Ausbildung und Zertifizierung in der industriellen Software-Wartung

Stefan Opferkuch, Jochen Ludewig

Abteilung Software Engineering, Institut für Softwaretechnologie
Universität Stuttgart
Universitätsstraße 38
70569 Stuttgart
opferkuch@informatik.uni-stuttgart.de
ludewig@informatik.uni-stuttgart.de

Abstract: Dieser Artikel beschreibt die aktuelle Lage der Ausbildung und Zertifizierung in der Software-Branche im Allgemeinen und in der industriellen Software-Wartung im Speziellen. Dabei zeigt sich, dass eine Ausbildung für die Tätigkeiten der Software-Wartung fehlt. Um diesen Missstand zu beheben, wird im Artikel das Modell eines idealen Software-Warters entworfen und werden dessen Fähigkeiten beschrieben. Aus diesen Fähigkeiten wird unter den gegebenen industriellen Rahmenbedingungen ein Ausbildungs- und Zertifizierungskonzept für die Software-Wartung in der Industrie abgeleitet.

1 Einführung

Vorab sei gesagt, dass wir hier für eine Person, die sich mit der Software-Wartung befasst, die Bezeichnung „*Software-Warter*“ oder einfach „*Warter*“ verwenden. Natürlich ist damit nichts über das Geschlecht dieser Person ausgesagt. Es ist auch nicht impliziert, dass diese Person *ausschließlich* mit Wartung beschäftigt ist.

Auf den ersten Blick werden in der Software-Wartung die gleichen Tätigkeiten wie in der Entwicklung ausgeführt: Anforderungen werden erhoben, neue Teile entworfen, Codezeilen erstellt oder verändert, Qualitätssicherungsmaßnahmen durchgeführt. Am Ende wird die Software ausgeliefert.

Auf den zweiten Blick wird allerdings deutlich, dass diese Tätigkeiten in der Wartung unter grundlegend anderen Rahmenbedingungen stattfinden. Nach unserer Taxonomie [LO04] ist die Software-Wartung jede Arbeit an einem bestehenden Software-System, die nicht von Beginn an geplant war oder hätte geplant werden können. Im Gegensatz zur Entwicklung, existiert bei der Wartung bereits eine Software, die in der Regel auch beim Kunden eingesetzt wird. Das hat Konsequenzen:

- Alle Veränderungen oder Erweiterungen der Software müssen mit den bereits bestehenden Teilen in Einklang gebracht werden. Unerwünschte Nebenwirkungen auf unveränderte Bestandteile der Software müssen ausgeschlossen werden.

- Die Benutzer brauchen die Software. Ein längerer Ausfall bedeutet oft großen wirtschaftlichen Schaden. Dadurch steht der Warter unter Druck.

- Bei einer ganz neuen Software rechnen die Benutzer mit Fehlern und akzeptieren Workarounds. Bei einer neuen Version der Software, die sich von der alten nicht erheblich unterscheidet, sind Fehler in bisher unauffälligen Funktionen inakzeptabel.

- Die Entwickler einer neuen Software überblicken mindestens die Komponente, an der sie arbeiten. Der Warter hat dagegen weder die Zeit noch die Möglichkeit, einen Überblick zu gewinnen.

- Die Entwicklung ist zeitlich begrenzt und zielt auf einen bestimmten Zustand; die Wartung läuft über unbestimmte Zeit, ihre Ziele klären sich von Fall zu Fall und sind nicht langfristig planbar.

Die Kosten der Entwicklung lassen sich mit verschiedenen Modellen abschätzen; die Kosten der Wartung sind kaum transparent. Wir wissen nur, dass schon vor knapp dreißig Jahren mehr Aufwand in die Software-Wartung als in die Software-Entwicklung geflossen ist [ZSG79].

Trotzdem konzentrieren sich fast alle Ausbildungsprogramme auf die Entwicklung; weder im akademischen noch im industriellen Umfeld gibt es eine Ausbildung für die Software-Wartung. In der Praxis sind die meisten Menschen in den Informatik-Berufen ohne irgendeine Informatik-Ausbildung zu ihrer Beschäftigung gekommen. Die Software-Wartung wird also von Personen gemacht, denen eine entsprechende Qualifikation fehlt. Dass die Software-Wartung in vielen Unternehmen als zu langsam, zu teuer und von zu schlechter Qualität wahrgenommen wird, ist nicht zuletzt in diesem Defizit begründet.

