Bibliography

Abbattista F, Lanubile F, Mastelloni G, Visaggio G (1994) An experiment on the effect of design recording on impact analysis. In: Müller A, Georges A (eds) Proceedings of the International Conference on Software Maintenance. IEEE Computer Society, pp 253-259

Agrawal H, Horgan JR (1990) Dynamic program slicing. In: Proceedings of the ACM SIGPLAN 1990 Conference on Programming Language Design and Implementation, White Plains, New York, pp 246-256

Alexander C, Ishikawa S, Silverstein M, King I, Angel S, Jacobson M (1977) A Pattern Language: Towns, Buildings, Construction, Oxford University Press, Oxford

Alford, MW, Lawson JT (1979) Software Requirements Engineering Methodology (Development). RADC-TR-79-168, US Air Force, Rome Air Development Center, Griffiss AFB, New York, NY

Ali-Babar M, Gorton I, Kitchenham BA (2006) A framework for supporting architecture knowledge and rationale management. In: Dutoit AH, McCall R, Mistrík I, Paech B (eds) Rationale Management in Software Engineering. Springer, pp 237-254

Ali-Babar M, Gorton I (2007) A tool for managing software architecture knowledge, In: the Workshop on the Sharing and Reusing Architectural Knowledge at the International Conference on Software Engineering, Minnesota.

Ambler SW (1998) Software Process Patterns, Cambridge University Press and Reasoning (KR2002). pp 375-384

Andréka H, Ryan M, Schobbens PY (2002) Operators and laws for combining preference relations. J Logic Comput 12(1):13-53

ANSI/IEEE (1987) ANSI/IEEE Standard for Software Unit Testing, ANSI/IEEE Std 1008-1987

Antón AI, Dempster JH, Siege DF (2000) Deriving goals from a use case based requirements specification for an electronic commerce system. In: Proceedings of the Sixth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ), Stockholm, Sweden, pp 10-19

Antón AI, Potts C (2001) Functional paleontology: system evolution as the user sees it. In: Proceedings of the 23rd International Conference on Software Engineering, Toronto, Canada, pp 421-430

Antón, AI, Potts C (1998) The use of goals to surface requirements for evolving systems. In: Proceedings of the 20th international Conference on Software Engineering. pp 157-166.

Antoniol G, Canfora G, de Lucia A, Casazza G (2000) Information retrieval models for recovering traceability links between code and documentation. In: Proceedings of the International Conference on Software Maintenance. pp 40-51

Arrow KJ (1963) Social Choice and Individual Values, 2nd edn, Wiley, New York

Avellis G, Borzacchini L, Cavallo A, Cotugno P, De Mastro G (1993) A blackboard architecture for intelligent assistance in software maintenance. In: Proceedings of the 6th International Workshop on Computer-Aided Software Engineering, Singapore, pp 180-189

Baker ER (2001) Which way, SQA?. IEEE Softw 18(1):16-18

Bañares-Alcántara R, King MP, Ballinger G (1995) Egide: a design support system for conceptual chemical process design. In: AI System Support for Conceptual Design: Proc. of the 1995 Lancaster International Workshop on Engineering Design, Springer-Verlag, New York, pp 138-152

Baniassad ELA, Murphy GC, Schwanninger C (2003) Design pattern rationale graphs: Linking design to source. In: Proceedings of the 25th International Conference on Software Engineering (ICSE 2005), May 3–10, pp 352–362

Bass L, Clements P, Kazman R (2003) Software Architecture in Practice, 2nd edn. Addison-Wesley, NY

Bass L, Clements P, Nord RL, Stafford J (2006) Capturing and using rationale for a software architecture. In: Dutoit AH, McCall R, Mistrík I, Paech B (eds) Rationale Management in Software Engineering, Springer, pp 255-272

Beck K (1999) Extreme Programming Explained: Embrace Change, Addison-Wesley, NY

Beck K (2002) Test-Driven Development: By Example, Addison-Wesley, NY

Beck K, Andres C (2005) Extreme Programming Explained: Embrace Change, 2nd edn, Addison Wesley, Boston

Bennett KH, Rajlich VT (2000) Software maintenance and evolution: A roadmap. In: Finkelstein, A (ed) The Future of Software Engineering, 22nd ICSE. Limerick, Ireland, pp 73-87

Bertolino A (2007) Software testing research: achievements, challenges, dreams. In: Proceedings of the 29th International Conference on Software Engineering, Minneapolis, Minnesota, pp 85-103

Biggerstaff TJ, Mitbander BG, Webster D (1993) The concept assignment problem in program understanding. In: Proceedings of the 15th International Conference on Software Engineering. Baltimore, Maryland, pp 482-498

Blackorby C, Donaldson D, Mongin P (2000) Social aggregation without the expected-utility hypothesis. UBC Department of Economics Discussion Paper no. 00-18

Blevins D (2001) Overview of the Enterprise JavaBeans component model. In: Councill B, Heineman G (eds) Component-based Software Engineering, Addison-Wesley, Upper Saddle River, pp 589-606

Boehm B (1979) Software engineering: R&D trends and defence needs. Research Direction in Software Technology, Wegner P. (ed) MIT Press, Cambridge MA, pp 1-9

Boehm B (1986) A spiral model of software development and enhancement. In: ACM SIGSOFT Software Engineering Notes. 11(4):22-42

Boehm B (2006a) A view of 20th and 21st century software engineering. In: Proceedings of the 28th international Conference on Software Engineering, Shanghai, China, pp 12-29

Boehm B (2006b) Value-based software engineering: overview and agenda. In: Biffl S, Arum A, Boehm B, Erdogmus H, Grunbacher P (eds) Value-Based Software Engineering, Springer, pp 3-14

Boehm B, Bose P (1994) A collaborative spiral software process model based on Theory W. In: Proceedings of the 3rd International Conference on the Software Process, Reston, VA, pp 59-68

Boehm B, Bose P, Horowitz E, Lee MJ (1995) Software requirements negotiation and renegotiation aids. In: Proceedings of the 17th international Conference on Software Engineering. Seattle,Washington, pp 128-142

Boehm B, Brown J, Kaspar H, Lipow M, MacLeod G, Merrit M (1979) Characteristics of Software Quality. TRW Series of Software Technology vol 1. North-Holland

Boehm B, Egyed A (1998) Software requirements negotiation: some lessons learned. In: Proceedings of the 20th international Conference on Software Engineering. Kyoto, Japan, IEEE Computer Society, pp 503-506

Boehm B, In H (1996) Identifying quality-requirement conflicts. IEEE Software. 13(2):25-35

Boehm B, Kitapci H (2006) The WinWin approach: using a requirements negotiation tool for rationale capture and use. In: Dutoit A, McCall R, Mistrík I, Paech B (eds) Rationale Management in Software Engineering, Springer, pp 173-190

Boehm B, Ross R (1989) Theory W software project management: principles and examples. IEEE Transactions on Software Engineering. 15(7):902-916

Boehm BW, Brown JR, Lipow M (1976) Quantitative evaluation of software quality. In: Proceedings of the 2nd International Conference on Software Engineering. San Francisco, California, pp 592-605

Boehm B, Bose P, Horowitz E, Lee MJ (1995) Software requirements negotiation and renegotiation aids: A Theory-W based spiral approach. In: Proceedings ICSE-17 (International Conference on Software Engineering), pp 243-253 IEEE Computer Society

Bohner SA, Arnold RS (1996) An introduction to software change impact analysis. In: Bohner SA, ArnoldRS (eds) Software Change Impact Analysis. IEEE Computer Society, pp 1-28

Booth R (2002) Social contraction and belief negotiation. In: Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002), pp 375-384

Bosch J (2004) Software architecture: The next step. Proc. 1st European Workshop Software Architecture (EWSA 04), Springer, pp 194-199

Bose P (1995) A model for decision maintenance in the WinWin collaboration framework. In: Proceedings of the 10th Conference on Knowledge Based Software Engineering. Boston, MA, pp 105-113

Bose P (1998) Change analysis in an architectural model: A design rationale based approach. In: Proceedings ISAW3 (International Software Architecture Workshop), Orlando, Florida, pp 5-8

Bozheva T, Gallo M (2006) Defining Agile Patterns. In: Dutoit A, McCall R, Mistrík I, Paech B (eds) Rationale Management in Software Engineering, Springer, pp 373-390

Brooks FP (1995) The mythical man-month. Anniversary Edition, 2nd edn Reading, MA: Addison-Wesley

Bruegge B, Dutoit AH (2004) Object-Oriented Software Engineering Using UML, Patterns, and Java. 2nd edn. Prentice Hall, NJ

Bruegge B, Dutoit A (2000) Object-Oriented Software Engineering: Conquering Complex and Changing Systems, Prentice Hall, NJ

Buckingham Shum S (2007), Hypermedia discourse: Contesting networks of ideas and arguments, Keynote Address, In: Proceedings of 15th International Conference on Conceptual Structures, Sheffield, UK, July 2007. Lect Notes Comput Sci. 4604:29-44.

Buckingham Shum S, Hammond N (1994) Argumentation-based design rationale: What use at what cost? Int J of Human–Computer Studies, 40(4), 603-652

Buckingham Shum S, MacLean A, Bellotti VME, Hammond NV (1997) Graphical Argumentation and Design Cognition, Human-Comput Interact. 12(3):267-300

Buckingham Shum S, Selvin AM, Sierhuis M, Conklin EJ, Haley CB, Nuseibeh B (2006) Hypermedia support for argumentation-based rationale: 15 years on from gIBIS and QOC. In: AH, McCall R, Mistrik I, Paech B (eds) Rationale Management in Software Engineering, Springer-Verlag: Berlin, pp 111-132

Budgen D (2003) Software design, 2nd edn Addison-Wesley, Harlow, England

Burge J, Brown DC (2002) Integrating design rationale with a process model, In: Workshop on Design Process Modeling. Artificial Intelligence in Design ‘02. Cambridge, UK

Burge JE (2005) Software Engineering Using design RATionale. Ph.D. thesis, Worcester Polytechnic Institute

Burge JE, Brown DC (2000) Inferencing over design rationale. In: Artificial Intelligence in Design ‘00, Gero J (ed) Kluwer Academic, pp 611-629

Burge JE, Brown DC (2006) Rationale-based support for software maintenance. In: Dutoit AH, McCall R, Mistrik I, Paech B (eds) Rationale Management in Software Engineering, Springer, Germany, pp 273-296

Burge JE, Brown DC (2007) Supporting requirements traceability with rationale, GTC’07: International Symposium on Grand Challenges in Traceability, March 2007, Slade, KY

Burge JE, Brown DC (2003) Rationale support for maintenance of large scale systems. In: Proceedings of the Workshop on Evolution of Large-Scale Industrial Software Applications (ELISA), ICSM ‘03, Amsterdam, Netherlands

Burge JE, Brown DC (2004) An integrated approach for software design checking using rationale. In: Design Computing and Cognition ‘04, Gero J (ed) Kluwer Academic, pp 557-576

Burge JE, Cross V, Kiper J, Maynard-Zhang P, Cornford S (2006) Enhanced design checking involving constraints, collaboration, and assumptions. In: Gero J (ed) Proceedings of the Conference on Design, Computing, and Cognition. Eindhoven Netherlands, pp 655-674

