Heinrich C. Mayr, Martin Pinzger (Hrsg.): INFORMATIK 2016,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2016 1941

Planung von Migrationsprojekten

Harry M. Sneed[1]

Abstract: Dieses halbtägige Tutorial behandelt die Planung einer Softwaremigration. Viele Unternehmen in Österreich und anderswo haben alte Softwaresysteme die sie gerne in die neue Welt hinüberretten möchten. Es stehen ihnen dazu einige Alternativen mit unterschiedlichen Kosten und Nutzen bevor. Die Alternativen reichen von einer kompletten Neuentwicklung bis zu einer Kapselung der alten Komponente. Die Auswahl der richtigen Alternative ist ausschlaggebend für den Erfolg der Migration. Wählen sie die falsche Alternative wird die Migration scheitern oder wesentlich mehr kosten als geplant. Der Schlüssel zur Auswahl der richtigen Alternative ist eine ausführliche Kosten/Nutzen Analyse. Sie müssen ihre Altsysteme messen um festzustellen wie groß, wie komplex und wie gut sie sind. Davon hängt es ab was sie damit anfangen können. Als nächstes werden die Kosten jeder Alternative auf der Basis der Größe, Komplexität und Qualität geschätzt. Danach wird der Nutzen einer jeden Alternative aufgrund einer Nutzwertanalyse ermittelt. Schließlich wird entschieden welche Alternative man vorzieht. Der Entscheidungsprozess wird erläutert an Hand von Beispielen aus der industriellen Praxis. Die Teilnehmer des Tutoriums erhalten automatisierte Werkzeuge um den Entscheidungsprozess zu unterstützen – Werkzeuge für die Analyse der Altsysteme sowie Werkzeuge für die Kostenschätzung der Migrationsalternative. Die Arbeitsweise und Ergebnisse der Werkzeuge werden in einer Fallstudie demonstriert.

Keywords: Legacy-Systeme, Softwaremigration, Softwaremessung, Aufwandsschätzung, COCOMO, Function-Point, Data-Point, Object-Point, Migrationskostenanalyse, Risikoanalyse, Migrationsnutzen, Migrationsfallstudien.

1 Themenhintergrund

Etliche Anwendungsbetriebe stehen vor dem Problem sich von ihrer Altlasten zu befreien. Es kann sein, dass sie ihre alten Daten migrieren wollen, ihren alten Code ablösen möchten oder wieder von vorne mit neuen Systemen anfangen wollen. Was das zweite Alternativ anbetrifft, haben sie die Wahl zwischen verschiedenen Befreiungsstrategien, darunter:

- Die automatische Konvertierung der alten Codes
- Die manuelle Reimplementierung des alten Codes
- Die Kapselung der alten Code-Komponente und
- Die Neuentwicklung der Codebasis

[1] Technische Universität Dresden, Institut für Softwaretechnik, ZT-Prentner-IT GmbH, Wien-Kagran, Harry.Sneed@T-Online

Welche Strategie sie wählen, hängt von den Kosten und Nutzen sowie von den Risiken der jeweiligen Strategie ab. D.h. sie müssen im Vorfeld der Migration eine Kosten/Nutzen/Risiko Analyse durchführen. Alle drei Analyseprojekte können neben einander stattfinden. Die Ergebnisse werden am Ende zusammengefasst und mit einander verglichen um zu einer Entscheidung zu kommen welches Alternativ das größte „Return on Investment" verspricht. Das RoI wird hier mit folgender Wirtschaftlichkeitsrechnung

ROI = (Nutzen – Kosten) / (Kosten * Risiken)

errechnet. Irrationale, politische Entscheidungen über die Zukunft der betrieblichen Softwaresysteme sind möglichst zu vermeiden. Daher der quantitative Ansatz. Es geht darum einen rationalen, nachvollziehbaren Entscheidungsprozess einzuführen. In diesem eintägigen Tutorial wird an Hand von Fallbeispielen aus der langjährigen Migrationspraxis des Referenten vermittelt wie man bei einer derartigen Voruntersuchung vorzugehen hat.

