Definitionen mit Quellenangabe

adaptive maintenance - Software maintenance performed to make a computer program usable in a changed environment.
[IEEE Std 610.12 (1990)]

ADL - focuses on the high-end structure of the overall application rather than the implementation details of any specific source module. ADLs provide both a concrete syntax and a conceptual framework for modelling a software system's conceptual architecture.
[Medvidovic, Rosenblum (1997)]

Anforderungsspezifikation - dokumentiert die wesentlichen Anforderungen an eine Software und ihre Schnittstellen, und zwar präzise, vollständig und überprüfbar.
[Ludewig, Lichter (2023)]

Anwendungsbereich – kann eine Organisation, ein Bereich innerhalb einer Organisation oder ein Arbeitsplatz sein.
Ein Anwendungsbereich ist weit genug gefasst, um die für die Konstruktion von Modellen relevanten fachlichen Zusammenhänge während der Ist-Analyse zu verstehen.
Durch einen Anwendungsbereich ist typischerweise auch eine entsprechende Anwendungsfachsprache festgelegt.
[Züllighoven (2005)]

Anwendungssoftware – Software, um fachliche Aufgaben in einem Anwendungsbereich oder mehreren Anwendungsbereichen zu erledigen. Dabei werden vorhandene oder neue Problemlösungsstrategien durch Software realisiert.
[Züllighoven (2005)]

architectural description is a model – document, product or other artifact – to communicate and record a system's architecture. An architectural description conveys a set of views each of which depicts the system by describing domain concerns.
[Ellis et al. (1996)]

architectural pattern expresses a fundamental structural organization schema for software systems. It provides a set of predefined subsystems, specifies their responsibilities, and includes rules and guidelines for organizing the relationships between them.
[Buschmann et al. (1996)]

architecture - 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)]

build process – The process of creating an executable artifact from input such as source code and configuration information. It primarily consists of compiling source code and packaging all files that are required for execution. Once the build is complete, a set of automated tests is executed that test whether the integration with other parts of the system uncovers any

error.
[Bass, Weber, Zhu (2015)]

Code and Fix - bezeichnet ein Vorgehen, bei dem Codierung oder Korrektur im Wechsel mit Ad-hoc-Tests die einzigen bewusst ausgeführten Tätigkeiten der Software-Entwicklung sind.
[Ludewig, Lichter (2023)]

complexity - (1) The degree to which a system or component has a design or implementation that is difficult to understand and verify.
Contrast with: simplicity.
(2) Pertaining to any of a set of structure-based metrics that measure the attribute in (1).
[IEEE Std 610.12 (1990)]

component - One of the parts that make up a system. A component may be hardware or software and may be subdivided into other components.
[IEEE Std 610.12 (1990)]

configuration control board (CCB) - A group of people responsible for evaluating and approving or disapproving proposed changes to configuration items, and for ensuring implementation of approved changes. Syn: change control board.
[IEEE Std 610.12 (1990)]

corrective maintenance - Maintenance performed to correct faults in software.
[IEEE Std 610.12 (1990)]

cycle - (1) A period of time during which a set of events is completed.
[IEEE Std 610.12 (1990)]

design - (1) The process of defining the architecture, components, interfaces, and other characteristics of a system or component.
(2) The result of the process in (1).
[IEEE Std 610.12 (1990)]

design recovery recreates design abstractions from a combination of code, existing documentation (if available), personal experience, and general knowledge about problem and application domains.
[Chikofsky, Cross (1990)]

egoless programming - A software development technique based on the concept of team, rather than individual, responsibility for program development. Its purpose is to prevent individual programmers from identifying so closely with their work that objective evaluation is impaired.
[EEE Std 610.12 (1990)]

Entwurfsmuster - beschreibt das Schema einer Lösung für ein bestimmtes, in verschiedenen konkreten Zusammenhängen wiederkehrendes Entwurfsproblem.
[Ludewig, Lichter (2023)]

Evolutionäre Software-Entwicklung - Vorgehensweise, die eine Evolution der Software unter dem Einfluss ihrer praktischen Erprobung einschließt. Neue und veränderte Anforderungen werden dadurch berücksichtigt, dass die Software in sequenziellen Evolutionsstufen

entwickelt wird.
[Ludewig, Lichter (2023)]

forward Engineering - is the traditional process of moving from high-level abstractions and logical, implementation-independent designs to the physical implementation of a system.
[Chikofsky, Cross (1990)]

information hiding - A software development technique in which each module's interfaces reveal as little as possible about the module's inner workings, and other modules are prevented from using information about the module that is not in the module's interface specification.
[IEEE Std 610.12 (1990)]

