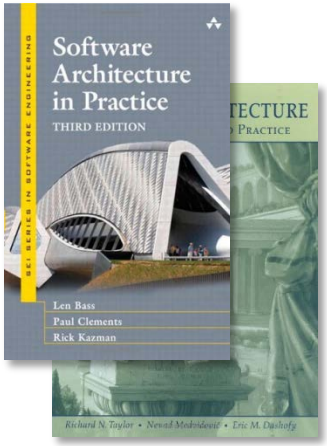
Bouwers, E.; van Deursen, A.; Visser, J., "Dependency profiles for software architecture evaluations,"
Proc. 27th IEEE International Conference on Software Maintenance (ICSM 2011), pp.540,543, 25-30 Sept. 2011

Potential Uses of Architecture Metrics

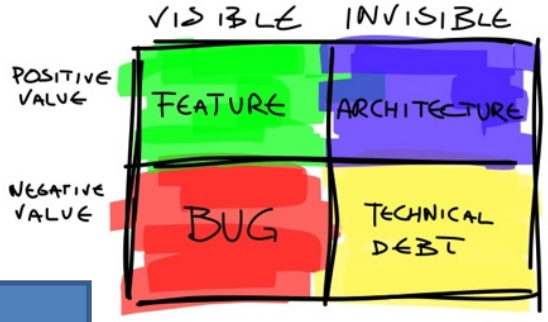
- Assess achievement of quality attributes
- Detect architecture erosion early
- Balance quality attribute trade-offs
- Make informed decisions on improvements
- Identify trends and react appropriately
- Support project planning
- Support business cases
- Conduct cost-benefit analysis
- Support risk management
- Select among design alternatives

