Software vergleichen

Andrea Herrmann
Infoman AG
D-70563 Stuttgart
andrea.herrmann@infoman.de

Motivation

Software wird nur selten auf der grünen Wiese neu entwickelt. Darum werden Anforderungen in der Praxis meist auch nicht auf dem weißen Blatt neu und vollständig beschrieben. Stattdessen wird oft eine auf dem Markt existierende Software ausgewählt und angepasst oder ein Altsystem weiterentwickelt. In dieser Situation gibt es zwei praktische Herausforderungen: Erstens möchte man aus einer Menge am Markt erhältlicher Softwareprodukte dasjenige auswählen, das die angestrebten Anforderungen möglichst gut erfüllt (Gap-Analyse). Im zweiten Schritt möchte man diejenigen Anforderungen detaillierter beschreiben und priorisieren, die in Bezug auf die gewählte Software oder auf das Altsystem Änderungen bedeuten, also die Delta-Anforderungen [1]-[5].

Sowohl bei der Gap-Analyse als auch der Delta-Analyse werden zwei Software-Systeme miteinander verglichen. Jedoch unterscheiden sich die Ziele der beiden Tätigkeiten und daher auch die Anforderungen an die Bewertung von Gap oder Delta:

- Die *Gap-Analyse* fragt: „Welchen Anteil des Nutzens, den ich mir wünsche, werde ich bekommen?“ Eines oder mehrere Produkte werden mit den Anforderungen verglichen. Trotz des Namens interessiert eher die Ähnlichkeit als der Unterschied, was sich auch in dem Begriff „similarity score“ ausdrückt, einem Wert, der misst, wie gut ein existierendes Produkt den Anforderungen ähnelt. Ziel der Gap-Analyse ist die Maximierung des Nutzens bei der Auswahl eines Produktes.
- Die *Delta-Analyse* fragt: „Wie viel muss ich am Altsystem mit wie viel Aufwand ändern?“ Hier werden die Anforderungen mit dem Altsystem verglichen. Ziel ist die Kosten-Minimierung.

Dieser Beitrag diskutiert für beide Tätigkeiten, wie man die Unterschiede zwischen Software quantifizieren kann, um die zu treffenden Entscheidungen zu unterstützen – nämlich die Produktauswahl und die Priorisierung von Deltas.

Wir vergleichen drei, sowohl für die Gap-Analyse als auch für die Delta-Analyse:

1. *Grafische Darstellung* von Unterschieden anhand farblich markierter UML-Diagramme
2. *Heuristiken* mit Hilfe von Kategorien
3. Quantifizierung von Unterschieden durch *gewichtete Summenbildung*

Stand der Forschung

Die Gap- und Delta-Analyse werden in der Literatur unter diversen Begriffen untersucht, insbesondere „package selection“ oder „COTS selection“ und erst neuerdings auch unter dem Begriff „Delta“ [1]-[5]. Die Produktauswahl wird oft behandelt als ein Spezialfall eines „multiple criteria decision-making problem“, so dass hier entsprechende Methoden anwendbar sind. Jadhav und Sonar [JS09] haben eine umfassende Literaturübersicht über das Gebiet erstellt und nennen dabei fünf Kategorien von Methoden: AHP, feature analysis, weighted average sum, expert system, fuzzy based approach. Die Analyse grafischer Darstellungen und Heuristiken haben sie nicht mit einbezogen.

Nirgends jedoch wird bisher zwischen Gap- und Delta-Analyse unterschieden.

Grafische Darstellung

Für den grafischen Vergleich wird gerne die standardisierte UML verwendet. Man modelliert erst die Version des Systems, gegen die verglichen werden soll. Dies ist bei der Gap-Analyse die Anforderungsspezifikation und bei der Delta-Analyse das Altsystem. Im grafischen Vergleich kann man mit Farben die Unterschiede hervorheben, beispielsweise schwarz = bleibt so, grün = neu, rot = ausblenden, gelb = ändern. Zur Quantifizierung kann eine Abstandsmetrik [GZJ07] oder Function Points [SW08] verwendet werden. So anschaulich auch die entstehenden Grafiken sein mögen, verursacht die Modellierung doch einen höheren Aufwand als die beiden folgenden Methoden.

Heuristiken

Heuristiken sind vereinfachte Entscheidungsverfahren. Man kann beispielsweise die drei Kategorien der Kano-Klassifikation von Anforderungen verwenden: Anforderungen der Kategorie „Must have/ basic“ sind Stand der Technik und müssen im System enthalten sein, bedeuten aber keine Eigenschaft, die das Produkt gegenüber anderen hervorhebt. „Performance“-Anforderungen machen den Kunden umso zufriedener, je besser sie erfüllt sind. „Delighters/ excitement”-Anforderungen sind unerwartete Eigenschaften, die dem Kunden sehr viel Nutzen bringen. Als Heuristik für die Gap-Analyse und Produktauswahl könnte man folgende wählen:

1. Ein Produkt, das nicht alle Basic-Anforderungen erfüllt, kommt nicht auf die Short List.
2. Die Performance-Anforderungen bestimmen den Similarity Score.
3. Delighter-Anforderungen geben bei gleichem Similarity Score den Ausschlag.

Für die Delta-Analyse braucht man andere Kategorien, die nicht auf Nutzen, sondern auf Kosten basieren.