Um die Ausbildung für die Software-Wartung zu verbessern, sind zwei Ansätze denkbar:

1. eine Ergänzung der Lehrpläne in den traditionellen Bildungseinrichtungen durch Wartungsthemen;

2. die Schulung der Software-Warter in der Praxis, um ihnen die wichtigsten Konzepte der Wartung zu vermitteln.

Es gibt an verschiedenen Hochschulen Bestrebungen, die Ausbildung im Bereich der Software-Wartung zu verbessern. Es wird jedoch lange Zeit dauern, bis Studenten, die eine solche Ausbildung genossen haben, in großer Zahl in die Praxis kommen und dort aktiv die Wartung gestalten können.

In diesem Artikel konzentrieren wir uns darum auf den zweiten Punkt, die Konzeption einer Schulung für die Masse der Software-Warter in der Industrie.

Ganz offensichtlich kann eine solche Schulung nicht mit einem Studium konkurrieren; aber sie kann als „Notreparatur“ die ärgsten Mängel in der Wartung reduzieren und die Qualität der Resultate verbessern. Damit wirkt sie langfristig Kosten senkend. Zugleich gibt die Schulung den Teilnehmern die Chance, ihre Kenntnisse durch ein Zertifikat zu belegen, sich also zu qualifizieren. Je besser sie ausgebildet sind, desto mehr Freude werden sie an ihrer Arbeit haben.

Im Kapitel 2 werden einige Zertifikate im Software-Engineering vorgestellt und ihre Ziele sowie die Personen, an die sie sich richten, diskutiert. In Kapitel 3 wird der Stand der Praxis in der Software-Wartung beschrieben. In Kapitel 4 wird anschließend erörtert, über welche Qualifikationen ein Software-Warter idealerweise verfügen sollte. Ein Ausbildungsprogramm für Software-Warter wird im Kapitel 5 vorgestellt.

2 Certified X

Aus offensichtlichen Gründen gibt es in den Software-Berufen sehr viel mehr Leute ohne einschlägige Ausbildung als in anderen Berufen. Die Statistiken zeigen, dass sich daran in der absehbaren Zukunft nicht viel ändern wird. Es lag darum nahe, Kurse zur Weiterbildung (die in Wahrheit oft eine Grundbildung sein muss) anzubieten.

Schon seit vielen Jahren gibt es Kurse und Qualifizierungsnachweise in Form von Zertifikaten verschiedener Hard- und Software-Hersteller. So bieten SAP, Microsoft, IBM, Sun, Cisco, Red Hat und viele andere Unternehmen die Möglichkeit an, sich durch eine Schulung in speziellen Technologien der jeweiligen Unternehmen weiterzubilden und diese Weiterbildung durch ein Zertifikat zu belegen.

Ein Beispiel ist der IBM Certified Specialist for Rational Unified Process (RUP). Zielgruppe des zweitägigen RUP-Kurses sind Manager, Projektleiter, Prozessverantwortliche und alle am Entwicklungsprozess beteiligten Personen. Notwendige Vorkenntnisse sind allgemeine Erfahrungen in der Softwareentwicklung. Am Ende des Kurses steht eine Multiple-Choice-Prüfung, deren Ergebnis darüber entscheidet, ob der Teilnehmer ein Zertifikat erhält, das seine Qualifikation im IBM-Produktumfeld dokumentiert.

Dieses Konzept wird in ähnlicher Form auch von den anderen oben genannten Unternehmen angewendet. Da die Zertifizierungen Technologie-abhängig sind und sich die Technologie ständig weiterentwickelt, müssen sie regelmäßig erneuert werden.

Seit einigen Jahren werden auch in Technologie-neutralen Bereichen des Software-Engineerings immer mehr Zertifizierungen angeboten. Die wohl bekannteste ist der Certified Tester [SL04] [GT07]. Im Januar 2002 begann das International Software Quality Institute, kurz ISQI, mit der Zertifizierung der Software-Tester. Damit sollte keine akademische Lehrveranstaltung erschaffen, sondern eine Mindestqualifikation für die

vielen Software-Entwickler definiert und angeboten werden, die im Bereich des Tests in den Unternehmen tätig sind, aber über keine formale Qualifikation dazu verfügen.