Related Fields

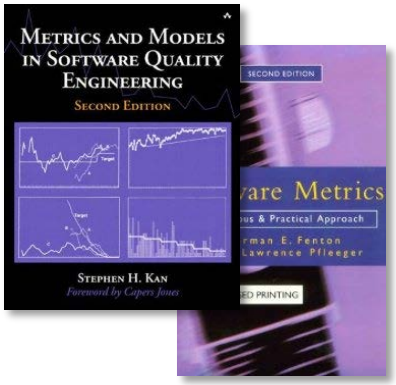
Software Analytics



Software Architecture



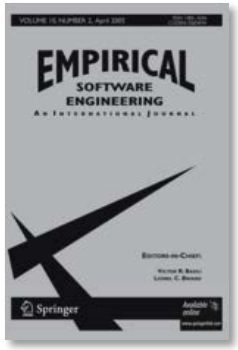
Technical Debt



Software Metrics

Software Quality

Software Maintainance & Evolution



Empirical SW Engineering

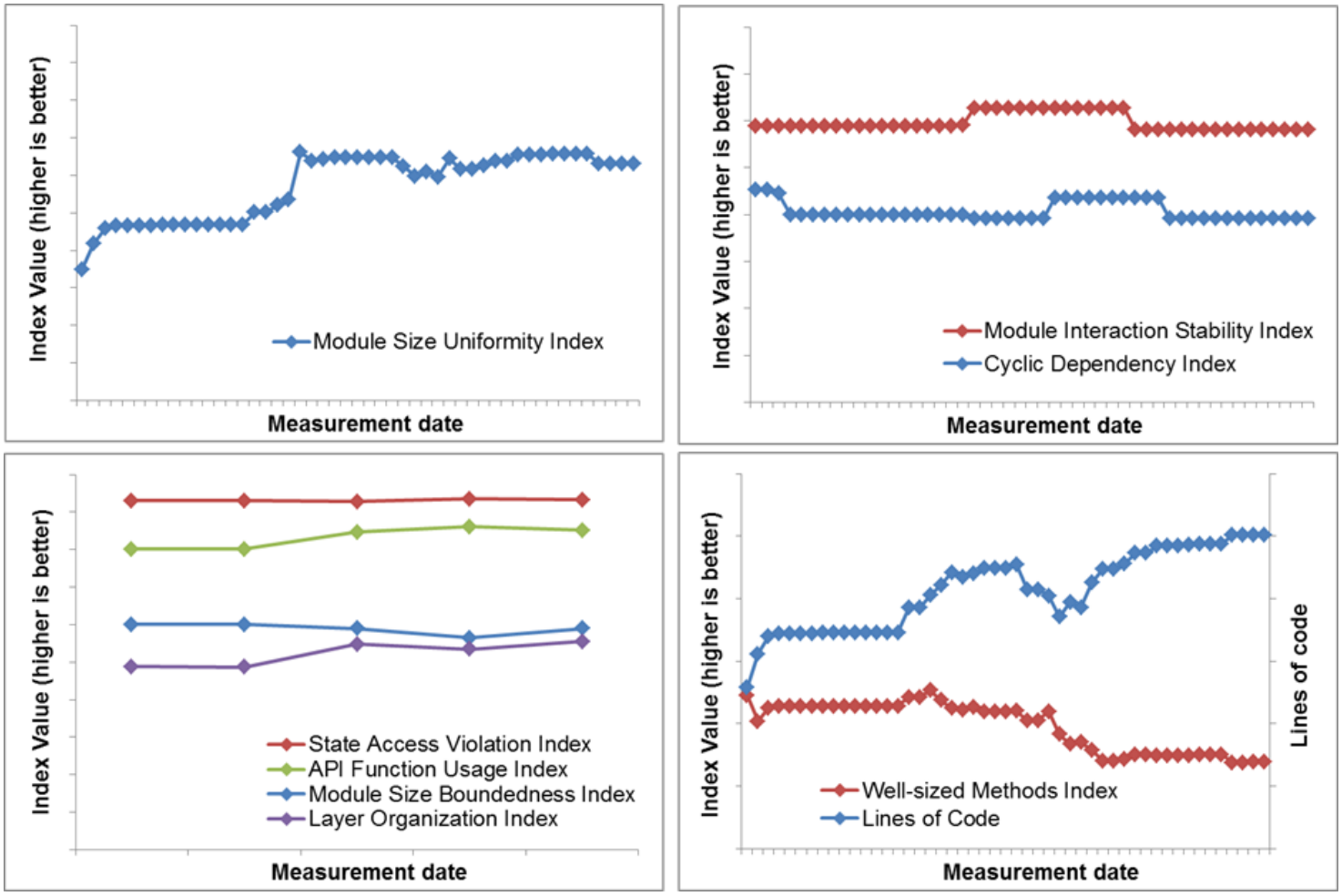
Qualitative Methods

Architecture Metrics from Source Code

#	Source	Abbr.	Name	Description	Required Input	Tool
Similarity of Purpose						
M7	Sarkar2007	CDM	Concept Domination Metric	Non-uniformity of the distribution of concepts	List of concepts, frequency of occurrences per mod.	Proprietary
M7	Sarkar2007	CCM	Concept Coherency Metic	Amount of mutual information between mod./concept	List of concepts, entropy for concepts	Proprietary
M7	Sarkar2007	APIU	API Function Usage Index	Percentage of API functions used by other modules	API definition, # calls to API	Proprietary
M6	Sant'anna2007	CDAC	Concern Diffusion over Arch. Components	Counts the components realizing an arch. concern	Mapping of components to architectural concerns	
M11	Sethi2009	CS	Concern Scope	Amount of design decisions influenced by a concern	Design decisions, concerns	
M11	Sethi2009	CO	Concern Overlap	Amount of design decisions infl. by multiple concerns	Design decisions, concerns	
Encapsulation						
M1	Briand1996	RCI	Ratio of Cohesive Interactions	Ratio of potential/known data declarations interactions	Module dependencies	
M1	Briand1996	IC	Import Coupling	Extend to which a module depends on externals	# imports per module	
M1	Briand1996	EC	Export Coupling	Interactions between internal/external data decl.	Module dependencies	
M3	Mancoridis1998	MQ	Modularization Quality	Diff. of inter- and intra-connectivity of subsystems	Module dependency graph, clusters	Bunch
M5	Martin2003	Ca	Afferent Couplings	# packages depending on classes in a package	Class dependencies	JDepend
M5	Martin2003	Ce	Efferent Couplings	# packages the classes of a package depend on	Class dependencies	JDepend
M6	Sant'anna2007	CLIC	Comp.-level Interlacing Betw. Concerns	Counts components sharing concerns	Mapping of components to architectural concerns	
M6	Sant'anna2007	LCC	Lack of Concern-based Cohesion	Counts the number of concerns by a component	Mapping of components to architectural concerns	
M7	Sarkar2007	MII	Module Interaction Index	Percentage of calls routed through APIs	Module and API definition	Proprietary
M7	Sarkar2007	NC	Non-API Function Closedness Index	Percentage of functions classified API or non-API	API definition	Proprietary
M7	Sarkar2007	IDI	Implicit Dependency Index	Percentage of explicit module dependencies	# Implicit module dep. (e.g., global variables, files)	Proprietary
M8	Sarkar2008	BCFI	Base class fragility index	Extent of base-class fragility in the system	Classes, ancestors, inherited methods, depend.	Proprietary
M8	Sarkar2008	IC	Inheritance-based intermodule coupling	Fraction of classes in other mod. defined by inherit.	Module definition, inheritance dependencies	Proprietary
M8	Sarkar2008	NPII	Not-programming-to-interfaces Index	Percentage of calls to to root interfaces	Interface definitions, call dependencies	Proprietary
M8	Sarkar2008	AC	Association-induced coupling	Percentage of class associations to other modules	Module definition, associations	Proprietary