Burnstein I (2003). Practical Software Testing, Springer Professional Computing

Canfora G, Casazza G, De Lucia A (2000) A Design rationale based environment for cooperative maintenance. Int J Soft Eng Know Eng 10(5):627-645

Canfora G, Cerulo L (2006) Fine grained indexing of software repositories to support impact analysis. In: Proceedings of the 2006 International Workshop on Mining Software Repositories, Shanghai, China, pp 105-111

Capilla R, Nava F, Duenas JC (2007) Modeling and documenting the evolution of architectural design decisions. In: the Proceedings of the Workshop on Sharing and Reusing Archtectural Knowledge at the International Conference of Software Engineering (ICSE), Minneapolis, Minnesota

Capilla R, Nava F, Pérez S, Dueñas JC (2006) A web-based tool for managing architectural design decisions. In: Proceedings of the Workshop on Sharing and Reusing Architectural Knowledge. ACM Digital Library. Softw Eng Notes 31 (5)

Carroll J (2000) Making use: scenario-based design of human–computer interaction, MIT press, Cambridge, MA

Carroll J, Rosson MB (1996) Deliberated evolution: stalking the View Matcher in design space. In: Moran TP, Carroll JM (eds) Design rationale: Concepts, techniques, and use. Lawrence Erlbaum, Mahwah, NJ, pp 107-145

Carroll JM, Rosson M (1992) Getting around the task-artifact cycle: how to make claims and design by scenario. ACM Trans Infor Syst 10(2):181-212

Carroll JM, Rosson MB, Chin G Jr, Koenemann J (1998) Requirements development in scenario-based design. IEEE Trans Soft Eng. 24(12):1156-1170

Carroll JM, Mack, RL (1985) Metaphor, computing systems, and active learning. Int J of Man-Machine Stud, 22:39-58

Carroll JM (1995) Scenario-based design: Envisioning work and technology in system development. New York: Wiley

Carroll JM, Alpert SR, Karat J, Van Deusen, MD, Rosson MB (1994) Capturing design history and rationale in multimedia narratives. In: Proceedings of CHI'94: Human Factors in Computing Systems. Boston. New York: ACM Press/Addison-Wesley, pp 192-197

Carroll JM, Rosson MB, Convertino G, Ganoe C (2006) Awareness and teamwork in computer-supported collaborations. Interac Comput, 18: 21-46

Chaib-Draa B, Dignum F (2002) Trends in agent communications language. Comput Intel, 18(2):89-101

Chapin N (2000) Software maintenance types—a fresh view. In: Proceedings of the International Conference on Software Maintenance, San Jose, CA, pp 247-252

Charette RN (1996) Large-scale project management is risk management. IEEE Soft 13(4): 110-117

Chaudron MRV, Groote JF, van Hee KM, Hemerik C, Somers LJAM, Verhoeff T (2004) Software Engineering Reference Framework. Technical Report CS-Report 04-039, Computer Science Reports, Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands

Chernak Y (2001) Validating and improving test-case effectiveness. IEEE Soft 18(1):81-86

Chewar CM, Bachetti E, McCrickard DS, Booker J (2005) Automating a design reuse facility with critical parameters: lessons learned in developing the LINK-UP System. In: Jacob R, Limbourg Q, Vanderderonckt J (eds) Computer-Aided Design of User Interfaces IV. Kluwer Academic, pp 235-246

Chung L, Nixon B, Yu E (1996) Dealing with change: an approach using non-functional requirements. Requirements Eng 1(4):238-260

Chung L, Nixon BA (1995) Dealing with non functional requirements: three experimental studies of a process-oriented approach. In: Proceedings of the 17th International Conference on Software Engineering, pp 25-37

Chung L, Nixon BA, Yu E, Mylopoulos J (2000) Non-functional requirements in software engineering, Kluwer Academic

Chung L, Yu E (1998) Achieving system-wide architectural qualities. In: Proceedings of the OMG-DARPAMCC Workshop on Compositional Software Architectures

Cimitile A, Lanubile F, Visaggio G (1992) Traceability based on design decisions. In: Proceedings of the Conference on Software Maintenance, Orlando, Florida, pp 309-317

Clapp J (1993) Getting started on software metrics. IEEE Soft 10(1):189-109, 117

Clark A (1987) From folk psychology to naïve psychology. Cogn Sci 11:139-154

Cleland-Huang J, Settimi R, BenKhadra O, Berezhan E, Christina S (2005) Goal centric traceability for managing non-functional requirements. In: Proceedings of the International Conference on Software Engineering, St Louis, pp 362-371

Clemen RT, Reilly T (2001) Making Hard Decisions. Duxbury, Forest Grove, CA

Clemen RT, Winkler RL (1999) Combining probability distributions from experts in risk analysis, Risk Analy 19:187-203

Clements P, Bachmann, Bass L, Garlan, Ivers J, Little R, Nord R, Stafford J (2002) Documenting software architecture: Views and beyond, Addison-Wesley

Clements P, Northrop L (2002) Software Product Lines Practices and Patterns, Addison-Wesley

CMMI Product Team (2006) CMMI For Development. Version 1.2. CMU/SEI-2006-TR-008

CMU (2002) Quality measures taxonomy http://www.sei.cmu.edu/str/taxonomies/view_qm.html

Coleman JS (1990) The Foundations of Social Theory. Cambridge, MA: Harvard University Press

Conklin EJ, Burgess-Yakemovic KC (1996) A process-oriented approach to design rationale. In: Moran T, Carroll J (eds) Design Rationale Concepts, Techniques, and Use. Lawrence Erlbaum Associates, pp 393-427

Conklin J, Begeman M (1988) gIBIS: A hypertext tool for exploratory policy discussion, ACM Trans on Office Inform Syst, 6(4):303-331

Conklin J, Burgess-Yakemovic K (1995) A process-oriented approach to design rationale. In: Design rationale concepts, techniques, and use, Moran T, Carroll J (eds). Lawrence Erlbaum Associates, pp. 293-428

Conklin J, Burgess-Yakemovic K (1991) A process-oriented approach to design rationale, Human-Computer Interaction, 6: 357-291

Connolly T, Jessup L, Valacich J (1990) Effects of anonymity and evaluative tone on idea generation in computer-mediated groups. Manage Sci 36(6): 689-703

Councill B, Heineman G (2001) Definition of a software component and its elements. In: Councill B, Heineman G (eds) Component-Based Software Engineering, Addison-Wesley, Upper Saddle River, pp 5-20

Crosby ME, Scholtz J, Wiedenbeck S (2002) The roles beacons play in comprehension for novice and expert programmers. In: Kuljis K, Baldwin L, Scoble R (eds) Proceedings of the Psychology of Programming Interest Group, Brunel University, pp 58-73

Cross N (2003) Evidence from protocol and other formal studies of design activity. In: Eastman C, McCracken M, Newstetter W (eds), Knowing and Learning to Design: Cognitive Perspectives in Design Education, Amsterdam: Elsevier

Curtis B, Krasner H, Iscoe N (1988) A field study of the software design process for large systems. Commun ACM 31(11):1268-1287

Cysneiros LM, Leite JCSP (2001) Using UML to reflect non-functional requirements. In: Proceedings of the 11th CASCON, November, IBM Canada, Toronto, pp 202-216

Cysneiros LM, Leite JCSP (2004) Nonfunctional requirements: from elicitation to conceptual models. IEEE Trans on Softw Eng, 30(5):328-350

Darimont R, Delor E, Massonet P, van Lamsweerde A (1997) GRAIL/KAOS: an environment for goal-driven requirements engineering. In: Proceedings of the 19th International Conference on Software Engineering. ACM, New York, NY, pp 612-613

de Boer RC, Farenhorst R, Clerc V, van der Ven JS, Lago P, van Vliet H (2006) A model for structuring software architecture project memories, In: Proceedings of the 8th International Workshop on Learning Software Organizations

De Grace P, Stahl LH (1998) Wicked problems, righteous solutions: a catalogue of software engineering paradigms, Yourdon

de Kleer J (1986) An assumption-based truth maintenance system. Artif Intell 28(2): 127-162

de la Garza J, Alcantara P (1997) Using parameter dependency network to represent design rationale. J Comput Civil Eng 2(2):102-112

DeMarco T, Lister TR (1999) Peopleware: Productive projects and teams, 2nd edn, New York: Dorset House

Demeyer S, Ducasse S, Nierstrasz O (2003) Object-Oriented Reengineering Patterns, Morgan Kaufmann

Department of Defense (2002) Mandatory Procedures for Major Defense Acquisition Programs (MDAPS) and Major Automated Information System (MAIS), DoD 5000.2-R

Devanbu P, Brachman R, Selfridge P, Ballard B (1991) Lassie: a knowledge-based software information system. In: Commun ACM 34(5):34-49

Dick J (2005) Design traceability. IEEE Softw 22(6):14-16

Dijkstra EW (1972) The humble programmer. Communications of the ACM 15(10):859-866

Dijsktra EW (1989) On the cruelty of really teaching computer science. Commun ACM 32(12):1398-1404

Do H, Rothermel G, Kinneer A (2006) Prioritizing JUnit test cases: An empirical assessment and cost-benefits analysis. Empir Softw Eng 11:33-70

Domeshek E, Kolodner JL (1996) The Designers' Muse: Providing Experience to Aid Conceptual Design of Complex Artifacts. In Maher ML and Pu P (eds), Issues and Applications of Case-Based Reasoning to Design, Lawrence Erlbaum Associates: Mahwah, NJ, pp 11-38

Doyle J (1979) A truth maintenance system. Artif Intell 12(3): 231-272

Druffel LE, Buxton JN (1980) Requirements for an Ada programming support environment: Rationale for Stoneman, In: Proceedings COMPSAC Chicago, pp 66-72

Dutoit AH, Paech B (2000) Supporting Evolution: Using Rationale in Use Case Driven Software Development, Proceedings of the Sixth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'2000), Stockholm, Sweden

Dutoit A, McCall R, Mistrik I, Paech B (eds) (2006a) Rationale Management in Software Engineering, Springer Verlag, Heidelberg

Dutoit A, McCall R, Mistrík I, Paech B (2006b) Rationale management in software engineering: Concepts and techniques. In: Dutoit A, McCall R, Mistrík I, Paech B (eds) Rationale Management in Software Engineering, Springer-Verlag, pp 1-48

Egyed A, Grunbacher P (2004) Identifying requirements conflicts and cooperation: How quality attributes and automated traceability can help. IEEE Software. 21(6):50-58

EIA (1998) Electronic Industries Alliance. Systems Engineering Capability Model (EIA/IS-731). Washington, DC

Eick S (1998) Maintenance of large systems. In: Stasko J, Dominigue J, Brown M, Price B (eds) Software Visualization: Programming as a Multimedia Experience. The MIT Press, pp 315-328

Eickelmann NS, Ruffolo F, Baik J, Anant A (2002) An Empirical Study of Modifying the Fagan Inspection Process and the Resulting Main Effects and Interaction Effects Among Defects Found, Effort Required, Rate of Preparation and Inspection, Number of Team Members and Product 1st Pass Quality. In: Proceedings of the 27th Annual NASA Goddard Software Engineering Workshop (Sew-27'02), pp 58-64