Es gibt drei Ausbildungsstufen: Foundation Level, Advanced Level und Expert Level, wobei die letzte Stufe derzeit (im Herbst 2007) noch in Vorbereitung ist. Im Foundation Level sind Themengebiete definiert, die das notwendige Grundlagenwissen schaffen sollen. Es wird ein gemeinsames Vokabular eingeführt, und alle Teilnehmer werden auf einen einheitlichen Wissenstand bezüglich des Software-Tests gebracht. Im Advanced Level erfolgt dann eine Konzentration auf drei Themengebiete: Test Manager, Functional Tester und Technical Tester. Die Lehrinhalte der einzelnen Ausbildungsstufen werden durch das International Software Testing Qualifications Board festgelegt [IS07a].

Die Schulungen dauern jeweils drei Tage; am Ende erfolgt ein Multiple-Choice-Test zur Überprüfung des Wissens. Im Gegensatz zu den Technologie-abhängigen Zertifizierungen muss die Zertifizierung beim Certified Tester nicht regelmäßig erneuert werden.

Ähnliche Zertifizierungen werden auf den Gebieten Software-Architektur, Projektmanagement [IS07b] und Requirements Engineering [IR07] angeboten.

Allen Zertifizierungsprogrammen ist gemeinsam, dass fundierte Grundlagen zu einem Gebiet des Software-Engineerings kompakt gelehrt werden. Dieses Prinzip ist auf weitere Gebiete übertragbar. In [Op07] wurde die Frage aufgeworfen, ob wir für die Software-Wartung einen „Certified Maintainer“, also eine nachweisbare Qualifizierung für die Wartung benötigen und welche Vorteile sich daraus ergäben. Dieser Frage gehen wir hier weiter nach.

3 Der Stand der Praxis

Der typische Software-Warter in einem Unternehmen ist ein Quereinsteiger und erst allmählich in die Software-Wartung hineingerutscht. Meist hat er zu Beginn seiner beruflichen Laufbahn eine branchenbezogene Ausbildung absolviert. Dies bedeutet, dass beispielsweise im verarbeitenden Gewerbe, z. B. bei Automobilzulieferern, häufig Software-Warter anzutreffen sind, die eine Ausbildung in Elektrotechnik oder Physik absolviert haben. Im Kredit- und Versicherungsgewerbe, also bei Banken oder Versicherungen, sind typisch Software-Warter mit kaufmännischer Ausbildung anzutreffen. Häufig haben die Mitarbeiter jahrelang in anderen Bereichen des Unternehmens gearbeitet, bevor sie zur Software-Wartung kamen. Sie verfügen also über keine Ausbildung, die sie auf diese Tätigkeit vorbereitet hätte; nur einen unternehmensinternen Programmierkurs haben sie typisch durchlaufen. Im verarbeitenden Gewerbe werden meist C, C++, Java oder auch Ada geschult und eingesetzt, im Kredit- und Versicherungsgewerbe COBOL, PL1 oder Java. Dieser Programmierkurs bleibt in vielen Fällen die einzige Informatik-Ausbildung. [Ha92] stellen in ihrer detaillierten Untersuchung der Software-Wartung in mehreren französischen Unternehmen fest, dass Mitarbeiter, die weder über eine Ausbildung in der Software-Wartung noch über eine entsprechend große Erfahrung verfügen, mit den durchzuführenden Aufgaben häufig überfordert sind.

Durch seine lange Tätigkeit im Unternehmen kennt der typische Software-Warter die Organisation und die internen Abläufe gut. Auf Grund einer relativ geringen Fluktuation betreut der typische Software-Warter über Jahre hinweg „seine“ Software. Dadurch kennt er die verschiedenen Vertreter der Fachbereiche und die Anwender der Software.

Auf Grund mangelnder Vorkenntnis, eines „gewachsenen“ Wartungsprozesses in vielen Unternehmen und dem Mangel an Schulungen über die Grundlagen und die Durchführung der Wartung hat der typische Software-Warter die Wartung durch das „trial and error“-Prinzip erlernt [TML98]. Er hat sich also durch Nachahmung und Ausprobieren verschiedener Vorgehensweisen ein sehr spezielles Wissen über die Wartung der von ihm betreuten Software aufgebaut.

4 Über welche Fähigkeiten sollte ein Software-Warter verfügen?

Nachdem die in der Praxis typische Qualifikation der Software-Warter dargelegt wurde, wird hier untersucht, über welche Qualifikationen ein Software-Warter im Idealfall verfügt. Dabei wird unterstellt, dass der Warter bereits mit der Domäne der Software ausreichend vertraut ist, so dass in dieser Hinsicht kein Ausbildungsbedarf besteht.