Inkrementelle Software-Entwicklung - Das zu entwickelnde System bleibt in seinem Gesamtumfang offen; es wird in Ausbaustufen realisiert. Die erste Stufe ist das Kernsystem. Jede Ausbaustufe erweitert das vorhandene System und wird in einem eigenen Projekt erstellt. Mit der Bereitstellung einer Erweiterung ist in aller Regel auch (wie bei der iterativen Entwicklung) eine Verbesserung der alten Komponenten verbunden.
[Ludewig, Lichter (2023)]

integration - The process of combining software components, hardware components, or both into an overall system.
[IEEE Std 610.12 (1990)]

integration testing - Testing in which software components, hardware components, or both are combined and tested to evaluate the interaction between them.
[IEEE Std 610.12 (1990)]

interface - is a boundary across which two independent entities meet and interact or communicate with each other.
[Bachmann et al. (2002)]

Iterative Software-Entwicklung - Software wird in mehreren geplanten und kontrolliert durchgeführten Iterationsschritten entwickelt. Ziel dabei ist, dass in jedem Iterationsschritt – beginnend bei der zweiten Iteration – das vorhandene System auf der Basis der im Einsatz erkannten Mängel korrigiert und verbessert wird. Bei jedem Iterationsschritt werden die charakteristischen Tätigkeiten Analysieren, Entwerfen, Codieren und Testen durchgeführt.
[Ludewig, Lichter (2023)]

Klassenbibliothek - besteht aus einer Menge von Klassen, die wiederverwendbar sind und allgemein – also unabhängig vom Anwendungskontext – nutzbare Funktionalität anbieten.
[Ludewig, Lichter (2023)]

Konfigurationsverwaltung - ist diejenige Rolle oder Organisationseinheit, die die Software-Einheiten und Konfigurationen identifiziert, verwaltet, bei Bedarf bereitstellt und ihre Änderungen überwacht und dokumentiert. Dazu gehört auch die Rekonstruktion älterer Software-Einheiten und Konfigurationen.
[Ludewig, Lichter (2023)]

law of continuing change - A system that is used undergoes continuing change until it becomes more economical to replace it by a new or restructured system.
[Lehman, Belady (1985)]

law of increasing entropy - The entropy of a system increases with time unless specific work is executed to maintain or reduce it.
[Lehman, Belady (1985)]

legacy system - A large software system that we don't know how to cope with but that is vital to our organization.
[Bennett (1995)]

maintenance - The process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment.
Syn: software maintenance.
See also: adaptive maintenance; corrective maintenance; perfective maintenance.
[IEEE Std 610.12 (1990)]

metric - A quantitative measure of the degree to which a system, component, or process possesses a given attribute.
See also: quality metric.
[IEEE Std 610.12 (1990)]

milestone - is defined as a scheduled event that marks the completion of one or more important tasks, and it is used to measure achievements and development progress. At a milestone, a predefined set of deliverables should have reached a predefined state to enable a review.
[Wallin et al. (2002):]

Modul - ist eine Menge von Operationen und Daten, die nur so weit von außen sichtbar sind, wie dies die Programmierer explizit zugelassen haben.
[Ludewig, Lichter (2023)]

modular decomposition - The process of breaking a system into components to facilitate design and development; an element of modular programming.
[IEEE Std 610.12 (1990)]

modularity - The degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.
[IEEE Std 610.12 (1990)]

module - (1) A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading; for example, the input to, or output from an assembler, compiler, linkage editor, or executive routine.
(2) A logically separable part of a program.
[IEEE Std 610.12 (1990)]

operation and maintenance phase - The period of time in the software life cycle during which a software product is employed in its operational environment, monitored for satisfactory

performance, and modified as necessary to correct problems or to respond to changing requirements.
[IEEE Std 610.12 (1990)]

perfective maintenance - Software maintenance performed to improve the performance, maintainability, or other attributes of a computer program.
[IEEE Std 610.12 (1990)]