Erdogmus H, Favaro J, Halling M (2006) Valuation of software initiatives under uncertainty: concepts, issues, and techniques. In: Biffl S, Aurum A, Boehm B, Erdogmus H, Grunbacher P (eds) Value-Based Software Engineering, Springer, pp 39-66

Ewald T (2001) Overview of COM+, In: Councill B, Heineman G (eds) Component-based Software Engineering, Addison-Wesley, Upper Saddle River, pp 573-588

Fabian A, Wahid S, Bhatia S, McCrickard DS (2006) Creating an Interactive Learning Environment with Reusable HCI Knowledge. In: Proceedings of the World Conference on Educational Multimedia/Hypermedia and Educational Telecommunications (ED-MEDIA '06), Orlando FL, June, pp 2314-2322

Falessi D, Beker M, Cantone G (2006) Design decision rationale: experiences and steps ahead towards systematic use. In: Proceedings of the Workshop on the Sharing and Reuse of Architectural Knowledge, Torino, Italy

Favaro J (1996) When the pursuit of quality destroys value. IEEE Soft 13(3):93-95

Filman RE (1998) Achieving ilities. In: Proceedings of the Workshop on Compositional Software Architectures, Monterey, CA, USA

Finkelstein A, Kramer J (2000) Software engineering: a roadmap. In: Proceedings of the Conference on the Future of Software Engineering, ICSE 2000, Limerick, Ireland, pp 3-22

Finkelstein AC, Gabbay D, Hunter A, Kramer J, Nuseibeh B (1994) Inconsistency handling in multiperspective specifications. IEEE Trans Softw Eng 20(8):569-578

Fischer G, Morch A (1988) CRACK: A critiquing approach to cooperative kitchen design. In: Proceedings of the International Conference on Intelligent Tutoring Systems (Montreal, Canada), ACM Press, New York, pp 176-185

Fischer G, Henninger S, Redmiles D (1991) Cognitive tools for locating and comprehending software objects for reuse. In: Proceedings of the 13th International Conference on Software Engineering, Austin, Texas, pp 318-328

Fischer G, Lemke A, McCall R, Morch A (1996) Making argumentation serve design. In: Moran T, Carroll J (eds) Design Rationale Concepts, Techniques, and Use, Lawrence Erlbaum Associates, pp 267-294

Fischer G, McCall R, Morch A (1989) JANUS: Integrating hypertext with a knowledge-based design. In: Proceedings of Hypertext '89, pp 105-117

Fisher R, Ury W (1981) Getting to Yes. Houghton-Mifflin

Fitzpatrick G (2003) The locales framework: understanding and designing for wicked problems. Fitzpatrick G, Series: Computer Supported Cooperative Work, vol 1, Springer Verlag, New York

Fowler M, Beck K, Brant J, Opdyke W, Roberts D (1999) Refactoring: Improving the Design of Existing Code, Addison-Wesley

Fowler M (2003) UML distilled, 3rd edn, New York: Addison-Wesley.

Fox J, Das S (2000) Safe and Sound Artificial Intelligence in Hazardous Applications, AAAI Press

France RB, Ghosh S, Dinh-Trong T, Solberg A (2006) Model-driven development using UML 2.0: Promises and pitfalls. Computer 39(2):59-66

Frankl P, Hamlet D, Littlewood B, Strigini L (1997) Choosing a testing method to deliver reliability. In: Proceedings of the 19th International Conference on Software Engineering, Boston, Massachusetts, pp 68-78

Fuggetta A (2000) Software process: a roadmap. In: Proceedings of the Conference on the Future of Software Engineering, Limerick, Ireland, pp 25-34

Gallagher KB, Lyle JR (1991) Using program slicing in software maintenance. In: IEEE Trans Softw Eng 17(8):751-761

Gamma E, Helm R, Johnson R, Vlissides J (1995) Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, MA

Gannod G, Burge J, Urban S (2007) Issues in the design of flexible and dynamic service-oriented systems. In: Proceedings of the International Workshop on Systems Development in SOA Environments. Co-located with ICSE, Minneapolis, MN.

Gibson VR, Senn JA (1989) System structure and software maintenance performance. Commun ACM 32(3):347-358

Gilboa I, Samet D, Schmeidler D (2004) Utilitarian aggregation of beliefs and tastes. J Polit Econ 112(4):932-938

Godfrey M, Tu Q (2002) Tracking structural evolution using origin analysis. In: Proceedings of the International Workshop on Principles of Software Evolution, Orlando, Florida, pp 117-119

Goldenson DR, Gibson DL (2003) Demonstrating the impact and benefits of CMMI: an update and preliminary results. CMU/SEI-2003-SR-009

Gomaa H, Farrukh GA (1998) Composition of software architectures from reusable architecture patterns. In: Proceedings of the 3rd International Workshop on Software Architecture, Orlando, Florida, pp 45-48

Gotel O, Finkelstein A (1994) An analysis of the requirements traceability problem. In: Proceedings of the 1st Int. Conf. on Requirements Engineering, IEEE Computer Society Press, pp 94-101

Gruber TR, Russell DM (1996) Generative design rationale: beyond the record and reply paradigm, In: Moran T, Carroll J (eds) Design Rationale Concepts, Techniques, and Use, Lawrence Erlbaum Associates, Mahwah, NJ, pp 323-349

Grudin J (1988) Why CSCW applications fail: Problems in the design and evaluation of organizational interfaces. Proceedings of the CSCW '88 Conference on Computer-Supported Cooperative Work, ACM, New York, pp 85-93

Grudin J (1996) Evaluating opportunities for design capture In: Moran T, Carroll J (eds) Design Rationale Concepts, Techniques, and Use, Lawrence Erlbaum Associates, Mahwah, NJ, pp 453-470

Grudin J (1994) Groupware and social dynamics: Eight challenges for developers. Commun ACM 37(1):92-105

Grünbacher P, Boehm B (2001) EasyWinWin: a groupware-supported methodology for requirements negotiation. In: Proceedings of the 8th European Software Engineering Conference.Vienna, Austria, ACM Press, pp 320-321

Grundy J, Hosking J, Mugridge R (1998) Inconsistency management for multiple-view software development environments. In: IEEE Transactions on Software Engineering. 24(11):960-981

Guindon R (1990) Knowledge Exploited by Experts During Software System Design, Int J Man-Machine Stud 33: 279-304

Hagge L, Houdek F, Lappe K, Paech B (2006) Using patterns for sharing requirements engineering process rationales. In: Dutoit A, McCall R, Mistrík I, Paech B (eds) Rationale Management in Software Engineering, Springer, pp 409-426

Hall T, Rainer A, Baddoo N, Beecham S (2001) An empirical study of maintenance issues within process improvement programmes in the software industry. In: Proc. of the International Conference on Software Maintenance. Florence, Italy, pp 422-430

Harrold MJ (2000) Testing: a roadmap. In: Proceedings of the Conference on the Future of Software Engineering. Limerick, Ireland, pp 61-72

Harsanyi J (1955) Cardinal welfare, individualistic ethics, and interpersonal coparisions of utility. J Polit Econ 63:309-321

Hayes JH, Dekhtyar A, Sundaram SK (2005) Improving after-the-fact tracing and mapping: supporting software quality predictions. IEEE Softw22(6):30-37

Haynes, S (2006) Three studies of design rationale as explanation**,** Dutoit A, McCall R, Mistrík I, Paech B (eds) Rationale Management in Software Engineering. Springer Verlag, Heidelberg, pp 53-71

Herbsleb JD (1999) Metaphorical representation in collaborative software engineering. ACM Work Activities Coordination and Collaboration Conference: WACC 99. San Francisco, New York: ACM, pp 117-126

Hild M, Jeffrey R, Risse M (1998) Preference aggregation after Harsanyi. In: Salles M, Weymark J (eds) Justice, Political Liberalism, and Utilitarianism. Cambridge: Cambridge University Press

Hirsch M (2002) Making RUP agile. In: OOPSLA 2002 Practitioners Reports, Seattle, Washington, ACM, New York, NY

Hofmann HF, Lehner F (2001) Requirements engineering as a success factor in software projects. IEEE Softw18(4):58-66

Hordijk W, Wieringa R (2006) Reusable rationale blocks: improving quality and efficiency of design choices. In: Dutoit A, McCall R, Mistrík I, Paech B (eds) Rationale management in software engineering, Springer, pp 353-371

Hudlicka E (1997) Summary of knowledge elicitation techniques for requirements analysis. Course Material for Human Computer Interaction, Worcester Polytechnic Institute

Hull MC, Jackson K, Dick J (2002) Requirements Engineering. Springer-Verlag. London, UK

Humphrey W (1995) A Discipline for Software Engineering, Addison-Wesley, Reading, MA

Humphrey W (2000) The Personal Software Process. CMU/SEI-2000-TR-022

IEEE (1990) IEEE standard glossary of software engineering terminology. IEEE Std 610.12-1990

IEEE (1993) IEEE Std 610.12-1990. IEEE Standard Glossary of Software Engineering Terminology. IEEE

IEEE (1998) IEEE Standard for Software Maintenance. IEEE Std 1219-1998

IEEE (2000) IEEE Std 1471-2000. Recommended Practice for Architectural Descriptions of Software-Intensive Systems, IEEE

IEEE (2004a) Guide to the Software Engineering Body of Knowledge (SWEBOK) 2004 edn, IEEE

IEEE (2004b) IEEE standard for software verification and validation. IEEE Std 1012-2004

IEEE/EIA (1996) Software life cycle processes. ISO/IEC 12207.0-1996

Jacobson I, Booch G, Rumbaugh J (1999) The Unified Software Development Process, Addison-Wesley Publishing Co

Jackson M (2007), Specialization in Software Engineering. Keynote paper. To appear in: Proceedings of 14th Asia Pacific Software Engineering Conference (APSEC 2007), Nagoya, Japan, 5-7 December 2007, IEEE Computer Society

Jensen RW, Tonies CC (1979) Software Engineering. Prentice Hall, NJ

Jung H, Kim S, Chung C (2004) Measuring software product quality: a survey of ISO/IEC 9126. IEEE Softw, 21(5):88-92

Juristo N, Moreno AM, Strigel W (2006) Guest editors' introduction: Software testing practices in industry. IEEE Softw 23(4):19-21

Kahn H (1962) Thinking about the unthinkable, New York: Horizon

Kahneman D, Tversky A (2000) Choices, Values, and Frames, New York: Cambridge University Press

Kajiko-Mattsson M (2001) The state of documentation practice within corrective maintenance. In: Proceedings of the International Conference on Software Maintenance, pp 354-363

Karacapilidis N, Papadias D (2001) Computer supported argumentation and collaborative decision making: the HERMES system. Inform Syst, 26(4):259-277

Karat J, Carroll JM, Alpert SR, Rosson MB (1995) Evaluating a multimedia history system as support for collaborative design. In: Nordby K, Helmersen P, Gilmore D, Arnesen S (eds) Proceedings of Human-Computer Interaction — INTERACT'95 Lillehammer, Norway, London: Chapman and Hall, pp 346-353

Karau SJ, Williams KD (1993) Social loafing: A meta-analytic review and theoretical integration. J Personality Soc Psychol 65(4): 681-706

Karlsson J, Ryan K (1997) A cost-value approach to prioritizing requirements. IEEE Softw14(5):67-74

