

Staatsexamen 66116 / 2016 / Frühjahr

Thema 1 / Teilaufgabe 2 / Aufgabe 1

(Softwaresysteme: Begriffe und Konzepte)

Stichwörter: Pflichtenheft, Softwaremaße, Evolutionäre Softwaremodelle, Versionsverwaltungssoftware, Funktionale Anforderungen, Nicht-funktionale Anforderungen, Kontinuierliche Integration (Continuous Integration)

(a) Welche Kriterien werden bei der Zielbestimmung im Pflichtenheft unterschieden?

Im Rahmen der Zielbestimmung werden die Ziele des Produktes in einer Art Prioritätenliste exakt eingegrenzt, die in drei weitere Kategorien gegliedert ist, die als **Muss-, Wunsch- und Abgrenzungskriterien** bezeichnet werden.

Musskriterien Zu den Musskriterien zählen die Produktfunktionen, die für ein Funktionieren des Produktes unabdingbar sind. Ohne ihre Umsetzung geht einfach nichts, deshalb müssen sie erfüllt werden.

Wunschkriterien Die nächste Kategorie, die Wunschkriterien, sind zwar entbehrlich, wurden aber ausdrücklich vom Auftraggeber gefordert. Ihre Umsetzung im Produkt ist also genauso eine Pflicht, wie die der ersten Kategorie, das Produkt würde aber auch ohne ihre Implementierung seinen Betrieb korrekt ausführen.

Abgrenzungskriterien Die letzte Kategorie beschreiben die Abgrenzungskriterien. Sie sollen die Grenzen des Produktes klarmachen und sind von daher beinahe wichtiger als die beiden ersten Fälle denn sie hindern die Entwickler daran, über alle Ziele hinauszuschießen.

<https://st.inf.tu-dresden.de/SalesPoint/v3.2/tutorial/stepbystep2/pflh.html>

(b) Nennen Sie drei Einsatzziele von Softwaremaßen.

Aus Baumann K. (1997) Softwaremaße. In: Unterstützung der objektorientierten Systemanalyse durch Softwaremaße. Beiträge zur Wirtschaftsinformatik, vol 23. Physica, Heidelberg. https://doi.org/10.1007/978-3-662-13273-9_2

Der Einsatz von Softwaremaßen kann vielfältige Vorteile mit sich bringen. Folgende Ziele können mit der Anwendung von Softwaremaßen verfolgt werden:

- Durch eine Bewertung kann überprüft werden, inwieweit einmal gestellte *Qualitätsanforderungen* erfüllt wurden.
- Eine Bewertung bereits erstellter oder noch zu erstellender Softwareprodukte kann Hilfestellung bei Entscheidungen, die für die *Projektplanung* oder das Entwicklungsmanagement zu treffen sind, leisten. So können Softwaremaße zur Einschätzung des notwendigen Aufwands, der zu erbringenden Kosten und des Personalbedarfs herangezogen werden.

- Ein möglichst frühzeitiges *Aufzeigen von Schwachstellen im Systemdesign*, die Beurteilung der Auswirkungen neuer Techniken oder Tools auf die Produktivität der Entwickler oder auf die Qualität des entwickelten Produkts sowie die Überprüfung der Einhaltung festgelegter Designstandards sind weitere mögliche Einsatzfelder für Softwaremaße.
- Darüber hinaus ist der Einsatz von Softwaremaßen *Teil umfassender Konzepte zum Softwarequalitätsmanagement*. Softwaremaße bzw. die zugehörige Meßergebnisse tragen zunächst dazu bei, die betrachteten Produkte oder Prozesse besser zu verstehen oder hinsichtlich interessierender Eigenschaften zu bewerten.

https://link.springer.com/chapter/10.1007%2F978-3-662-13273-9_2

(c) Welche Arten von Softwaremaßen werden unterschieden?

Die Unterscheidung nach dem Messgegenstand ergibt folgende Arten von Maßen:

Externe Produktmaße. Diese beschreiben Merkmale des Produkts, die von außen sichtbar sind, wenngleich eventuell nur indirekt. Dazu gehören insbesondere Maße für Qualitätsattribute wie die Wartbarkeit, die Zuverlässigkeit, die Benutzbarkeit, die Effizienz etc.

Interne Produktmaße. Basis für die Charakterisierung externer Produktattribute sind interne Attribute, die sich zumindest zum Teil besser messen lassen. Hierzu gehören z. B. die Größe, die Modularität und die Komplexität.

Prozessmaße. Diese charakterisieren Eigenschaften des Produktionsprozesses, z. B. die typische Produktivität, Kostenaufteilung auf bestimmte Arbeiten, Dauer von Arbeitsschritten etc.