4.1 Grundwissen über Software-Engineering

Jeder, der an Software arbeitet und darüber kommuniziert, braucht Grundkenntnisse des Software-Engineerings, insbesondere eine sichere begriffliche Grundlage. Er muss auch mit den Standards vertraut sein, die es auf diesem Gebiet gibt. Das gilt natürlich auch für den Software-Warter.

4.2 Umgang mit Problemmeldungen

Die Bearbeitung von Problemmeldungen ist die wichtigste Tätigkeit des Software-Warters. Problemmeldungen können dabei unterschiedliche Ursachen haben. Die Anwender der Software können Problemmeldungen einbringen, um eine Funktionalität der Software abändern oder erweitern zu lassen. Die Problemmeldungen können sich auch auf Fehler in der Software beziehen, die behoben werden müssen. Problemmeldungen innerhalb der Wartungsorganisation können sich beispielsweise auf Verbesserungen der Wartbarkeit der Software beziehen.

Der Software-Warter muss in der Lage sein, Problemmeldungen zu analysieren und zu bewerten. Dabei muss der Warter auf die Vollständigkeit und die Verständlichkeit der Problemmeldungen achten.

Da sich bei der Wartung im Gegensatz zur Entwicklung die Software im Betrieb befindet, besteht ein größeres Abhängigkeitsverhältnis der Anwender bezüglich des reibungslosen Einsatzes der Software. Aus diesem Grund muss Missverständnissen bei Problemmeldungen vorgebeugt werden. Bei Unklarheiten oder unvollständigen Anfragen muss der Software-Warter gezielte Rückfragen an den Initiator der Problemmeldung

stellen können. Er muss in der Lage sein, eine Analyse der Änderungsanforderungen durchzuführen.

Alle Problemmeldungen müssen vor ihrer Umsetzung bezüglich der Auswirkungen der durchzuführenden Änderung auf die Software überprüft werden. Der Software-Warter muss also den Aufbau und die Funktionsweise der Software kennen, um eine Analyse der Auswirkungen durchführen zu können.

Anschließend muss der Software-Warter die notwendigen Änderungen der Software so durchführen, dass die Anforderungen, die in der Problemmeldung formuliert sind, erfüllt werden, bestehende Funktionalität erhalten bleibt und keine unerwünschten Nebeneffekte auftreten.

Damit ein Zugriff auf die Problemmeldungen, die im Laufe der Wartung anfallen, möglich ist, muss der Software-Warter die Verwaltung und Archivierung der Meldungen organisieren. Nur so kann in der Wartung überblickt werden, welche Problemmeldungen bereits erledigt wurden, welche momentan bearbeitet werden und welche noch offen sind.

4.3 Verwaltung der Software

Die zu wartende Software und natürlich auch die Problemmeldungen werden im Konfigurationsmanagement verwaltet. Der Software-Warter muss die Prinzipien des Konfigurationsmanagements verstehen und anwenden können. Arbeits-, Test- und Releaseumgebung müssen strikt getrennt werden, da in der Wartung häufig verschiedene Veränderungen der Software teilweise parallel durchgeführt werden müssen.

Der Software-Warter muss Werkzeuge für die Konfigurationsverwaltung einsetzen. Ein Konfigurationsmanagement ohne Werkzeugunterstützung ist bei einer Vielzahl von Artefakten, wie sie für komplexe Software typisch ist, nicht praktikabel. Darum muss der Software-Warter den Zugriff auf seine zu wartende Software mittels eines Werkzeugs für das Konfigurationsmanagement beherrschen.

4.4 Bearbeitung der Software

Zur Bearbeitung der Software muss der Software-Warter in der Lage sein, Entwurfsdokumente, also strukturelle Darstellungen der Software auf verschiedenen Abstraktionsstufen, zu lesen, zu verstehen und zu verändern. Er muss darüber hinaus in der Lage sein, fehlende oder unvollständige Entwurfsdokumente zu ergänzen, so dass diese Dokumente für die zukünftige Wartung zur Verfügung stehen.

Bei der Bearbeitung des Quellcodes ist es wichtig, dass der Warter sich an bestehende Codierrichtlinien hält. Er sollte das Prinzip des „ego-less-programming“ [IE90] verstehen und umsetzen. Wichtige Konzepte wie das Refactoring muss er kennen und beherrschen.