Kazman R, Asundi J, Klein M (2003) Quantifying the value of architecture design decisions: lessons from the field, Proceedings ICSE-25 (International Conference on Software Engineering), Portland, OR: IEEE Computer Society, pp 557-562

Kelly T, Littman J (2001) Lessons in Creativity from IDEO, America's Leading Design Firm. New York: Doubleday

Kimelman D, Rosenburg B, Roth T (1998) Visualization of dynamics in real world software systems. In: Stasko J, Dominigue J, Brown M, Price B (eds) Software Visualization: Programming as a Multimedia Experience. The MIT Press, pp 293-314

King JMP, Bañares-Alcántara R (1997) Extending the scope and use of design rationale records. Artif Intell Eng Des Anal Manuf11:155-167

Kirschner, PA, Buckingham Shum, SJ, Carr, CS (eds) (2003) Visualizing Argumentation. London: Springer

Kitchenham B, Linkman S (1998) Validation, verification, and testing: Diversity rules. IEEE Softw. 15(4):46-49

Kitchenham BA, Travassos GH, von Mayhauser A, Niessink F, Schneidewind NF, Singer J, Takada S, Vehvilainen R, Yang, H (1999) Towards an ontology of software maintenance, J Softw Maintenance: Res Practice 11:365-389

Klein GA (1998) Source of power: How people make decisions. Cambridge, MA: MIT Press

Klein GA (1997a) Developing expertise in decision making. Thinking and Reasoning 3(4):337-352

Klein M (1997b) An exception handling approach to enhancing consistency, completeness and correctness in collaborative requirements capture. Concurrent Eng Res Appl 5(1):37-46

Klein M, Kazman R (1999) Attribute-Based Architectural Styles. CMU/SEI-99-TR-022. Software Engineering Institute, Carnegie Mellon University, Pittsburg, PA

Kleppe J, Warmer J, Bast W (2003) MDA explained, the model driven architecture: practice and promise, Addison-Wesley

Knodel J, Muthig D (2006) The role of rationale in the design of product line architectures – a case study from industry. In: Dutoit A, McCall R, Mistrík I, Paech B (eds) Rationale Management in Software Engineering, Springer, pp 297-312

Knuth DE (1992) Literate programming. Stanford, California: Center for the Study of Language and Information, CSLI Lecture Notes, no. 27

Ko AJ, Aung H, Myers BA (2005) Eliciting design requirements for maintenance-oriented IDEs: a detailed study of corrective and perfective maintenance tasks. In: Proceedings of the 27th international Conference on Software Engineering. St. Louis, MO, pp 126-135

Kolodner J (1993) Case-based reasoning. Morgan Kaufmann Publishers, San Mateo, California

Kontio J (1996) A case study in applying a systematic method for COTS selection In: Proceedings of the 18th International Conference on Software Engineering. Berlin, Germany, pp 201-209

Koschke R, Quante J (2005) On dynamic feature location. In: Proceedings of the 20th IEEE/ACM international Conference on Automated Software Engineering. Long Beach, CA, pp 86-95

Kraut RE, Streeter LA (1995) Coordination in software development, Communications of the ACM, 38(3), 69-81

Kraut RE (2003) Applying social psychological theory to the problems of group work. In J.M. Carroll (ed), HCI Models, theories, and frameworks: Toward a multidisciplinary science. San Francisco: Morgan-Kaufmann, pp 325-356

Kruchten P (1999) The Rational Unified Process: An Introduction, Addison-Wesley

Kunz W, Rittel HWJ (1970) Issues as elements of information systems, Working Paper 131, Center for Urban and Regional Development, University of California, Berkeley

Kuusela J, Savolainen J (2000) Requirements engineering for product families. In: Proceedings of the 22nd International Conference on Software Engineering, Limerick, Ireland, pp 61-69

Lai VS, Wong BK, Cheung W (2002) Group decision making in a multiple criteria environment: A case using the AHP in software selection, Eur J Oper Res 137:134-144

Lam W, Shankararaman V (1999) Requirements change: a dissection of management issues. In: Proceedings of the 25th Euromicro Conference, p. 2244

Lammers S (1986) Programmers at Work. Microsoft Press, Redmond, Washington.

Lanza M (2001) The evolution matrix: recovering software evolution using software visualization techniques. In: Proceedings of the 4th International Workshop on Principles of Software Evolution. Vienna, Austria, pp 37-42

Larman C (2004) Agile and Iterative Development: A Manager's Guide, Addison-Wesley, New York

Larman C, Basili VR (2003) Iterative and incremental development: A brief history, IEEE Computer, IEEE, pp 47-56

Latane B, Bourgeois, MJ (2001) Dynamic social impact and the consolidation, clustering, correlation, and continuing diversity of culture. In: Hogg MA, Tindale RS (eds) Blackwell Handbook of Social Psychology: Group Processes. Oxford, UK: Blackwell

Lave J, Wenger E (1991) Situated Learning. Legitimate Peripheral Participation, Cambridge, UK: University of Cambridge Press

Lave J (1988) Cognition in practice: Mind, Mathematics and Culture in Everyday Life. New York: Cambridge University Press

Law J, Rothermel G (2003) Whole program Path-Based dynamic impact analysis. In: Proceedings of the 25th International Conference on Software Engineering, Portland, Oregon, pp 308-318

Lawson B (1979) Cognitive Strategies in Architectural Design. Ergonomics 22(1): 59-68

Leavitt HJ (1951) Some effects of certain communication patterns on group performance. J Abnorm Soc Psychol 46:38-50.

Lee J (1990) SIBYL: A tool for managing group decision rationale. Proceedings of the 1990 ACM Conference on Computer-Supported Cooperative Work, ACM, New York, pp 79-92

Lee J (1991) Extending the Potts and Bruns model for recording design rationale. In: Proceedings of the 13th International Conference on Software Engineering, Austin, TX, pp 114-125

Lee J, Lai KY (1996) What's in design rationale? In: Moran T, Carroll J (eds) Design Rationale Concepts, Techniques, and Use. Lawrence Erlbaum Associates, pp 21-51

Lee J (1990) SIBYL: A qualitative design management system, In: Winston PH, Shellard, S (eds) Artificial Intelligence at MIT: Expanding frontiers, pp. 104-133, Cambridge MA: MIT Press

Lehman MM (1996) Laws of software evolution revisited. In: Montangero C (ed.) Lecture Notes In Computer Science, vol. 1149. Springer-Verlag, London, pp 108-124

Lehman MM (2005) The role and impact of assumptions in software development, maintenance, and evolution. In: Proceedings of the IEEE International Workshop on Software Evolvability. Budapest, Hungary, pp 3-14

Lehman MM, Frenández-Ramil J (2006) The role and impact of assumptions in software engineering and its products. In: Dutoit AH, McCall R, Mistrik I, Paech B (eds) Rationale Management in Software Engineering. Springer, Germany, pp 311-328

Lewis C, Rieman J, Bells B (1996) Problem-centered design for expressiveness. In: Moran TP, Carroll JM (eds) Design Rationale, Concepts, Techniques and Use, Lawrence Erlbaum Associates, Mahwah, New Jersey, pp 147-184

Lewis JA, Henry SA, Kafura DG, Schulman RS (1991) An empirical study of the object-oriented paradigm and software reuse. In: Proceedings of OOPSLA, pp 184-196

Lientz BP, Swanson, EB (1988) Software maintenance management. Addison-Wesley

Lindquist C (2005) Fixing the requirements mess. CIO Magazine. 15 November, http://www.cio.com/archive/111505/require.html

Liskov, Guttag (1986) Abstraction and specification in program development, MIT Press, Cambridge

Lloyd P, Scott P (1994) Discovering the design problem. Des Stud 15(2): 125-140

Lougher R, Rodden T (1993) Group support for the recording and sharing of maintenance rationale. Softw Eng J 8(6):295-306

Lozano-Tello A, Gomez-Perez A (2002) BAREMO: How to choose the appropriate software component using the analytic hierarchy process. In: Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering, pp 781 - 788

Mackay WE, Ratzer AV, Janecek P (2000) Video artifacts for design: Bridging the gap between abstraction and detail. In: ACM DIS 2000: Conference on Designing Interactive Systems. Brooklyn, New York, New York: ACM, pp 72-82

MacLean A, Young RM, Bellotti VME, Moran T (1996) Questions, Options and Criteria. In: Moran TP, Carroll JM (eds) Design Rationale, Concepts, Techniques and Use. Lawrence Erlbaum Associates, Mahwah, New Jersey, pp 53-106

MacLean A, Young RM, Bellotti VME, Moran T (1991) Questions, Options and Criteria: Elements of design space analysis. J Human–Comput Interact 6:201-250

MacLean A, Young RM, Moran TP (1989) Design rationale: the argument behind the artifact. In: Proceedings of the SIGCHI conference on human factors in computing systems: wings for the mind. ACM Press, pp 247-252

Madsen KH (1994) A guide to metaphorical design. Commun ACM 37(12):57-62

Maletic JI, Marcus A (2001) Supporting program comprehension using semantic and structural information. In: Proceedings of the 23rd International Conference on Software Engineering. Toronto, Canada, pp 103-112

Mannion M, Keepence B, Kaindl H, Wheadon J (1999) Reusing single system requirements from application family requirements. In: Proceedings of the International Conference on Software Engineering. Limerick, Ireland, pp 453-462

Manola F (1999) Providing Systematic Properties (Ilities) and Quality of Service in Component-Based Systems. Technical report, Object Services and Consulting, Inc

Martin J (1991) Rapid Application Development. Macmillan, NY

Matena V, Hapner M (1999) Enterprise JavaBeans specification. v1.1. Sun Microsystems, java.sun.com/products/ejb/docs.html

Mayer B (1997) Object-oriented software construction, 2nd edition, Prentice-Hall

Maynard-Ried II P, Shoham Y (2001) Belief fusion: aggregating pedigreed belief states. Journal of Logic, Language and Information. Kluwer Academic Publishers. 10:183-209

McAndrews DR (2000) The Team Software Process (TSPSM): An Overview and Preliminary Results of Using Disciplined Practices. Technical Report CMU/SEI-2000-TR-015

McCall R (1979a) On the structure and use of issue systems in design, Doctoral Dissertation 1978, University of California, Berkeley, University Microfilms

McCall R (1979b) Final Report for Project STIEC (Scientific and Technical Information in the European Community), Studiengruppe für Systemforschung, Heidelberg

McCall R (1986) Issue-Serve Systems: A Descriptive Theory of Design, In: Design Methods and Theories, vol 20, no 3, DMG, San Luis Obispo, pp 443-458

McCall R (1991) PHI: a conceptual foundation for design hypermedia. Des Stud 1:30-41

McCall R, Bennett P, Johnson E (1994) An overview of the Phidias II HyperCAD system", In: Proceedings of the 1994 Conference of The Association for Computer Aided Design in Architecture. A. Harfmann, M. Fraser (eds), Association for Computer Aided Design in Architecture, pp. 63-74

McCall R, Johnson E (1997) Using argumentative agents to catalyze and support collaboration in design. Autom Constr 6(4):299-309

McCall R, Bennett P, d'Oronzio P, Ostwald J, Shipman F, Wallace N (1990) Phidias: A PHI-Based Design Environment Integrating CAD Graphics into Dynamic Hypertext. In: Proceedings of the European Conference on Hypertext (ECHT 1990)