M8	Sarkar2008	SAVI	State Access Violation Index	Extend of intermodule access to internal state	Module definition, state accesses	Proprietary
M10	Anan2009	IEAS	Entropy of an architectural slicing	Amount of information encoded in a arch. layer	Mapping of modules to layers, dependency graph	
M10	Anan2009	ASC	Architecture Slicing Cohesion	Ratio of intra- and intermodule coupling	Mapping of modules to layers, dependency graph	
M11	Sethi2009	DV	Decision Volatility	Stability of a decision decision ag. ext. influences	Design decisions, env. impact, impact scope	
Compilability, Extendibility, Testability						
M2	Lakos1996	CCD	Cumulative Component Dependency	Sum of component dependencies in a subsystem	Component dependency graph for a subsystem	SonarJ
M2	Lakos1996	ACD	Average Cumulative Comp. Dependency	CCD divided by components in subsystem	Component dependency graph for a subsystem	SonarJ
M2	Lakos1996	NCCD	Normalized Cumulative Comp.	CCD divided by CCD of a binary dependency tree	Component dependency graph for a subsystem	SonarJ
M4	Allen2001	COUM	Coupling of a module	Amount of information in intermodule-edges graphs	Module dependency graph	
M4	Allen2001	ICM	Intramodule coupling of a module	Amount of information in intramodule-edges graph	Module dependency graph	
M4	Allen2001	COHM	Cohesion of a module	Amount of information in intramodule coupling	Module dependency graph	
M5	Martin2003	A	Abstractness	Ratio of abstract classes to total classes in package	Class definitions in a package	JDepend
M5	Martin2003	I	Instability	Ratio of efferent to total coupling [I=Ce/(Ce+Ca)]	Class dependencies	JDepend
M5	Martin2003	DMS	Distance from the Main Sequence	Perpendicular dist. of a package from the line A + I = 1	Abstractness and Instability	JDepend
M7	Sarkar2007	MISI	Module Interaction Stability Index	Percentage of module depending on stable layers	Mapping of modules to layers, fan-in, fan-out	Proprietary
M7	Sarkar2007	NTDM	Normalized Testability Dependency Metric	Percentage of module independent testing	Test dependencies between modules	Proprietary
M8	Sarkar2008	PPI	Plugin Polution Index	Amount of superfluous code in a plugin module	Extension API, abstract methods in plugins	Proprietary
M11	Sethi2009	CI	Change impact	Amount of design decisions changed during evolution	Design decisions, evolution scenario	
M11	Sethi2009	IL	Independence Level	System perc. changeable under stable design rules	Independent module set in augmented constr. netw.	
Acyclic Dependencies						
M5	Martin2003	PDC	Package Dependency Cycles	Cyclic dependencies between packages	Package dependency graph	JDepend
M7	Sarkar2007	Cyclic	Cyclic Dependencies Index	Extent of cyclic dependencies between modules	Module dependency graph	Proprietary
M7	Sarkar2007	LOI	Layer Organization Index	Cyclic dependencies between layers	Mapping of modules to layers, dependency graph	Proprietary
M9	Sangwan2008	XS	Excessive Structural Complexity	Cyclic dependencies violation times amount of dep.	Module dependency graph	Structure101
Size						
M7	Sarkar2007	MSBI	Module Size Boundness Index	Deviation of module sizes from a threshold	Lines of code per module, optimal module size	SourceMonitor
M7	Sarkar2007	MSUI	Module Size Uniformity Index	Distribution of module sizes	Lines of code per module	SourceMonitor