Der Software-Warter muss mit allen Werkzeugen und Entwicklungsumgebungen vertraut sein, die in seiner Umgebung verfügbar sind und die Wartung unterstützen.

4.5 Entwurfsmuster

In der Software-Entwicklung werden Entwurfsmuster eingesetzt. Der Software-Warter muss diese Entwurfsmuster kennen und erkennen; andernfalls wird er die logischen Strukturen durch die Wartung beschädigen oder zerstören.

4.6 Wartungsprozess

Der Software-Warter sollte bei der Umsetzung von Änderungen einen bekannten, wohldefinierten Wartungsprozess verwenden, damit Änderungen gleicher Art auf gleiche Weise durchgeführt werden. Nur so ist gewährleistet, dass keine wichtigen Schritte vergessen werden und dass der erforderliche Aufwand mit akzeptabler Genauigkeit geschätzt werden kann. Der Warter sollte mindestens ein Prozessmodell für die Wartung kennen und in der Lage sein, es den Gegebenheiten in seinem Unternehmen anzupassen.

Zum Prozess gehört auch das Prinzip der getrennten Arbeitsumgebungen: Änderungen werden nicht am Produktivsystem durchgeführt, sondern in einer abgeschotteten Wartungsumgebung. Tests der Software erfolgen in einer Testumgebung. Erst zum Schluss werden die geänderten Komponenten in die Produktionsumgebung überführt.

Da bei der Software-Wartung verschiedene Personengruppen, z. B. Kunden, Anwender, Supporttechniker, Manager, Subunternehmer, Fachexperten usw., beteiligt sind, muss der Software-Warter mit diesen Personengruppen kommunizieren und die Wartung abstimmen können. Dazu muss er die Schnittstellen und Rollen im Wartungsprozess kennen und verstehen – umso mehr, wenn das Unternehmen und die Wartung über verschiedene Standorte verteilt sind.

4.7 Metriken

Wie in allen anderen Bereichen des Software-Engineerings brauchen wir auch in der Wartung Metriken, um unsere Arbeit und ihre Auswirkungen zu quantifizieren, die Zusammenhänge zu verstehen und den Prozess zu verbessern. Der Warter sollte darum das Prinzip der Metriken verstehen und geeignete Metriken anwenden und interpretieren können.

Für den Einsatz der Metriken kommen verschiedene Aspekte in Frage: Zunächst sind die Problemmeldungen zu erfassen und zu verfolgen, so dass Stand und Trend des Änderungsbedarfs sichtbar werden. Metriken über den Zustand der Software vor und nach Änderungen können strukturelle Probleme der Software anzeigen. Schließlich dienen Aufwandsmetriken dazu, die Kosten der Wartung zu erfassen und die Genauigkeit der Aufwandsschätzungen zu verbessern.

4.8 Qualitätssicherung und Prozessqualität

Der Software-Warter hat wichtige Funktionen zur analytischen Qualitätssicherung; umso mehr, wenn es in der Wartung keine spezielle Rolle für die Qualitätssicherung gibt. Er muss die erprobten Inspektionstechniken kennen und in der Lage sein, einen systematischen Test vorzubereiten und durchzuführen. Nach Korrekturen und Erweiterungen muss die geänderte Funktionalität getestet werden.

In der Software-Wartung spielt der Regressionstest eine wichtige Rolle. Der Software-Warter muss mit diesem Verfahren vertraut sein und für die entsprechende Organisation der Daten und der Dokumente sorgen.

Fragen der Prozessqualität liegen auf einer anderen Ebene. Wo in der Entwicklung eine definierte Prozessqualität erreicht wurde, darf die Wartung diese Errungenschaft nicht kompromittieren. Die Software-Warter müssen sicherstellen, dass die Erfüllung der Anforderungen an den Prozess auch in der Wartung sichergestellt ist.

4.9 Wiederverwendung

Wenn eine Software(-Komponente) wiederverwendet werden soll, sind meist Änderungen notwendig, die darauf abzielen, Bindungen an eine spezielle Verwendung zu beseitigen. Der Software-Warter sollte mit den Prinzipien der Wiederverwendung vertraut sein, um solche Änderungen möglichst selbständig durchführen zu können.

Umgekehrt sollte er auch erkennen, wenn eine Problemmeldung durch die Wiederverwendung einer vorhandenen Komponente elegant beantwortet werden kann.