McCall R, Bennett P, d'Oronzio P, Oswald J, Shipman FM III, Wallace N (1992) PHIDIAS: Integrating CAD graphics into dynamic hypertext. In: Streitz N,

Rizk A, André J (eds) Hypertext: Concepts, Systems and Applications, Cambridge University Press, New York, NY, pp 152-165

McCall R, Mistrik I, Schuler W (1981) An Integrated Information and Communication System for Problem Solving. Proceedings of the Seventh International CODATA Conference. Pergamon, London

McCall R, Mistrik I (2005) Capture of software requirements and rationale through collaborative software development. Requirements Engineering for Sociotechnical Systems, Mate JL and Silva A (eds) Information Science, Hershey, PA, pp 303-317

McKerlie D, MacLean A (1993) QOC in action (abstract): using design rationale to support design. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. Amsterdam, The Netherlands ACM, New York, NY, pp 519

Mehta A, Heineman GT (2002) Evolving legacy system features into fine-grained components. In: Proceedings of the 24th International Conference on Software Engineering. Orlando, Florida, pp 417-427

Mellor SJ, Clark AN, Futagami T (2003) Guest editors' introduction: Model-driven development. IEEE Softw 20(5):14-18

Mens K, Mens T, Wermelinger M (2002) Supporting software evolution with intentional software views. In: Proceedings of the International Workshop on Principles of Software Evolution. Orlando, Florida, pp 138-142

Mens T, Demeyer S (2001) Future trends in software evolution metrics. In: Proceedings of the 4th International Workshop on Principles of Software Evolution, Vienna, Austria, pp 83-86

Meyer T, Ghose AK, Chopra S (2001). Multi-agent context-based merging. In: Proceedings of Common Sense 2001: The Fifth Symposium on Logical Formalizations of Commonsense Reasoning, New York, USA, May 2001

Meyer B (1997) Object-Oriented Software Construction, 2nd ed. Prentice-Hall

Mockus A, Fielding RT, Herbsleb JD (2002) Two Case Studies of Open Source Software Development: Apache and Mozilla. ACM Trans Softw Eng Methodol 11(3):209-346

Mohamed A, Ruhe G, Eberlein A (2005) Decision support for customization of the COTS selection process. In: Proceedings of 2nd International Workshop on Models and Processes for the Evaluation of COTS Components (MPEC'05). 27th International Conference on Software Engineering. St. Louis, Missouri, pp 1-4

Mongin P (1998) The paradox of the Bayesian experts and state-dependent utility theory. J Math Econ 29(3):331-361

Morisio M, Seaman C, Parra A, Basili VR, Condon S, Kraft S (2000) Investigating and improving a COTS-Based software development process. In: Proceedings of the 22nd International Conference on Software Engineering. Limerick, Ireland, pp. 32-41

Myer B (1988) Object Oriented Software Construction, Prentice Hall

Myers KL, Zumel NB, Garcia PE (1999) Automated capture of rationale for the detailed design process. In: Proceedings of the Eleventh National Conference

on Innovative Applications of Artificial Intelligence (IAAI-99), AAAI, Menlo Park, CA, pp 876-883

Narayanan NH, Kolodner JL (1995) Case libraries in support of design education: the DesignMuse Experiences, Proceedings of the 25th Annual Frontiers in Education Conference, IEEE Press

Nentwich C, Emmerich W, Finkelstein A, Ellmer E (2003) Flexible consistency checking. ACM Trans Softw Eng Methodol 12(1):28-63

Newman MW, Landay JA (2000) Sitemaps, Storyboards, and Specifications: A Sketch of Web Site Design Practice. In Proceedings of Designing Interactive Systems: DIS 2000. New York, NY. pp 263-274, August 17-19, 2000

Ng TH, Cheung SC, Chan WK, Yu YT (2006) Toward effective deployment of design patterns for software extension: a case study. In Proceedings of the 2006 International Workshop on Software Quality. Shanghai, China, pp 51-56

Ngo-The A, Ruhe G. (2005) Decision Support in Requirements Engineering, In: Arum A, Wohlin C (eds) Engineering and Managing Software Requirements. Springer, pp 267-286

Nierstrasz O, Ducasse S, Gĭrba T (2005) The story of moose: an agile reengineering environment. In: Proceedings of the 10th European Software Engineering Conference Held Jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering. Lisbon, Portugal, pp 1-10

Nuseibeh B (1997) Ariane 5 Who Dunnit? IEEE Softw 14(3):15-16

Nuseibeh B, Easterbrook S (2000) Requirements engineering: a roadmap. In: Proceedings of the Conference on the Future of Software Engineering. Limerick, Ireland, pp 35-46

Nuseibeh B, Easterbrook S, Russo A (2000) Leveraging Inconsistency in Software Development. Computer 33(4):24-29

Oberto R (2002) FAIR/DART Option #2. Advanced Projects Design Team. NASA Jet Propulsion Laboratory

Object Management Group (2000) The Common Object Request Broker: Architecture and Specification Version 2.4

Oinas-Kukkonen H (1988) Evaluating the usefulness of design rationale in CASE. Eur J Inform Syst, 7(3):185-191

OMG (2005a) UML Object Constraint Language (OCL) Specification. v2.0. The Object Modeling Group

OMG (2005b) UML Superstructure. v2.1.1, The Object Modeling Group

Ossher H, Tarr P (1999) Multi-dimensional Separation of Concerns in Hyperspace, Tech. Report RC 21453(96717), IBM, TJ Watson Research Center, NY

Overhage S (2004) UnSCom: A standardized framework for the specification of software components. In: Weske M, Liggesmeyer P (eds) Object-Oriented and Internet-Based Technologies, 5th Annual International Conference on Object-Oriented and Internet-Based Technologies, Concepts, and Applications for a Networked World. Lecture Notes in Computer Science (LNCS) 3263, Springer, pp 169-184

Parnas DL, Clements PC (1986) A Rational Design Process: How and why to fake it. IEEE Trans Softw Eng SE-12(2):251-257

Parsons S (2001) Qualitative Methods for Reasoning under Uncertainty, MIT Press

Parsons S, Hunter A (1998) A review of uncertainty handling formalisms. In: Hunter A, Parsons S (eds) Applications of Uncertainty Formalisms. Lecture Notes In Computer Science, vol. 1455. Springer-Verlag, London, pp 8-37

Patterson DA (2005) 20th Century vs. 21st Century C&C: The SPUR Manifesto, The President's Letter, Communications of the ACM, Vol 48, No 3, pp 15-16

Paulk MC, Curtis B, Chrissis MB, Weber CV (1993) Capability Maturity Model for Software, Version 1.1. CMU/SEI-93-TR-024

Payne C, Allgood CF, Chewar CM, Holbrook C, McCrickard DS (2003) Generalizing Interface Design Knowledge: Lessons Learned from Developing a Claims Library. IEEE International Conference on Information Reuse and Integration (IRI 2003), Las Vegas, pp 362-369

Pelled L, Eisenhardt K, Xin K (1999) Exploring the black box: An analysis of work group diversity, conflict, and performance. Admin Sci Q 44:1-28

Peña-Mora F, Sriram D, Logcher R (1995) Design rationale for computer-supported conflict mitigation. ASCE J Comput in Civil Eng 9(1):57-72

Peña-Mora F, Vadhavkar S (1997) Augmenting design patterns with design rationale. Artif Intell Eng Des, Anal Manuf 11(2): 93–108

Pennock D, Maynard-Ried II P, Giles CL, Horvitz E (2000) A normative examination of ensemble learning algorithms. In: Proceedings of the 17th International Conference on Machine Learning, pp 735-742

Pigoski TM (1996) Practical Software Maintenance. Wiley

PMBOK (2003) A Guide to the Project Management Body of Knowledge. Project Management Institute Standards Committee, IEEE Std 1490-2003

Pollice G, Augustine L, Lowe C, Madhur J (2003) Software Development for Small Teams: a Rup-Centric Approach. Addison Wesley Longman

Potts C (1996) Supporting software design: integrating design methods and design rationale. In: Moran TP, Carroll JM (eds) Design rationale: concepts, techniques, and use. Lawrence Erlbaum Associates, Mahwah, New Jersey, pp 295-321

Potts C, Bruns (1988) Recording the reasons for design decisions, In: Proceedings of the 10th International conference on software engineering, Singapore, pp 418-427

Potts C (1989) A generic model for representing design methods, Proc. 11th Int. Conf. Software Eng., Pittsburgh, IEEE Comp. Soc. Press

Potts C, Takahashi K, Anton A (1994) Inquiry-based requirements analysis, IEEE Softw 11(2):21-32

Prechelt L, Unger B, Tichy WF, Brössler P, Votta LG (2001) A controlled experiment in maintenance comparing design patterns to simpler solutions. IEEE Trans Softw Eng 27(12):1134-1144

Price B, Baecker R, Small I (1998) An introduction to software visualization. In: Stasko J, Dominigue J, Brown M, Price B (eds) Software Visualization: Programming as a Multimedia Experience. The MIT Press, pp 3-27

Queille J, Voidrot J, Wilde N, Munro M (1994) The impact analysis task in software maintenance: a model and a case study. In: Proceedings of the International Conference on Software Maintenance, Victoria, BC, pp 234-242

Rajlich V (2006) Changing the software paradigm. Commun ACM 49(8):67-70

Ramesh B, Dhar V (1992) Supporting systems development by capturing deliberations during requirements engineering, IEEE Trans Softw Eng 18(6), 498-510

Ramesh B, Dhar V (1994) Representing and maintaining process knowledge for large-scale systems development, EEE Expert 9(2):54-60

Ramires J, Antunes P, Respício A (2005) Software requirements negotiation using the software quality function deployment. In: Fuks H, Lukosch S, Salgado A (eds) Groupware: Design, Implementation, and Use. Lecture Notes in Computer Science. Vol. 3706, Heidelberg, Springer-Verlag, pp 308-324

Ramler R, Biffl S, Grunbacher P (2006) Value-based management of software testing. In: Biffl S, Aurum A, Boehm B, Erdogmus H, Grunbacher P (eds) Value-Based Software Engineering, Springer

Randal B, Buxton JN (1970) Software engineering techniques: A report on a conference sponsored by the NATO Science Committee, NATO

Rasmussen J (1974) The human data processor as a system component: Bits and pieces of a model. Riso-M-1722. Roskilde, Denmark: Danish Atomic Energy Commission

Raymond E (2001) The Cathedral and the Bazaar. Revised Edition. O'Reilly

Redwine S, Riddle W (1985). Software technology maturation, In: Proc. of the 8th ICSE, pp 189-200

Reeves B, Shipman FM III (1992) Supporting communication between designers with artifact-centered evolving information spaces In: Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work, November 1-4, Toronto, Ontario, Canada, pp 394-401

Reiss SP (2002) Constraining software evolution. In: Proceedings of the International conference on Software maintenance, Montreal, Quebec Canada, pp 162-171

Reiss SP, Kennedy CM, Wooldridge T, Krishnamurthi S (2003) CLIME: an environment for constrained evolution demonstration description. In: Proceedings of the 25th International Conference on Software Engineering. Portland, Oregon, pp 818-819

Reitman WR (1965) Cognition and Thought.: An Information Processing Approach. New York: Wiley

Réquilé-Romanczuk A, Cechich A, Dourgnon-Hanoune A, Mielnik J (2005) Towards a knowledge-based framework for COTS component identification. In: Proceedings of the Second international Workshop on Models and Processes for the Evaluation of Off-the-Shelf Components. St. Louis, Missouri, ACM Press, New York, NY, pp 1-4

Riesbeck C, Schank R (1989) Inside Case-Based Reasoning. Lawrence Erlbaum. Hillsdale, NJ

Rittel H (1980) APIS: A concept for an argumentative planning information system. Working paper 324, Institute of Urban and Regional Development, University of California, Berkeley

Rittel H, Weber M (1973) Dilemmas in a general theory of planning. Policy Sciences 4: 155-169

Rittel H (1972a) On the planning crisis: Systems analysis of the first and second generations. Bedriftsokonomen, Norway, 8:390-396

Rittel H. (1972b) Son of Rittelthink, DMG Newsletter 3-10 Berkeley: University of California.