Heiko Koziolek. *Sustainability evaluation of software architectures: A systematic review*.
In Proc. 7th Int. ACM/SIGSOFT Conf. on the Quality of Software Architectures (QoSA'11). ACM, June 2011.

Architecture Metric Trends at ABB

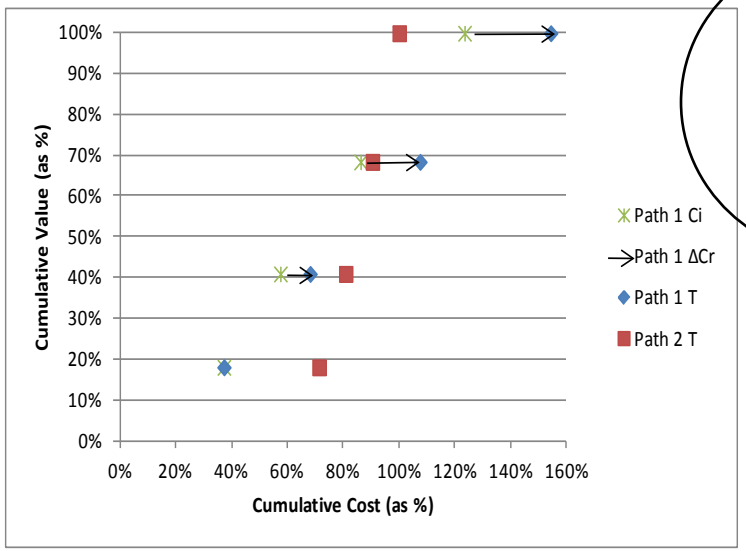


Heiko Koziolek, Dominik Domis, Thomas Goldschmidt, and Philipp Vorst. *Measuring architecture sustainability.* IEEE Software, 30(6):54-62, November 2013.