4.10 Aufwandsschätzung für Wartungsarbeiten

Für die Planung der Wartung und die Priorisierung der Problemmeldungen braucht das Change Control Board zuverlässige Aufwandsschätzungen. Der Software-Warter muss diese erstellen können.

4.11 Dokumentation

Da sich die Wartung einer Software über Jahre, oft Jahrzehnte erstreckt, wird eine umfassende Dokumentation benötigt, die nicht nur den Anfangszustand, sondern auch alle Veränderungen spiegelt. Der Software-Warter muss wissen, welche Informationen in welchen Dokumenten abgelegt sind und wie er darauf zugreifen kann. Er muss die Dokumente bei Bedarf ergänzen und fehlende Dokumente erstellen können.

4.12 Re-Engineering

Wenn die Software unwartbar geworden ist, muss entschieden werden, ob sie ganz oder in Teilen ersetzt oder durch Re-Engineering verbessert werden soll. Der Warter sollte

diese Entscheidung vorbereiten. Findet ein Re-Engineering statt, so leistet der Warter wichtige Beiträge durch Reverse Engineering und Bereinigung der Strukturen.

Häufig existiert zu einer Software keine aktuelle Dokumentation (mehr). Der Software-Warter muss in der Lage sein, aus den vorhandenen Quellen, typisch dem Quellcode und veralteten Dokumenten, abstraktere Modelle abzuleiten, die Aufschluss über die Architektur der Software und das Zusammenspiel ihrer Teile liefern (Reverse Engineering).

Die ebenfalls zum Reverse Engineering gerechnete Wiedergewinnung der Anforderungen geht über die Möglichkeiten des Warters hinaus.

5 Ein Themenkatalog für die Ausbildung der Software-Warter

Da sich die Ausbildung und Zertifizierung an Software-Warter aus der Industrie richtet, müssen bei der Gestaltung zwei Rahmenbedingungen beachtet werden:

1. Die Ausbildung darf keine unrealistischen Anforderungen an die Vorkenntnisse stellen. Die Teilnehmer haben in der Regel keine einschlägige Berufsausbildung.

2. Damit die Ausbildung in den Unternehmen akzeptiert wird, darf sie nicht zu lange dauern. Kurse müssen auf zwei bis drei Tage beschränkt sein. Wie beim Certified Tester sind aber gestufte Kurse denkbar, die nacheinander absolviert und mit Zertifikaten abgeschlossen werden können.

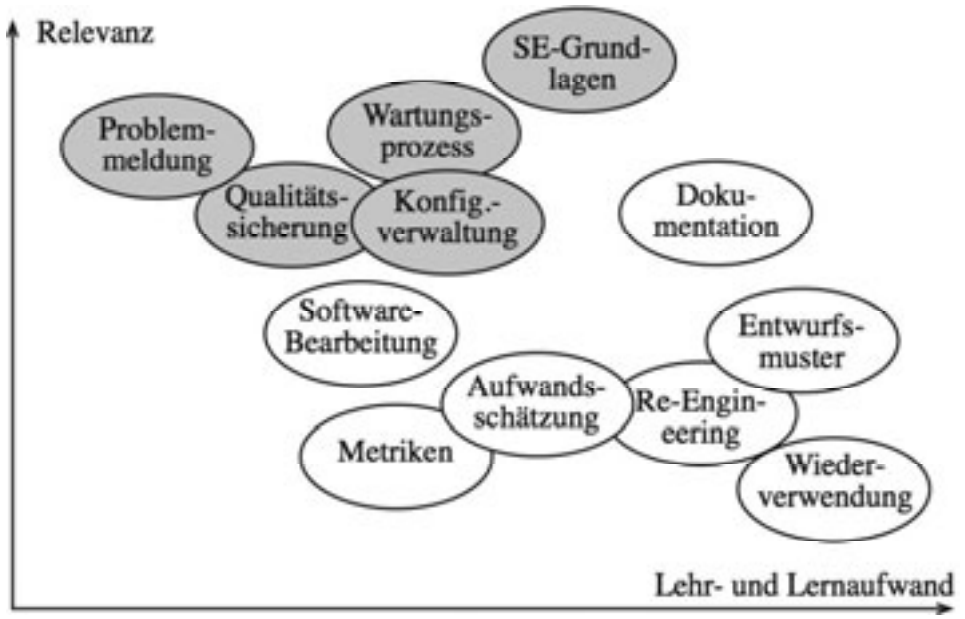

Abbildung 1: Fähigkeiten des Software-Warters nach Relevanz und Lehr-/Lernaufwand

Entsprechend gilt es, eine Teilmenge der in Kapitel 4 aufgezählten Kenntnisse und Fähigkeiten zu identifizieren, die sich in kurzer Zeit vermitteln lassen und für die tägliche Arbeit des Warters besonders nützlich sind. In Abbildung 1 sind die in Kapitel 4 identifizierten Fähigkeiten eines idealen Software-Warters in einer Grafik zusammengefasst. Die Fähigkeiten sind dabei nach Relevanz für die industrielle Software-Wartung und nach dem Aufwand, diese Fähigkeiten in einer Ausbildung zu lehren und zu erlernen, sortiert.