Pipe-Filter-Architekturmuster - dient dazu, eine Anwendung zu strukturieren, die Daten auf einem virtuellen Fließband verarbeitet.
[Ludewig, Lichter (2023)]

preventive maintenance - Maintenance performed for the purpose of preventing problems before they occur.
[IEEE Std 610.12 (1990)]

process - (1) A sequence of steps performed for a given purpose; for example, the software development process.
(2) See also: task; job.
(3) To perform operations on data.
[IEEE Std 610.12 (1990)]

product line architecture - A core asset that is the software architecture for all the products in a software product line. A product line architecture explicitly provides variation mechanisms that support the diversity among the products in the software product line.
[Northrop, Clements (2007)]

project - A temporary activity that is characterized by having a start date, specific objectives and constraints, established responsibilities, a budget and schedule, and a completion date. If the objective of the project is to develop a software system, then it is sometimes called a software development or software engineering project.
[Thayer (1997)]

prototype - A preliminary type, form, or instance of a system that serves as a model for later stages or for the final, complete version of the system.
[IEEE Std 610.12 (1990)]

prototyping - A hardware and software development technique in which a preliminary version of part or all of the hardware or software is developed to permit user feedback, determine feasibility, or investigate timing or other issues in support of the development process.
[IEEE Std 610.12 (1990)]

Qualität - Gesamtheit von Eigenschaften und Merkmalen eines Produktes oder einer Tätigkeit, die sich auf die Eignung zur Erfüllung gegebener Erfordernisse beziehen.
[DIN 55350-11:1995-08]

quality assurance - (1) A planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements.
(2) A set of activities designed to evaluate the process by which products are developed or

manufactured.
[IEEE Std 610.12 (1990)]

quality metric - (1) A quantitative measure of the degree to which an item possesses a given quality attribute.
(2) A function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which the software possesses a given quality attribute.
[IEEE Std 610.12 (1990)]

quality model - defined set of characteristics and of relationships between them, which provides a framework for specifying quality requirements and evaluating quality.
[ISO/IEC 25000 (2014)]

Rahmenwerk (Framework) - ist eine Architektur aus Klassenhierarchien, die eine allgemeine generische Lösung für ähnliche Probleme in einem bestimmten Kontext vorgibt.
[Züllighoven (2005)]

rapid prototyping - A type of prototyping in which emphasis is placed on developing prototypes early in the development process to permit early feedback and analysis in support of the development process.
[IEEE Std 610.12 (1990)]

redocumentation is the creation or revision of a semantically equivalent representation within the same relative abstraction level.
[Chikofsky, Cross (1990)]

reengineering- is the examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form.
[Chikofsky, Cross (1990)]

Referenzarchitektur - definiert Software-Bausteine für einen Anwendungsbereich durch ihre Strukturen und Typen, ihre Zuständigkeiten und ihre Interaktionen.
[Ludewig, Lichter (2023)]

regression testing - Selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements.
[IEEE Std 610.12 (1990)]

reliability - The ability of a system or component to perform its required functions under stated conditions for a specified period of time.
[IEEE Std 610.12 (1990)]

requirement - (1) A condition or capability needed by a user to solve a problem or achieve an objective.
(2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.

(3) A documented representation of a condition or capability as in (1) or (2).
[IEEE Std 610.12 (1990)]

requirements analysis - (1) The process of studying user needs to arrive at a definition of system, hardware, or software requirements.
(2) The process of studying and refining system, hardware, or software requirements.
[IEEE Std 610.12 (1990)]

requirements specification language - A specification language with special constructs and, sometimes, verification protocols, used to develop, analyze, and document hardware or software requirements.
[IEEE Std 610.12 (1990)]

re-structuring - is a transformation from one form of representation to another at the same relative level of abstraction. The new representation is meant to preserve the semantics and external behavior of the original.
[Chikofsky, Cross (1990)]

reusability - The degree to which a software module or other work product can be used in more than one computer program or software system.
[IEEE Std 610.12 (1990)]

reverse Engineering - is the process of analyzing a system to identify the system's components and their interrelationships and create representations of the system in another form or at a higher level of abstraction.
[Chikofsky, Cross (1990)]

Risiko - Ein Problem, das noch nicht eingetreten ist, aber wichtige Projektziele oder Projektergebnisse gefährdet, falls es eintritt. Ob es eintreten wird, kann nicht sicher vorausgesagt werden.
[Hindel et al. (2009)]

robustness - The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions.
[IEEE Std 610.12 (1990)]