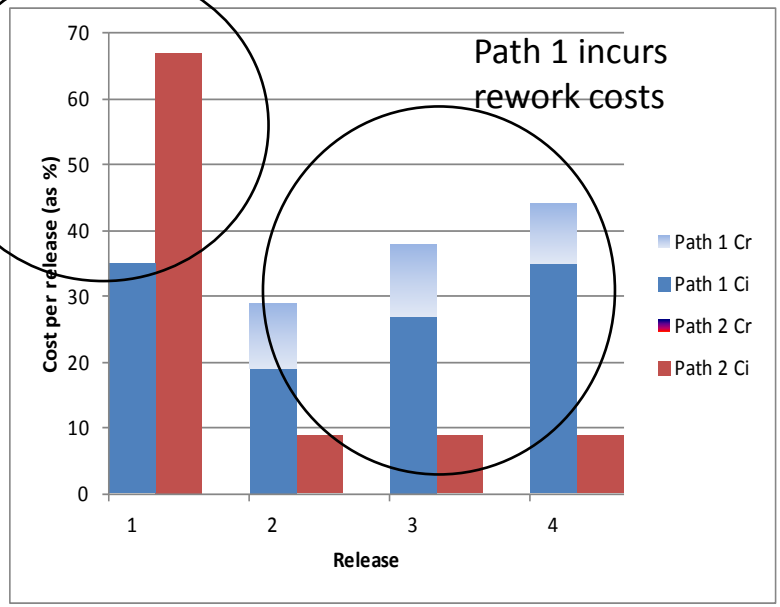
Dependency Analysis of Release Paths

Path 1:
value focused;
functionality first.

Path 2: *cost focused;*
architecture push.



Initial release architecting cost for Path 2



		Release 1	Release 2	Release 3	Release 4
Path #1	Cumulative value	36	81	135	197
	% of total value	18%	41%	68%	100%
	Cost (Ci + Cr)	35	64	101	145
	% of total implementation cost	37%	68%	108%	155%
Path #2	Cumulative value	36	81	135	197
	% of total value	18%	41%	68%	100%
	Cost (Ci + Cr)	67	76	85	94
	% of total implementation cost	71%	81%	90%	100%

Robert L. Nord, Ipek Ozkaya, Philippe Kruchten, Marco Gonzalez. *In Search of a Metric for Managing Architectural Debt*. Joint 10th Working IEEE/IFIP Conference on Software Architecture (WICSA) & 6th European Conference on Software Architecture (ECSA), August 20-24, 2012, Helsinki, Finland.

Quantifying Architecture Quality

Challenges

- Insufficient and unproven metrics for quantifying architecture quality to guide the re-architecting process.
- Code-level refactoring techniques do not scale effectively to support architecture-level evaluation for re-architecting.

There has been an increasing focus on tools for the purpose of structural analysis.

- increasing sophistication,
- support for some structural analysis in addition to code analysis,
- first steps towards analyzing financial impact by relating structure analysis to cost and effort for rework.

D. Falessi, P. Kruchten, R. Nord, and I. Ozkaya. *Technical Debt at the Crossroads of Research and Practice: Report on the Fifth International Workshop on Managing Technical Debt*, ACM SIGSOFT Software Engineering Notes, Volume 39, Issue 2, March 2014 (to appear).

Open Research Questions

- Which software architecture metrics are useful?
  - quality, value, costs, uncertainty
  - understandability, maintainability, evolvability, concern dispersion, modularization
- How to calculate software architecture metrics?
  - directly from architecture or from other artifacts
  - what can be measured in models?
  - tool support / integration with design and code tools
- How to validate software architecture metrics?
  - tools and techniques
  - empirical evidence
- How to use software architecture metrics?
  - visualization
  - decision support
  - project and business planning

Program today

- 1:30 pm
  - Welcome Address (Robert Nord, Heiko Koziolek)
  - [Invited Talk] Eric Harper: *Industry Perspectives on Requirements for and Value of Software Architecture Metrics*
  - [Research paper] Muhammad Atif Javed: *Empirical Evaluation of the Understandability of Architectural Component Diagrams*
  - [Position paper] Stephan Sehestedt: *Toward Quantitative Metrics for Architecture Models*
- 3:00 pm coffee break
- 3:30 pm
  - [Invited Talk] Jean-Guy Schneider: *On the Challenges in Extracting Metrics from Java Bytecode*
  - Brainwriting: collecting research questions and ideas from all participants
  - Summary & Wrap-up
- 5:15 pm workshop end
- 7:00 pm dinner

Quick round among all participants

- Please introduce yourself and state your personal expectations for the workshop!
- …

Thank you!

- Authors
- Program Committee
- Attendees
- Sponsors