Robillard MP (2005) Automatic generation of suggestions for program investigation. In: Proceedings of the 10th European Software Engineering Conference Held Jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering. Lisbon, Portugal, pp 11-20

Roman G (1985) A taxonomy of current issues in requirements engineering. Computer, 18(4):14-22

Rosson MB, Alpert SA (1990) The cognitive consequences of object-oriented design. Human-Computer Interaction, 5:345-379

Rosson MB, Carroll JM (1996) The reuse of uses in Smalltalk programming. ACM Trans Computer-Human Inter, 3(3):219-253

Rothermel G, Elbaum S (2003) Putting Your Best Tests Forward. IEEE Softw 20(5):74-77

Royce WW (1970) Managing the development of large software systems: concepts and techniques. In: Proceedings of IEEE WESTCON, Los Angeles, California, pp 1-9

Saaty TL (1980) The Analytic Hierarchy Process, McGraw-Hill, New York

Saiedian H, Dale R (2000) Requirements engineering: making the connection between the software developer and customer. Inform Softw Technol 42:419-428

Schmid K (2002) A comprehensive product line scoping approach and its validation. In: Proceedings of ICSE. Orlando, Florida, pp 593-603

Schmidt DC (2006) Guest editor's introduction. Model-Driven Engineering. Computer 39(2):25-31

Schmidt DC, Buschmann F (2003) Patterns, frameworks, and middleware: their synergistic relationships. In: Proceedings of the 25th International Conference on Software Engineering. Portland, Oregon, IEEE Computer Society, Washington, DC, pp 694-704

Schneider K (2006) Rationale as a by-product, In: Rationale Management in Software Engineering, Dutoit A, McCall R, Mistrik I, Paech B (eds) Springer Verlag, Heidelberg, pp 91-109

Schön D (1983) The Reflective Practitioner. How Professionals Think in Action. Basic Books, New York

Schultz DJ (1979) A case study in system integration using the Build approach. In: Proceedings of the 1979 Annual Conference. pp 143-151

Scriven M (1967) The methodology of evaluation. In: Tyler R, Gagne R, Scriven M (eds) Perspectives of Curriculum Evaluation, Rand McNally, pp 39-83

SEI (1997) Integrated Product Development Capability Maturity Model, Draft Version 0.98. Pittsburgh, PA: Enterprise Process Improvement Collaboration and Software Engineering Institute, Carnegie Mellon University

Shafer G (1976) A Mathematical Theory of Evidence. Princeton University Press, Princeton, New Jersey

Sharp H, Finkelstein A, Galal G (1999) Stakeholder Identification in the Requirements Engineering Process. In: Proceedings of the 10th International Workshop on Database and Expert Systems Applications, pp 387

Shipman FM III, McCall R (1994) Supporting knowledge-base evolution with incremental formalization. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Boston, Massachusetts, United States, pp 285-291

Shipman F, McCall R (1997) Integrating different perspectives on design rationale: Supporting the emergence of design rationale from design communication, Artif Intell Eng Des, Anal, Manuf 11(2):141-154

Shipman FM III, Marshall CC (1999a) Formality considered harmful: Experiences, emerging themes, and directions on the use of formal representations in interactive systems. Computer-Supported Cooperative Work 8(3):333-352

Shipman FM III, Marshall CC (1999b) Spatial hypertext: An alternative to navigational and semantic links. ACM Comput Surv 31(4):1-5

Siff M, Reps T (1999) Identifying modules via concept analysis. IEEE Trans Softw Eng 25(6):749-768

Sim S, Duffy A (1994) A new perspective to design intent and design rationale. In: Workshop Notes for Representing and Using Design Rationale. Artificial Intelligence in Design. pp 4-12

Simon HA (1957) Models of Man: Social and Rational. New York: Wiley

Singer J (1998) Practices of software maintenance. In: Proc. of the International Conference on Software Maintenance (ICSM'98). Bethesda, Maryland, USA, pp 139-145

Smith RP, Tjandra P (1998) Experimental observation of iteration in engineering design. Res Eng Des 10(2):107-117

Sneed H (1995) Planning the reengineering of legacy systems. IEEE Softw. 12(1):24-34

Sneed HM (2001) Impact analysis of maintenance tasks for a distributed object-oriented system. In: Proceedings of the IEEE International Conference on Software Maintenance, Florence, Italy, pp 180-189

Software Engineering Coordinating Committee (2004) Guide to the Software Engineering Body of Knowledge (SWEBOK), IEEE Computer Society

Sommerville I (2007) Software Engineering. 8th edn, Addison Wesley

Spanoudakis G, Zisman A (2001) Inconsistency management in software engineering: survey and open research issues. In: Handbook of Software Engineering and Knowledge Engineering 1, World Scientific, pp 329-380

Srivastava A, Thiagarajan J (2002) Effectively prioritizing tests in development environment. In: Proceedings of the 2002 ACM SIGSOFT International Symposium on Software Testing and Analysis. Roma, Italy, pp 97-106

Stahl T, Volter M (2006) Model-driven software development, Wiley, NY
Stark GE, Oman P, Skillicorn A, Ameele A (1999) An examination of the effects of requirements changes on software maintenance releases. J Softw Maintenance 11(5):293-309
Stobie K (2005) Too darned big to test. Queue 3(1):30-37
Stone M (1961) The linear opinion pool. Ann Math Stat 32:1339-1342
Subramaniam GV (2000) Object model resurrection — an object oriented maintenance activity. In: Proceedings of the 22nd International Conference on Software Engineering. Limerick, Ireland, pp 324-333
Subramanian N, Chung L (2001) Software architecture adaptability: an NFR approach. In: Proceedings of the 4th International Workshop on Principles of Software Evolution. Vienna, Austria, pp 52-61
Subramanian N, Chung L (2002) Tool support for engineering adaptability into software architecture. In: Proceedings of the International Workshop on Principles of Software Evolution. Orlando, Florida, pp 86-96
Sutcliffe AG, Carroll JM (1999) Designing claims for reuse in interactive systems design. Int J Human-Comput Stud 50(3):213-241
Tang A, Han J (2005) Architecture rationalization: A methodology for architecture verifiability, traceability and completeness. Proc. 12th Annual IEEE Int. Conf. and Workshop on the Engineering of Computer Based Systems (ECBS'05), pp 135-144
Tang A, Jin Y, Han J (2005a) A rationale-based architecture model for design traceability and reasoning. J Sys Softw 80:918-934
Tang A, Jin Y, Han J, Nicholson A (2005b) Predicting change impact in architecture design with Bayesian Belief Networks. In: proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture, pp 67-76
Tang A, Babar M, Gorton I, Han J (2006) A survey of architecture design rationale. J Sys Softw 79:1792-1804
Tarr P, Clarke LA (1998) Consistency management for complex applications. In: Proceedings of the 20th International Conference on Software Engineering, Kyoto, Japan, pp 230-239
Taylor RN, van der Hoek A (2007) Software Design and Architecture: The once and future focus of software engineering. In: 2007 Future of Software Engineering. International Conference on Software Engineering. Minneapolis, MN, pp 226-243
Thayer RH, Dorfman M (1990) Introduction, issues, and terminology. In: Thayer RH, Dorfman M (eds) System and Software Requirements Engineering, IEEE Computer Society Press, Los Alamitos, CA, 1st edn, pp 1-3
Thelin T, Runeson P, Wohlin C (2003) An experimental comparison of usage-based and checklist-based reading. IEEE Trans Softw Eng 29(8):687-702.
Tip F (1995) A survey of program slicing techniques. J Prog Lang 3(3):121-189
Tonella P, Antoniol G (1999) Object oriented design pattern inference. In: Proceedings of the IEEE International Conference on Software Maintenance, pp 230-238
Toulmin S (1958) The Uses of Argument. Cambridge University Press, Cambridge

Turner M, Budgen D, Brereton P (2003) Turning software into a service. IEEE Comput 36(10):38-44

Tyree J, Akerman A (2005) Architecture decisions: Demystifying architecture. IEEE Softw 22(2):19-27

van Lamsweerde A (2001) Goal-oriented requirements engineering: a guided tour. In: Proceedings of the 5th IEEE International Symposium on Requirements Engineering, Toronto, pp 249-263

van Lamsweerde A (2003) From system goals to software architecture. In: Bernardo M, Inverardi P (eds) Formal Methods for Software Architectures, Springer-Verlag

van Lamsweerde A (2004) Goal-oriented requirements engineering: a roundtrip from research to practice. In: Proceedings of the 12th International Conference on Requirements Engineering, Tokyo, Japan, pp 4-8

van Lamsweerde A, Letier E (2000) Handling obstacles in goal-oriented requirements engineering. IEEE Trans Softw Eng Special Issue on Exception Handling 26(10): 978-1005

Vetschera R (2006) Preference-based decision support in software engineering. In: Biffl S, Aurum A, Boehm B, Erdogmus H, Grunbacher P (eds) Value-Based Software Engineering, Springer, pp 67-89

Vincenti W (1990) What engineers know and how they know it. John Hopkins University Press

Voas JM, Miller KW (1995) Software testability: the new verification. IEEE Softw 12(3):17-28

von Mayrhauser A, Vans AM (1994) Comprehension processes during large scale maintenance. In: Proceedings of the 16th International Conference on Software Engineering. Sorrento, Italy, pp 39-48

Wahid S, Smith J.L, Berry B, Chewar CM, McCrickard, DS (2004) Visualization of Design Knowledge Component Relationships to Facilitate Reuse. Proceedings of the 2004 IEEE International Conference on Information Reuse and Integration (IRI '04), Las Vegas NV, November, pp 414-419.