In die Ausbildung für die Software-Wartung werden diejenigen Themen aufgenommen, die große Relevanz aufweisen und mit möglichst geringem Aufwand gelehrt und gelernt werden können. In Abbildung 1 sind diese Themen grau hinterlegt.

5.1 Grundkurs Software-Wartung

Voraussetzung für alle anderen Themen ist eine begriffliche Grundlage und ein elementares Verständnis für Software-Engineering. In einem dreitägigen Kurs steht dafür nicht mehr als ein Tag zur Verfügung. Das ist allerdings das absolute Minimum; wenn wir die Zeit für dieses wichtige Thema weiter reduzieren, gefährden wir den Erfolg des ganzen Kurses.

Die in Abbildung 1 grau hinterlegten Themen bilden das weitere Programm des Grundkurses. Wir können (und müssen) dafür jeweils einen halben Tag vorsehen. Das resultierende Programm ist in Abbildung 2 skizziert. Es wird durch eine Prüfung in der üblichen Form (also durch Ankreuzen und Ergänzen eines Fragebogens) komplettiert.

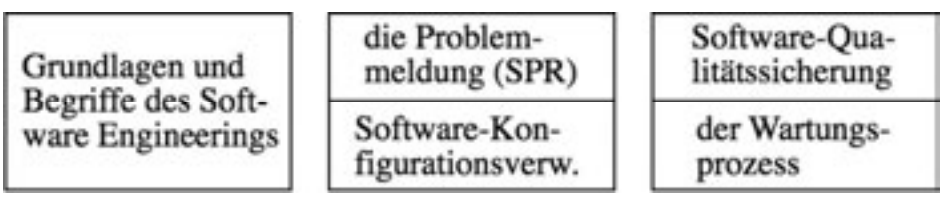

Abbildung 2: Grundkurs Software-Wartung

Im Block 2 (Problemmeldung, SPR) wird der Ablauf einer Wartung behandelt. Die Teilnehmer lernen, wie die Wartung angestoßen und durch die Instanzen verfolgt wird, wer welche Zuständigkeiten hat und welche Dokumente beteiligt sind.

Der Block 3 behandelt die Software-Konfigurationsverwaltung. Dabei werden auch Werkzeuge vorgestellt und diskutiert. Das Prinzip der Arbeitsumgebungen kann in diesem Zusammenhang ebenfalls vermittelt werden.

Techniken der analytischen Qualitätssicherung, die für die Wartung geeignet sind, werden im Block 4 behandelt. Dazu gehört der Test (mit dem Schwerpunkt Regressionstest), aber auch die Inspektion, wobei Verfahren im Vordergrund stehen, die anders als das Review mit wenigen Personen auskommen.

Block 5 (Wartungsprozess) verbindet schließlich die einzelnen Themen und fügt sie zu einem kompletten Bild der Wartung zusammen.

5.2 Ergänzungskurse Software-Wartung

Während der Inhalt des Grundkurses kaum umstritten sein dürfte, ist durchaus nicht klar, wie eine Fortsetzung aussehen sollte. Wir sehen, von Abbildung 1 ausgehend, zwei Möglichkeiten:

A Alle restlichen Themen werden in einem Fortsetzungskurs behandelt.

B Die restlichen Themen werden in zwei Gruppen aufgeteilt, beispielsweise in die Gruppe „Technik der Software-Wartung“ (Software-Bearbeitung, Entwurfsmuster, Dokumentation und Metriken) und in die Gruppe „Wartungsmanagement“ (Aufwandsschätzung, Wiederverwendung, Re-Engineering, Prozessqualität).