Gewichtete Summenbildung

Bei der gewichteten Summenbildung vergleicht man Alternativen dadurch, dass man für jede Alternative i beurteilt, wie gut sie eine Reihe von Kriterien (hier: die Anforderungen) erfüllt. Hierbei nimmt man die Benutzerperspektive ein. Man summiert diese Erfüllungsgrade x_{ij} auf, wobei jede Anforderung j ein Gewicht w_j erhält. Bei der Gap-Analyse wird ein Produkt i bewertet durch den Gesamterfüllungsgrad, also $\Sigma_j x_{ij} w_j / \Sigma_j w_j$. Die Gewichte werden entsprechend der Nützlichkeit vergeben. (Wie nützlich ist diese Anforderung?) Ein Erfüllungsgrad 80% bedeutet, die Anforderung ist teilweise erfüllt, was 80% so nützlich ist als wäre die Anforderung vollständig erfüllt. Man wird nun von allen Produkten dasjenige wählen, das den höchsten Gesamterfüllungsgrad aufweist.

Die Delta-Analyse geht ähnlich vor, allerdings mit folgenden Unterschieden: Die Gewichte der Anforderungen messen ihren Anteil am Gesamten (gemessen in Kosten oder Erstellungsaufwand). Ein Erfüllungsgrad von 80% bedeutet, dass 20% des Aufwands noch zu erbringen sind, um die Anforderung ganz zu erfüllen. Hier ist eher eine technische Sicht nötig, um Kosten zu beurteilen zu können. Das Delta Δ_{ij} einer Anforderung j ist $\Delta_{ij} = |1- x_{ij}|$. Das Gesamt-Delta eines Systems i ist dann $\Sigma_j \Delta_{ij} w_j / \Sigma_j w_j$.

Bei der Releaseplanung möchte man nun z.B. diejenigen Anforderungen j als erste implementieren, die das beste Kosten-Nutzen-Verhältnis aufweisen. Hier kann man so vorgehen, dass man für jede Anforderung j das Δ_{ij} aus der Delta-Analyse durch das Gewicht w_j aus der Gap-Analyse teilt und die Anforderungen mit dem größten Quotienten am höchsten priorisiert.

Eine wichtige Frage ist noch die nach der Skalierbarkeit dieser Methoden. Die Bewertung des Erfüllungsgrads ist gar nicht einfach, wenn die Anforderungen eine andere Struktur aufweisen als die Funktionalitäten eines Produkts, also beispielsweise nicht jedem Nutzungsszenario genau ein Menüpunkt oder Reiter der Software entspricht. Dann muss man für jede Anforderung prüfen, wo und wie sie erfüllt wird. Hierdurch entsteht ein Aufwand, der proportional ist zum Produkt aus der Anzahl der Anforderungen und der Anzahl der Produktfunktionen. Man kann die Komplexität von Anforderungen oft verringern, indem man sie hierarchisch verfeinert und zunächst nur die groben Anforderungen betrachtet. Allerdings nutzt diese Vereinfachung hier wenig, denn: Wenn auf einem groben Niveau kein Gap besteht – z.B. weil sowohl die Anforderungen als auch das Produkt ein Abrechnungsmodul enthalten – so folgt daraus nicht auch, dass im Detail keine Gaps bestehen. Man muss also doch die gesamte Anforderungshierarchie auf Gaps (und Deltas) hin analysieren.

Diskussion

Dieser Artikel unterscheidet zum ersten Mal zwischen der nutzenorientierten Gap-Analyse und der kostenorientierten Delta-Analyse. Diese Unterscheidung ist neu und wirft ein neues Licht auf bestehende Methoden. Betrachtet man beispielsweise SAAM [6], [7] (das bewertet, wie gut Szenarien durch alternative Architekturentwürfe erfüllt werden), so mischt sich hier eine nutzenorientierte Bewertung der Anforderungserfüllung, die zur Gap-Analyse gehört, mit einer technischen Sicht, die zur Delta-Analyse gehört.

Referenzen

[1] C. Ebert: „Systematisches Requirements Engineering und Management - Anforderungen ermitteln, spezifizieren, analysieren und verwalten“. dpunkt.verlag, 2. Auflage, 2008

[2] C. Rupp, D. Schüpferling, C. Pikalek: Deltarequirements– Auf den Spuren der Projektrealität. Informatikspektrum, Band 32, Heft 2, April 2009

[3] A. Herrmann, A. Wallnöfer, B. Paech: Specifying Changes Only – A Case Study on Delta Requirements. REFSQ - Workshop on Requirements Engineering for Software Quality, Foundations of Software Quality, Springer, 2009, S. 45-58

[4] A. Herrmann, S. Schier: Die Spezifikation von Delta-Anforderungen. OBJEKTspektrum Nr. 6, November/ Dezember 2010, S. 10-13

[5] A. Herrmann: Den Nebel lichten: unspezifizierte Software spezifizieren und weiterentwickeln (Tutorial). iqnite, Düsseldorf 2011

[JS09] A.S. Jadhav, R.M. Sonar: Evaluating and selecting software packages: A review. Information and Software Technology 51, 2009, S. 555–563

[GZJ07] J. Gao, L. Zhang, W. Jiang: Procuring Requirements for ERP Software Based on Semantic Similarity. International Conference on Semantic Computing ICSC 2007, S. 61-70

[SW08] J. Sheng, B. Wang: Evaluating COTS Components Using Gap Analysis. The 9th International Conference for Young Computer Scientists, ICYCS 2008, S. 1248-1253

[6] P. Clements, L. Bass, R. Kazman, G. Abowd: Predicting Software Quality by Architectural-Level Evaluation. Fifth International Conference on Software Quality, Oct 1995

[7] R. Kazman, G. Abowd, L. Bass, P. Clements: Scenario-based analysis of software architecture. IEEE Software 13(6), Nov 1996, 47-55