Schnittstelle - Die Grenze zwischen zwei kommunizierenden Komponenten. Die Schnittstelle einer Komponente stellt die Leistungen der Komponente für ihre Umgebung zu Verfügung und/oder fordert Leistungen, die sie aus der Umgebung benötigt.
[Ludewig, Lichter (2023)]

software - Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system.
See also: application software; support software; system software
Contrast with: hardware.
[IEEE Std 610.12 (1990)]

software architecture - of a program or computing system - is the structure or structures of the system which comprise software elements, the externally visible properties of those

elements, and the relationships among them.
[Bass, Clements, Kazman (2003)]

software component - is an architectural entity that (1) encapsulates a subset of the system‘s functionality and/or data, (2) restricts access to that subset via an explicitly defined interface, and (3) has explicitly defined dependencies on its required execution context.
[Taylor, Medvidovic, Dashofy (2010)]

Software development cycle - The period of time that begins with the decision to deve-lop a software product and ends when the software is delivered. This cycle typically includes a requirements phase, design phase, implementation phase, test phase, and sometimes, installation and checkout phase.
Notes: (1) The phases listed above may overlap or be performed iteratively, depending upon the software development approach used.
(2) This term is sometimes used to mean a longer period of time, either the period that ends when the software is no longer being enhanced by the developer, or the entire software life cycle.
[IEEE Std 610.12 (1990)]

software development process - The process by which user needs are translated into a software product. The process involves translating user needs into software requirements, transforming the software requirements into design, implementing the design in code, testing the code, and sometimes, installing and checking out the software for operational use.
Note: These activities may overlap or be performed iteratively.
[IEEE Std 610.12 (1990)]

software engineering - (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.
(2) The study of approaches as in (1).
[IEEE Std 610.12 (1990)]

Software Engineering - ist die Entdeckung und Anwendung solider Ingenieurprinzipien mit dem Ziel, auf wirtschaftliche Art Software zu bekommen, die zuverlässig ist und auf realen Rechnern läuft.
[F. L. Bauer]

Software Engineering - ist die Herstellung und Anwendung einer Software, wobei mehrere Personen beteiligt sind oder mehrere Versionen entstehen.
[D. L. Parnas]

Software Engineering - ist jede Aktivität, bei der es um die Erstellung oder Veränderung von Software geht, soweit mit der Software Ziele verfolgt werden, die über die Software selbst hinausgehen.
[Ludewig, Lichter (2023)]

software life cycle - The period of time that begins when a software product is conceived and ends when the software is no longer available for use. The software life cycle typically

includes a concept phase, requirements phase, design phase, implementation phase, test phase, installation and checkout phase, operation and maintenance phase, and, sometimes, retirement phase.
Note: These phases may overlap or be performed iteratively.
[IEEE Std 610.12 (1990)]

software quality assurance - See: quality assurance.
[IEEE Std 610.12 (1990)]

software requirements specification (SRS) - Documentation of the essential requirements (functions, performance, design constraints, and attributes) of the software and its external interfaces.
[IEEE Std 610.12 (1990)]

Software reuse - is the process whereby an organization defines a set of systematic operating procedures to specify, produce, classify, retrieve, and adapt software artifacts for the purpose of using them in its development activities.
[Mili et al. (2002)]

Software-Einheit - Ein Stück Software, das im Software-Lebenslauf entsteht und für die Entwicklung, den Betrieb oder die Wartung des Software-Systems relevant ist. Eine Software-Einheit kann unabhängig von anderen Software-Einheiten bearbeitet, gespeichert und ersetzt werden. Sie besteht nicht aus kleineren Software-Einheiten, ist also im Sinne der Verwaltung atomar.
[Ludewig, Lichter (2023)]

Software-Entwicklung nach dem Treppenmodell - Das zu entwickelnde System wird in definierten Ausbaustufen realisiert und ausgeliefert. Die erste Stufe ist das Kernsystem. Jede weitere Ausbaustufe erweitert das vorhandene System um Leistungen und Merkmale, die überwiegend bereits zu Beginn des Gesamtprojekts geplant worden sind.
[Ludewig, Lichter (2023)]