Da die Lösung A nur eine sehr oberflächliche Behandlung der Themen erlaubt, ziehen wir die Lösung B vor. Im Kurs „Technik der Software-Wartung“, der auf eine bestimmte Programmiersprache zugeschnitten sein kann, geht es um die konkreten Änderungen und Erweiterungen, die der Warter durchführt, und um die Werkzeuge, die er dabei einsetzt. Es sollte möglich sein, dabei auch praktische Übungen in den Kurs einzubauen.

Der Kurs „Wartungsmanagement“ richtet sich vor allem an Software-Warter, die eine Führungsrolle übernehmen, also beispielsweise die Leitung eines größeren Wartungsprojekts.

Beide Kurse werden analog zum Grundkurs mit einer Prüfung abgeschlossen.

6 Zusammenfassung und Ausblick

Das Ausbildungs- und Qualifizierungsangebot zum Certified Tester ist in der Praxis auf großes Interesse gestoßen. Auch auf anderen Gebieten des Software-Engineerings, dem Projektmanagement, der Software-Architektur und der Anforderungsanalyse, wurden ähnliche Programme entwickelt. Natürlich sind solche Kurse keine Alternative und kein Ersatz für eine solide fachliche Ausbildung; aber da viele Menschen in Informatik-Berufen eine solche Ausbildung nicht erhalten haben und nie erhalten werden, ist das Konzept geeignet, die Situation in der Praxis deutlich zu verbessern.

In diesem Artikel wurde ein Ansatz für eine Ausbildung und Zertifizierung derer vorgestellt, die sich um die Wartung der Software kümmern. Nach allen Statistiken ist das die

große Mehrheit der Softwareleute. Auf der Grundlage der Erfordernisse und Rahmenbedingungen wurde eine Liste von Themen aufgestellt, und die Themen wurden entsprechend ihrer Relevanz und entsprechend dem Aufwand für die Vermittlung priorisiert.

Wenn das Konzept diskutiert und verbessert ist und darüber ein Konsens erzielt wurde, müssen die Themen weiter präzisiert und mit Inhalten aufgefüllt werden. Ebenso müssen Prüfungsmodalitäten für die Zertifizierung festgelegt werden.

Damit aus der Ausbildung für die Software-Wartung tatsächlich eine Zertifizierung wie beim Certified Tester entstehen kann, muss eine nationale oder internationale Organisation die Trägerschaft übernehmen, also Schulungsprogramme definieren und standardisieren. Dazu gehören auch die Prüfungen.

Literaturverzeichnis

[GT07] GTB: German Testing Board. http://german-testing-board.info. November 2007.

[Ha92] Haziza, M. et al.: Software Maintenance: An Analysis of Industrial Needs and Constraints. In: Conference on Software Maintenance, 1992. Proceedings. IEEE Computer Society Press. S. 18-26.

[IE90] IEEE Std 610.12 (1990): IEEE Standard Glossary of Software Engineering Terminology. IEEE Standards Association.

[IR07] IREB: International Requirements Engineering Board. http://certified-re.de. November 2007.

[IS07a] ISTQB: International Software Testing Qualifications Board. http://istqb.org. November 2007.

[IS07b] ISQI: Zertifizierungs- und Ausbildungsprogramm: Certified Tester, Certified Professional for Software Architecture, Certified Professional for Project Management, Certified Professional for Requirements Engineering. http://www.isqi.org. November 2007.

[LO04] Ludewig, J.; Opferkuch, S.: Software-Wartung - eine Taxonomie. In: Softwaretechnik-Trends, Band 24 Heft 2, Gesellschaft für Informatik, Mai 2004, Seiten 35-36.

[Op07] Opferkuch, S.: Benötigen wir einen „Certified Maintainer“? Softwaretechnik-Trends, Band 27, Heft 2, Gesellschaft für Informatik, ISSN 0720-8928, Mai 2007, Seiten 36-37.

[SL04] Spillner, A.; Linz, T.: Basiswissen Softwaretest: Aus- und Weiterbildung zum Certified Tester; Foundation Level nach ASQF- und ISTQB-Standard. dpunkt-Verlag, 2. Überarb. Auflage, 2004.

[TML98] Taylor M. J.; Moynihan, E. P.; Laws, A.: Training for Software Maintenance. In: Journal of Software Maintenance: Research and Practice. Band 10, Heft 6, 1998. S. 381-393.

[ZSG79] Zelkowitz, M. V.; Shaw, A. C.; Gannon, J. D.: Principles of Software Engineering and Design. Prentice-Hall, Inc., Englewood Cliffs, N.J., USA, 1979.