Ressourcenmaße. Diese charakterisieren Eigenschaften der im Prozess verwendeten Ressourcen, z. B. die Auslastung von Rechnern, die typische Produktivität und Fehlerrate eines bestimmten Ingenieurs etc.

<http://page.mi.fu-berlin.de/prechelt/swt2/node5.html>

Was sind die 3 Arten der (einfachen) Maße zur Messung von Software?

<https://quizlet.com/de/339799466/softwaretechnik-theorie-flash-cards/>

- Größenorientiert/nach Größe
- Funktionsorientiert/nach Funktion
- Objektorientiert/nach Objekt

(d) Nennen Sie drei Merkmale evolutionärer Softwareentwicklung.

Wie für jede Form der Software-Entwicklung stellt sich die Frage nach einem geeigneten methodischen Ansatz. Generell lässt sich der Prozess der Software-Entwicklung in die folgenden drei Abschnitte gliedern:

- (i) von der Problembeschreibung bis zur Software-Spezifikation (auch Problem-analyse, Systemanalyse, Requirements Engineering bzw. frühe Phasen der Software-Entwicklung ge- nannt),
- (ii) die Software-Entwicklung im engeren (technischen) Sinn,
- (iii) die Integration der Software in den Anwendungszusammenhang.

Diese Einteilung gilt ebenso für die evolutionäre Software-Entwicklung, allerdings wird hier der Schwerpunkt anders gelegt. Steht beim klassischen Ansatz in der Regel die Software-Entwicklung im engeren Sinn im Mittelpunkt, so werden beim evolutionären Ansatz alle drei Abschnitte möglichst gleichgewichtig und im Zusammenhang betrachtet. Daraus resultiert eine natürliche Tendenz zu einer zyklischen Vorgehensweise.

Daneben gibt es weitere Querbezüge: Orientiert sich die Software-Entwicklung an einem festen Ziel, verfolgt dieses jedoch durch schrittweises Erweitern zunächst unvollkommener Teillösungen, so wird dieses Vorgehen inkrementell genannt. Unfertige Versuchsanordnungen oder Teilsysteme, die am Beginn einer solchen Entwicklung stehen, werden oft als Prototypen bezeichnet. Wählt man Prototyping als Vorgehensweise, so kann entweder das spätere Produkt inkrementell aus einem oder mehreren Prototypen weiterentwickelt werden oder man setzt nach einer (Wegwerf-) Prototypphase neu auf. Beides sind Formen evolutionärer Entwicklung.

<http://waste.informatik.hu-berlin.de/~dahme/edidahm.pdf>

(e) Nennen Sie drei Vorteile des Einsatzes von Versionsverwaltungssoftware.

- (i) Möglichkeit, auf ältere Entwicklungsversionen zurückzukehren, wenn sich die aktuelle als nicht lauffähig bzw. unsicher erwiesen hat (z. B. git revert)
- (ii) Möglichkeit, dass mehrere Entwickler gleichzeitig an ein- und denselben Softwareprojekt arbeiten und Möglichkeit, dass ihre Entwicklungen durch das Versionsverwaltungssystem zusammengeführt werden können (z. B. git merge)
- (iii) Möglichkeit, den Zeitpunkt oder den Urheber eines Softwarefehlers ausfindig zu machen (z. B. git blame)

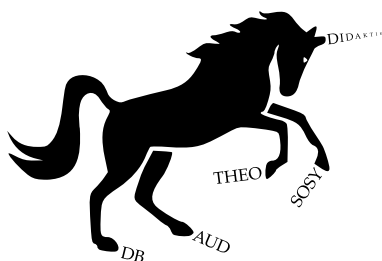
(f) Worin besteht der Unterschied zwischen funktionalen und nicht-funktionalen Anforderungen?

Funktionale Anforderung: Anforderung an die Funktionalität des Systems, also „Was kann es?“

Nicht-funktionale Anforderung: Design, Programmiersprache, Performanz

(g) Was verbirgt sich hinter dem Begriff „Continuous Integration“?

Continuous Integration: Das fertige Modul wird sofort in das bestehende Produkt integriert. Die Integration erfolgt also schrittweise und nicht erst, wenn alle Module fertig sind. Somit können auch neue Funktionalitäten sofort hinzugefügt werden (*rightarrow* neue Programmversion).



Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.



Diese Materialsammlung unterliegt den Bestimmungen der Creative Commons Namensnennung-Nicht kommerziell-Share Alike 4.0 International-Lizenz.

Hilf mit! Die Hermine schafft das nicht alleine! Das ist ein Community-Projekt. Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net. Der \LaTeX -Quelltext dieses Dokuments kann unter folgender URL aufgerufen werden: <https://github.com/hbschlang/lehramt-informatik/blob/main/Staatsexamen/66116/2016/09/Thema-1/Teilaufgabe-2/Aufgabe-1.tex>