Software-System - ist eine Menge von Software-Einheiten und ihren Beziehungen, wenn sie gemeinsam einem bestimmten Zweck dienen. Dieser Zweck ist im Allgemeinen komplex, er schließt neben der Bereitstellung eines ausführbaren Programms (oder auch mehrerer) gewöhnlich die Organisation, Verwendung, Erhaltung und Weiterentwicklung ein.
[Ludewig, Lichter (2023)]

Software-Wartung - ist jede Arbeit an einem bestehenden Software-System, die nicht von Beginn der Entwicklung an geplant war oder hätte geplant werden können und die unmittelbare Auswirkungen auf den Benutzer der Software hat.
[Ludewig, Opferkuch (2004)]

specification - A document that specifies, in a complete, precise, verifiable manner, the requirements, design, behavior, or other characteristics of a system or component, and, often, the procedures for determining whether these provisions have been satisfied.
[IEEE Std 610.12 (1990)]

specification language - A language, often a machine-processible combination of natural and formal language, used to express the requirements, design, behavior, or other characteristics

of a system or component. For example, a design language or requirements specification language.
Contrast with: programming language; query language.
[IEEE Std 610.12 (1990)]

Strenges Wasserfall- oder Einbahnstraßenmodell - Jeder Aktivität im aktivitätsorientierten Wasserfallmodell ist eine spezielle Phase zugeordnet.
[Ludewig, Lichter (2023)]

stub - (1) A skeletal or special-purpose implementation of a software module, used to develop or test a module that calls or is otherwise dependent on it.
(2) A computer program statement substituting for the body of a software module that is or will be defined elsewhere.
[IEEE Std 610.12 (1990)]

system - A collection of components organized to accomplish a specific function or set of functions.
[IEEE Std 1471 (2000)]

system life cycle - The period of time that begins when a system is conceived and ends when the system is no longer available for use.
[IEEE Std 610.12 (1990)]

taxonomy - A scheme that partitions a body of knowledge and defines the relationships among the pieces. It is used for classifying and understanding the body of knowledge.
[IEEE Std 610.12 (1990)]

technical debt – In software-intensive systems, technical debt is the collection of design or implementation constructs that are expedient in the short term, but set up a technical context that can make future changes more costly or impossible.
[Avgeriou et al. (2016)]

test driver - A software module used to invoke a module under test and, often, provide test inputs, control and monitor execution, and report test results. Syn: test harness.
[IEEE Std 610.12 (1990)]

Testen - ist die – auch mehrfache – Ausführung eines Programms auf einem Rechner mit dem Ziel, Fehler zu finden.
[Ludewig, Lichter (2023)]

use case - A sequence of interactions between an actor (or actors) and a system triggered by a specific actor, which produces a result for an actor.
[Jacobson et al. (1992)]

user story - A description of a need from a user's perspective together with the expected benefit when this need is satisfied.
Notes:
1. User stories are typically written in natural language using a phrase template and are accompanied by acceptance criteria.
2. In agile development, user stories are the main means for communicating needs

between product owner and the development team.
[Glinz, 2020]

version - (1) An initial release or re-release of a computer software configuration item, associated with a complete compilation or recompilation of the computer software configuration item.
(2) An initial release or complete re-release of a document, as opposed to a revision resulting from issuing change pages to a previous release.
[IEEE Std 610.12 (1990)]

Wasserfall- oder Dokumentenmodell - Die Software-Entwicklung wird als Folge von Aktivitäten betrachtet, die durch Teilergebnisse (Dokumente) gekoppelt sind. Diese Aktivitäten können auch gleichzeitig oder iterativ ausgeführt werden. Davon abgesehen ist die Reihenfolge der Aktivitäten fest definiert, nämlich (sinngemäß) Analysieren, Spezifizieren, Entwerfen, Codieren, Testen, Installieren und Warten.
[Ludewig, Lichter (2023)]

waterfall model - A model of the software development process in which the constituent activities, typically a concept phase, requirements phase, design phase, implementation phase, test phase, and installation and checkout phase, are performed in that order, possibly with overlap but with little or no iteration.
[IEEE Std 610.12 (1990)]

Werkzeug - dient zur Ausführung einer Arbeit, die – wenigstens prinzipiell – auch ohne das Werkzeug geleistet werden könnte. Im Produkt ist das Werkzeug nicht mehr enthalten, es sei denn als Ausrüstung für die Wartung.
[Ludewig, Lichter (2023)]

Literaturangaben

Avgeriou, P., P.B. Kruchten, I. Ozkaya, C. Seaman (2016): Managing Technical Debt in Software Engineering. Report from Dagstuhl Seminar 16162.