Walker RJ, Holmes R, Hedgeland I, Kapur P, Smith A (2006) A lightweight approach to technical risk estimation via probabilistic impact analysis. In: Proceedings of the 2006 International Workshop on Mining Software Repositories, Shanghai, China, pp 98-104

Wallin P, Fröberg A, Axelsson A (2007) Making decisions in integration of automotive software and electronics: A method based on ATAM and AHP, In: Proceedings SEAS'07 (Fourth International Workshop on Software Engineering for Automotive Systems), IEEE Computer Society, pp 5

Wang L, Tan KC (2005) Software testing for safety-critical applications. IEEE Instrum Measure Mag 8(2):38-47

Wang N, Schmidt D, O'Ryan C (2001) Overview of the CORBA component model. In: Councill B, Heineman G (eds) Component-Based Software Engineering, Addison-Wesley, Upper Saddle River, pp 557-571

Wang X, Xiong G (2001) Design rationale as part of corporate technical memory, In: International Conference on Systems, Man, and Cybernetics, Tucson, AZ: 3:1904-1908

Weick KE (1995) Sensemaking in organizations. Sage: Thousand Oaks, CA
Weiser M (1981) Program slicing. In: Proceedings of the 5th International Conference on Software Engineering, San Diego, California, pp 439-449
Weizenbaum, J (1966) ELIZA—A computer program for the study of natural language communication between man and machine. Comm ACM 9(1):36-35
Wellman MP (1990) Fundamental concepts of qualitative probabilistic networks. Artif Intell 44:257-303
Wenger E (1998) Communities of practice: Learning, meaning, and identity. New York: Cambridge University Press
Whitehead J (2007) Collaboration in Software Engineering: A Roadmap. In: 2007 Future of Software Engineering, International Conference on Software Engineering. Minneapolis, MN, pp. 214-225
Whittaker S, Schwarz H (1995) Back to the Future: Pen and Paper Technology Supports Complex Group Coordination, in Proceedings of ACM CHI'95Conference on Human Factors in Computing Systems, pp 495-502
Workshop on Multi-Dimensional Separation of Concerns, International Conference on Software Engineering 2000, http://www.research.ibm.com/hyperspace/workshops/icse2000
Zadeh LA (1965) Fuzzy sets. Inform Control 8:338-353
Zadeh LA (1978) Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets Syst 1:3-28
Zhu L, Gorton I (2007) UML profiles for design decisions and non-functional requirements. In: Proceedings of the Workshop on the Sharing and Reusing of Architectural Knowledge, at the International Conference of Software Engineering (ICSE), Minneapolis, Minnesota.
Zimmermann T, Weisgerber P, Diehl S, Zeller A (2004) Mining version histories to guide software changes. In: Proceedings of the 26th International Conference on Software Engineering, pp 563-572
Zimring CM, Do ED, Domeshek ED, Kolodner JL (1995) Supporting case-study use in design education: A computational case-based design aid for architecture. In: Mohsen JP (ed.) Computing in Civil Engineering: Proceedings of the Second Congress. New York: American Society of Civil Engineers, pp 1635-1642
Ziv H, Richardson DJ, Klosch R (1996) The uncertainty principle in software engineering. Technical Report UCI-TR-96-33, University of California, Irvine

Glossary

abstraction
A view of an object that focuses on the information relevant to a particular purpose and ignores the remainder of the information [IEEE Std 610.12-1990]

adaptability
Adaptability concerns the ease of altering a system to meet the needs of a user [Randal and Buxton 1970]

adaptive maintenance
Changes made to the software during maintenance that do not change its functionality

agile methods
Software development methods that use iterative development to provide a more agile response to changing requirements

analysis
The phase in the software lifecycle that analyzes the system requirements in order to build a model describing the application domain

analysis of design
A process that provides a *view* of the design process that is not otherwise available

Analytic Hierarchy Process (AHP)
A decision-making technique where alternatives are evaluated by making a series of pairwise comparisons

anthropomorphism
Software analysis and design method that involves metaphorically thinking about software components as animate

anti-model
Vivid characterizations of features and outcomes that a problem solver or decision-makers definitely wants to avoid

architectural description
A collection of products to document an architecture [IEEE Std 1471-2000]

architectural design

The result of the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system [adapted from IEEE Std 610.12-1990]

architectural framework

Describes the elements of a concrete architecture in terms of components, connectors, and dependencies

architectural style

A family of architectures constrained by component/connector vocabulary, topology, and semantic constraints [adapted from Garlan and Shaw 1993]

architectural tactic

A transformation of an architecture to achieve particular quality attribute goals

architecturally significant requirements (ASR)

Software requirements that have broad cross-functional implications such performance, usability, maintainability, and security. These include nonfunctional requirements (NFRs) and quality attributes

architecture

Architecture is the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution [IEEE Std 1471-2000].

Architecture is a description (model) of the basic arrangement and connectivity of parts of a system (either a physical or a conceptual object or entity) [ISO 15704 1999]

architecture decision

A high-level design decision that an architect or designer takes to satisfy the functional and nonfunctional requirements of a system

architecture description

A collection of products to document an architecture [IEEE Std 1471-2000]

artifact

The result of any activity in the software lifecycle such as requirements, architecture model, design specifications, source code, and test scripts [http://w3.umh.ac.be/genlog/SE/SE-contents.html]

assumption

A proposition that is believed to be, but not known to be, true

Attribute-based Architectural Styles (ABASs)

Architectural styles associated with an attribute reasoning framework associated with a quality attribute

awareness (in collaboration)
Relevant knowledge about collaborators; for example, their identity, activity, goals and expectations, and focus of attention. Effective collaboration requires awareness

bad smell
Code structures that signal potential problems or poor designs that indicate the need for refactoring

basic software
Software that is used by the computer hardware to give the system its basic functions, like an operating system, or perform elementary tasks such as a compiler [Chaudron et al. 2004]

benchmark
A benchmark is a set of tests used to compare the performance of alternative tools, methods, or techniques [http://w3.umh.ac.be/genlog/SE/SE-contents.html]

black-box reuse
A kind of reuse where a component is reused without changing anything within the component

black-box testing
Software testing that looks only at the inputs and expected outputs and is not aware of the internal contents of the code

business case
Description of the system in terms of the stakeholders that have to make the decision to develop the system, together with an analysis of the development and operational cost of the system, and of the benefits of the system and the revenues it might generate [Chaudron et al. 2004]

Capability Maturity Model (CMM)
A process model developed by the Software Engineering Institute to assess the maturity of a software organization's process by classifying it into one of five levels

Capability Maturity Model Integration (CMMI)
A replacement for the CMM that assesses the maturity level for 22 process areas

Computer-Aided Software Engineering (CASE)
The use of software tools to assist in the development and maintenance of software

classical decision model
Decision model in which solution alternatives are exhaustively enumerated, analyzed, and contrastively evaluated

common ground
The knowledge shared by interlocutors or collaborators. Effective collaboration requires periodic verification of common ground

commonalities
The set of features or properties of a component (or system) that are the same, or common, between systems [http://w3.umh.ac.be/genlog/SE/SE-contents.html]

community of practice
Groups of actors that share values, norms, concepts, behavior scripts, and strategies pertaining to a domain of human endeavor

compatibility
The ability of two or more systems or components to perform their required function while sharing the same hardware or software environment [IEEE Std 610.12-1990]

component
A component is a self-contained piece of software with clearly defined interfaces and explicitly declared context dependencies [Stahl and Volter 2006]

Component-Based Software Engineering (CBSE)
A software engineering approach that builds software systems from re-usable software components

conceptual framework (or frame of reference)
Establishes terms and concepts pertaining to the content and use of a specific architectural descriptions

confirmation bias
Tendency of human decision-makers to seek and prize data that confirms their decisions over data that disconfirms their decisions

consistency
The degree of uniformity, standardization, and freedom from contradiction among the documents or parts of a system or component [IEEE Std 610.12-1990]

consistency management
The process of managing the consistency between the different software artifacts developed during the software development process

coordination
Self-management among collaborators to ensure that individual contributions can be synthesized into effective wholes

corrective maintenance
Software maintenance changes that are made in order to repair defects

correctness
The ability of software products to perform their exact tasks, as defined by their specification [Meyer 1997]

Commercial Off-the-Shelf (COTS)
Software products that are purchased rather than custom made

criterion-based evaluation
A way of evaluating a decision alternative or artifact feature which consists of (1) the statement of a criterion, e.g. a goal, and (2) an assessment of the alternative or feature with respect to the stated criterion, these two elements in effect constituting a single argument for or against the alternative or feature

decision-centric rationale approaches
Rationale approaches that deal with the rationale for decision-making in artifact creation

Decision Representation Language (DRL)
Lee's revision and extension of the Potts and Bruns approach to rationale. DRL's schema corresponds roughly to a superset of QOC's schema and has dependency relationships between elements. Like QOC, DRL uses a form of criterion-based evaluation

defect
A problem in a software artifact that causes it to be incorrect

deliberate *(verb)*
To consider what the answer to *a question* should be and, more specifically, to evaluate one or more proposed answers to a question

design *(noun)*
An artifact description that is detailed enough to be used to construct that artifact

design pattern
Names, abstracts, and identifies the key aspects of a common design structure that make it useful for creating a reusable object-oriented design [Gamma et al. 1995]

design rationale (DR)
DR is the reasoning underlying the creation and use of artifacts

design space decisions
Decisions as to what features that an artifact will have

Design Space Analysis
Representation of a set of design space decision tasks together with their decision alternatives and the evaluations of these alternatives

domain-oriented issue base (DOIB)
PHI-based collections of issues, positions, arguments, and subissues that commonly arise in a particular design domain

detailed design
The result of the process of refining and expanding the preliminary design of a system or component to the extent that the design is sufficiently complete to be implemented [IEEE Std 610.12-1990]

domain
An area of knowledge or activity characterized by a set of concepts and terminology understood by practitioners in that area [Booch et al., 1990]

enhancive maintenance
Software maintenance changes that are made to add additional features or otherwise improve a software system

extensibility
The ease of adapting software products to changes of specification [IEEE Std 610.12-1990]

extreme programming (XP)
A popular agile method that proposes taking software best practices "to the extreme"

familiarity bias
Tendency of human decision-makers to consider familiar data and interpretations as typical

fixation
Tendency of designers to make solution decisions before adequately understanding the full problem space, and then to disproportionate adduce confirmatory evidence to justify and maintain those decisions

flexibility
The case with which a system or component can be modified for use in application or environments other than those for which it was specifically designed [IEEE Std 610.12-1990]

framework
A generic structure that can be adapted or extended via systematic extension or configuration [adapted from Stahl and Volter 2006]

functional design
The result of the process of defining the working relationships among the components of a system [IEEE Std 610.12-1990]

functional requirement
A requirement that describes functionality that the system must provide in order to be acceptable to the customer

functionality
The extent of services provided by a system [adapted from Mayer 1997]

generality
The degree to which a system or component performs a broad range of functions [IEEE Std 610.12-1990]

generative paradigm
An alternative devised by Gruber and Russell to the "record and replay" paradigm used by almost all other approaches to rationale. Rather than recording rationale, the generative paradigm involves recreating it after the fact by deriving it from various data obtained automatically during design

glass-box testing
Software testing based on information about the structure of the code. Examples would be branch or path testing

hypermedia (hypertext)
Information structure consisting of nodes of content including link anchors to other nodes of content

iconic models
Graphical models in Euclidean space of artifacts that will occupy Euclidian space when constructed

ility
A quality attribute, or nonfunctional requirement. The name comes from the form of many requirements such as scalability, reusability, modifiability, etc.