Bachmann F., L. Bass, P. Clements, D. Garlan, J. Ivers, R. Little, R. Nord, J. Stafford (2002): Documenting Software Architecture: Documenting Interfaces. CMU/SEI-2002-TN-015, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

Bass, L., I. Weber, L. Zhu (2015): DevOps: A Software Architect's Perspective. Addison-Wesley, Boston, MA.

Bass, L., P. Clements, R. Kazman (2003): Software Architecture in Practice. 2nd ed., The SEI Series in Software Engineering, Addison-Wesley Pearson Education, Boston.

Bennett, K. (1995): Legacy Systems: Coping With Success. IEEE Software, Vol. 12 (1), 19-23.

Buschmann, F., R. Meunier, H. Rohnert, E. Sommerlad, M. Stal (1996): Pattern-Oriented Software Architecture – A System of Patterns. John Wiley & Sons, Chichester, England.

Chikofsky, E.J., J.H. Cross (1990): Reverse Engineering and Design Recovery: A Taxonomy. IEEE Software, Vol. 7 (1), 13-17.

DIN 55350-11 (1995): Begriffe zu Qualitätsmanagement und Statistik – Teil 11: Begriffe des Qualitätsmanagements.

Ellis, W.J., R.F. Hilliard, P.T. Poon, D. Rayford, T.F. Saunders, B. Sherlund, R.L. Wade (1996): Toward a Recommended Practice for Architectural Description. Proceedings of 2nd IEEE International Conference on Engineering of Complex Computer Systems, Montreal, 408-413.

Glinz, M. (2020): Wörterbuch der Requirements Engineering Terminologie.

Version 2.0.Hindel, B., K. Hörmann, M. Müller, J. Schmied (2009): Basiswissen Software-Projekt-management. 3. Aufl., dpunkt.verlag, Heidelberg.

IEEE Std 1471 (2000): Recommended Practice for Architectural Description of Software-Intensive Systems. IEEE Standards Association.

IEEE Std 610.12 (1990): IEEE Standard Glossary of Software Engineering Terminology. IEEE Standards Association.

ISO/IEC 25000 (2014): Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE.

Jacobson, I., M. Christerson, P. Jonsson, G. Övergaard (1992): Object-Oriented Software Engineering – A Use Case Driven Approach. ACM Press, Wokingham, England, Addison-Wesley, Reading, MA.

Lehman, M.M., M.A. Belady (1985): Program Evolution – Processes of Software Change.

Academic Press, London.

Ludewig, J., H. Lichter (2023): Software Engineering – Grundlagen, Menschen, Prozesse, Techniken. 4. Aufl., dpunkt.verlag Heidelberg.

Ludewig, J., S. Opferkuch (2004): Software-Wartung – eine Taxonomie. Softwaretechnik-Trends, Band 24 (2), 35-36.

Medvidovic, N., D.S. Rosenblum (1997): Domains of Concern in Software Architectures and Architecture Description Languages. Proceedings of the USENIX Conference on Domain-Specific Languages, Santa Barbara, CA, 199-212.

Mili, H., A. Mili, S. Yacoub, E. Addy (2002): Reuse-Based Software Engineering – Techniques, Organization, and Controls. John Wiley & Sons, New York, NY.

Northrop, L. M., P. C. Clements (2007): A Framework for Software Product Line Practice, Version 5.0.

Taylor, R.N., N. Medvidovic, E.M. Dashofy (2010): Software Architecture Foundations, Theory, and Practice. John Wiley & Sons, Hoboken, NJ.

Thayer, R.H. (1997): Software Engineering Project Management. Tutorial – Software Engineering Project Management, revised edition, IEEE Computer Society Press.

Wallin, C., S. Larsson, F. Ekdahl, I. Crnkovic (2002): Combining Models for Business Decisions and Software Development. 28th EUROMICRO Conference, Dortmund, IEEE Computer Society, 266-271.

Züllighoven, H. (2005): Object-Oriented Construction Handbook – Developing Appli-cation-Oriented Software with the Tools and Materials Approach. dpunkt.verlag, Heidelberg/Morgan Kaufmann Publishers, San Francisco, CA.