implementation
The software lifecycle phase where the software (i.e., the source code) is written

ill-structured problem
Term used by Reitman and later by Simon to refer to open-ended problems, like software design, that cannot be uniquely decomposed into verifiable steps. See also "wicked problems"

inconsistency
A state in which two or more overlapping elements of different software models make assertions about aspects of the system they describe which are not jointly satisfiable [Spanoudakis and Zisman 2001]

incremental delivery
A software development process where the software is developed and delivered in increments rather than as one completed system at the end of development

inspection
A verification technique that involves reviewing the software artifacts to look for defects

integrated rationale
Rationale for a software system that is stored with or as part of the software that it describes and explains

integration testing
Tests performed to ensure that the subsystems that comprise a software system work correctly together

interaction
The mutual influence of two actors and/or components. Interaction is performed via an interface [Chaudron et al. 2004]

interface
For two components, or a component and an external actor, a model of types of the messages that are exchanged and the order in which this may occur [Chaudron et al. 2004]

Issue-Based Information System (IBIS)
A way of modeling argumentation; it was invented by Rittel in 1970. See also "wicked problems"

Knowing-in-Action
In Schön's theory of Reflective Practice, the process of performing tasks in an intuitive, nonreflective manner that involves unselfconscious engagement in the task at hand

Lehman's laws
Eight laws that describe how software systems evolve

maintenance
Software modifications made to systems after they have been delivered to the customer

metaphor (in software design)
A direct comparison of a software component to a physical object (desk top), a social institution (library), an animate entity (garbage collector), etc. to assist in comprehension and communication. See also "anthropomorphism"

metrics
Measurements of software properties or processes used to evaluate the software and/or the process by which it was developed

model
A formal representation of an aspect of a system. Typical examples are data models that give a static view and process models that give a dynamic view [Chaudron et al. 2004]

model (with concurrent multiple views of RBSE)
Describes logical organization, dynamic behavior, software organization, process decomposition, and physical realization

Model-Driven Architecture (MDA)
In MDA, models are the central elements of the software development process. The main goal is to transform platform-specific models, possibly automatically, into platform-independent models [Kleppe et al. 2003]

model-driven development
Software developed by first building a model of the system and then transforming it into the code

multiscale visualization
Visualizations with qualitatively/structurally distinct levels of zoom

narrative
An informal design rationale representation in which stories or scenarios describe how and why a design decision was reached, or how and why a user experienced a design system

naturalistic decision-making
Decision-making methodology that emphasizes identifying and leveraging the strengths of human decision-making, instead of merely remediating weaknesses and fallacies

nonfunctional requirement (NFR)
Software requirements that describe desirable properties of the software that do not map to specific functionality but instead apply to the system as a whole

ontology
A set of entities, their definitions, and the relationships between them

Open-source software
Software where the source code is freely available for use and modification

open–closed principle
A class that follows the open-closed principle is open to extension and closed to modification

operational environment
The environment in which the software is operating after delivery

Pareto optimality
A solution that cannot be improved further by one criteria without worsening in another

pattern mining
A process of extracting and documenting architecturally significant information from patterns to support the architecture design and evaluation process

perfective maintenance
Changes to improve a software system that are not in response to defects

Personal Software Process (PSP)
A methodology for improving individual software process by collecting and using metrics captured during software development on an individual basis

platform
A set of subsystems and technologies that provide a coherent set of functionality through interfaces and specified usage patterns, which any application supported by that platform can use without concern for the details of how the functionality provided by the platform is implemented [Kleppe et al. 2003]

portability
The property of a system which permits it to be mapped from one environment to a different environment [Randal and Buxton 1970]

post-specification traceability
The ability to trace from a software requirement forward to the code that implements it and the tests that verify it has been implemented

Potts–Bruns rationale approach
A modification of IBIS for use in software design. The crucial innovation of their approach is to include in their schema elements that represented "intermediate artifacts," i.e., the various models, documents, and prototypes produced during design to represent the software being designed

pre-specification traceability
The ability to trace from a software requirement backward to the customer request that it responds to

preventative maintenance
Changes to a software system to avoid anticipated future problems

problem-based evaluation
Informal design rationale approach in which a set of problem scenarios are used to evaluate a design proposal analytically

Procedural Hierarchy of Issues (PHI)
A refinement of IBIS whose main innovation is to show that frequently a decision on one issue depends on the decisions made on others. PHI models rationale as a quasihierarchical structure of issues linked by dependency relationships

product line
A collection of existing and potential products that address a coherent business area and share a set of similar characteristics. All these products are made by the same process and for the same purpose, and differ only in style, model or size [http://w3.umh.ac.be/genlog/SE/SE-contents.html]

program comprehension
The process of understanding the source code of a software system

quality assurance

An approach to ensure that the software product, and the processes used to develop it, conform to the software specification and other required standards and procedures

Questions, Options, and Criteria (QOC)

A rationale approach resembling IBIS but not derived from it. Like IBIS, QOC centers on decision tasks that are represented as *questions*, but unlike IBIS, QOC deals only with "design space" questions, i.e. those that determine features of the designed artifact, rather than the wider range of questions dealt with by IBIS. QOC's main innovation is the use of criterion-based evaluation in the first level of argumentation of decision alternatives (options)

Rapid Application Development (RAD)

A software development process that makes heavy use of Computer Aided Software Engineering (CASE) tools to build software systems quickly

rationale approach

A way of modeling and using rationale

rationale database

Structured repository of reusable rationales, accessible via type of system, application, scenario, issue, position, argument, etc.

Rationale Based Software Engineering (RBSE)

Research on and use of rationale capture and delivery to support every aspect of software engineering.

rationale capture problem

The difficulty of capturing rationale in a structured form. This is considered by many to be the main impediment to widespread use of rationale approaches in artifact creation in general and software development in particular

rationale management

The capture, representation, retrieval, and use of the reasoning behind decisions made during the system development process

rationale management system

Software tools developed to support rationale management

RATSpeak

Burge's extension of DRL to make it more suitable for software engineering. RATSpeak introduces new types of elements into its schema and provides an argument ontology tailored to software engineering. These additions enable automated checking and inference making

recognition-primed decision model
Expert decision model in which situations are rapidly classified and addressed as exemplars of known prototypes

re-engineering
Rewriting all or part of an existing software system to improve its quality. This typically refers to a legacy system currently in use that is no longer maintainable.

refactoring
Making modifications to code to correct "bad smells" and to prepare the code for future extension. Refactoring does not add or change functionality

Reflection-in-Action
In Schön's theory of Reflective Practice, the process of explicitly reflecting on why an intuitive performance of a task *broke down*, i.e., led to unforeseen results

Reflective Practice
Schön's theory that design and other practical problem solving activities consist of repeated alternation between two processes that he labeled Knowing-in-Action and Reflection-in-Action

regression testing
Repeating earlier tests on a previously tested product to ensure that new modifications have not introduced defects into existing code

requirement
A property that is demanded to be fulfilled by a software system [adapted from Chaudron et al. 2004]

requirements elicitation
Obtaining requirements from various system stakeholders by a variety of techniques including interviews, observation, and prototyping

requirements engineering
The process of eliciting and documenting software requirements to ensure completeness and consistency

requirements traceability
The ability to trace the impact of a requirement on the delivered system in order to ensure that all requirements have been satisfied

reuse
Using existing code when building a new system. This may or may not involve modifying that code

reverse engineering
Using the source code to create the specification and models that describe the system

satisficing
Evaluation metric used in problem solving and decision-making in which the first acceptable solution is adopted. Contrasts with optimization

Scenario Claims Analysis (SCA)
Informal design rationale in which core user interactions afforded by a software system are described by scenarios and implicit design tradeoffs in the scenarios (claims)

semantic inference
Inferences that use the semantics of the items being analyzed. In the case of rationale, it means inferencing over the contents and not just the structure.

Service-oriented development
Systems are built around a Service Oriented Architecture (SOA) using loosely coupled distributed services that can be accessed transparently of their platform implementation

situated cognition
Approach to analyzing human thought that regards the actors and objects of social and material contexts as constitutive resources

social capital
A sense of generalized reciprocity within a social group

social loafing
The tendency of people to work less hard when working in the context of others doing the same work

Software Engineering (SE)
The development and maintenance of software by the systematic application of engineering techniques in the software domain [adapted from IEEE Std 610.12-1990]

Software Engineering Institute (SEI)
A federally funded (USA) software engineering research center that conducts software engineering research in a number of areas that include software architecture, software product lines, and software process improvement

Software Engineering Rationale (SER)
SER emphasizes that rationale models are used during all activities of software development, including requirements engineering, architectural design, implementation, testing, and system deployment

Software Process (SP)
A related set of activities and processes that are involved in developing and evolving a software system [Sommerville 2007]

software process improvement
Evaluating and modifying a software development process to achieve a higher level of repeatability, maturity, and performance

solution-first bias
Tendency of designers to rapidly frame a solution to a problem they do not yet fully understand

Spiral model
A software lifecycle model that utilizes iteration where each trip around the spiral involves determining objectives, assessing risk, developing the current phase of the product, and planning the next phase

stakeholder
Anyone who has interest in the success of the software project

syntactic inference
Inference over rationale that looks only at the structure of the argumentation and not at the contents

system
A generic term for a group of interrelated, interdependent or interacting elements serving a collective purpose [Chaudron et al. 2004]

system architecture
The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution [IEEE Std 610.12-1990]

system engineering
The process of developing a system that must fulfill a certain purpose using the systematic application of engineering techniques, and of which software engineering is a part, provided the system has a software subsystem [Chaudron et al. 2004]

team software process
A methodology for working in teams that includes a framework for managing, tracking, and reporting on the team's performance

test case
A software testing document that consists of the input for and expected result of running the test

Test-driven development
A software development methodology where unit tests are written first and then the code is written to pass the test

traceability
The degree to which a relationship can be established between two or more products of the development process [adapted from IEEE Std 610.12-1990]

testing
Executing a piece of software to look for defects
traditional approach to rationale capture
The approach in which rationale is structured according a given rationale schema as it is recorded
Unified Modeling Language (UML)
A standardized specification language for object modeling [http://www.omg.org/uml]
Unified Process (UP)
A software development framework utilizing incremental and iterative development. The Unified Process contains four phases: inception, elaboration, construction, and transition
unit testing
Testing the smallest testable pieces of source code
usage-centric rationale approaches
Rationale approaches that deal with rationale derived from the experiences of users as they use artifacts
validation
Ensuring that the software system conforms to its specification
Value-based Software Engineering
A theory of software development where the emphasis is on providing value to all the system stakeholders
verification
Ensures that the software system is fit for its intended use
view
A view is a representation of a whole system from the perspective of a related set of concerns [IEEE Std 1471-2000]
viewpoint
A viewpoint is a specification of the conventions for constructing and using a view. Typical viewpoints are structure, behavior, functionality, security, distribution, performance, usability, usefulness, and reliability [IEEE Std 1471-2000]
V-model
A software development lifecycle model where each development stage is paired with the corresponding verification stage
war-room (in design)
A dedicated design workroom in which analyses and artifacts are pinned to the walls
Waterfall model
A sequential software development model where development flows from one stage to the next

white-box testing

Software testing that is based on information about the structure of the code. Examples would be branch or path testing.

wicked problems

Rittel's theory of problems of artifact creation as fundamentally open-ended and potentially controversial. According to Rittel, such problems cannot be solved using a strictly scientific approach or purely automated methods. Instead, their solution requires methods that support creative human problem solving by means of an "argumentative approach." Wicked problems theory was used to justify Rittel's pioneering work on design rationale

Index