2 Tutorium Inhalt

Das Tutorium behandelt die drei Schienen einer Wirtschaftlichkeitsanalyse:

- Risikoanalyse
- Kostenschätzung
- Nutzenanalyse.

2.1 Risikoanalyse

Bei der Risikoanalyse geht es darum die wichtigsten Risiken zu identifizieren und zu bewerten. Dazu sind Standardformulare zu verwenden, in denen die finanzielle Auswirkung und die Wahrscheinlichkeit eines jeden Risikos eingeschätzt werden. Hier werden auch die möglichen Gegenmaßnahmen erfasst, Maßnahmen welche die Risikoauswirkung eindämmen können. Aus ihnen geht der Risikoreduktionsfaktor hervor. Die Risiken werden in diverse Risikoklassen aufgeteilt – z.B. organisatorische Risiken, personale Risiken, technische Risiken und Geschäftsrisiken. Am Ende wird ein Risikofaktor als

Risiken = (Auswirkungsgrad * Wahrscheinlichkeit) * (1 – Risikoreduzierungsfaktor)

Das Ergebnis – die Risiken – fließt in die Wirtschaftlichkeitsrechnung hinein.

2.2 Kostenschätzung

Die Kostenschätzung einer Migration setzt voraus, dass das Migrationsobjekt, in diesem Falle der Code, vermessen wird. Die Vermessung zielt darauf hin, bestimmte quantitative Größen aus dem Code zu gewinnen, die in Personenaufwand umgesetzt werden können. Solche Größen sind u.a. Codezeilen, Anweisungen, Data-Points, Function-Points und Object-Points. Diese Points werden durch eine automatisierte Analyse des Source-Codes gewonnen, die allenfalls mehrere Stunden dauern sollte. Die gleichen Größen werden auf gleicher Weise aus Systemen abgeleitet, die schon migriert worden – die Benchmark-Systeme. Zu diesen Benchmark-Projekten werden hoffentlich Aufwandsdaten vorliegen. Durch die Gegenüberstellung der Aufwände mit den Systemgrößen wird die Produktivität in diesen Projekten ermittelt. Über die Teilung der Größen des neu zu migrierenden Systems durch die mittlere Produktivität der bereits migrierten Systeme kommt man zum rohen Aufwand.

Der rohe Aufwand wird durch einen Einflussfaktor und möglicherweise durch einen Komplexitäts- und Qualitätsfaktor justiert. Diese Faktoren spiegeln die Differenz zwischen den Benchmark-Systemen und dem Zielsystem wider.

Aufwand = (Systemgröße / Produktivität) * Einflussfaktor * Komplexität * Qualität

Das Ergebnis ist der Personalaufwand, der sich in Personalkosten umsetzen lässt. Die Personalkosten sind durch Overheadkosten, Softwarekosten und Hardwarekosten zu ergänzen. Die ergänzten Kosten geben die Gesamtkosten der Migration.

2.3 Nutzenanalyse

Die Nutzenanalyse einer Migration setzt voraus, dass man quantitative Daten über interne und externe Qualitätseigenschaften hat. Zu den internen Eigenschaften gehören Maßzahlen die aus dem Code ableitbar sind, Eigenschaften wie Wiederverwendbarkeit, Portierbarkeit, Testbarkeit und Wartbarkeit. Hinzu kommen die eigentlichen Wartungsaufwände, Fehlerraten, System Performanzwerte, Operationskosten und Benutzerzufriedenheitsgrade. Diese sind die Istwerte. Die Sollwerte sind die gleichen Maßzahlen multipliziert mit dem jeweiligen Verbesserungsfaktor. Ein Verbesserungsfaktor von 1,2 bedeutet, dass wir uns einen 20% Verbesserung durch die Migration versprechen. Der Gesamtverbesserungsfaktor ist die Summe aller einzelnen Verbesserungsfaktoren. Er wird mit dem gegenwärtigen Nutzen der Zielapplikation multipliziert um den neuen Nutzwert zu errechnen.

Neuer Nutzwert = alter Nutzwert * Gesamtverbesserungsfaktor

Es kann auch zu einem neuen Nutzwert kommen der niedriger als der alte Nutzwert ist, nämlich dann wenn einzelne Verbesserungsfaktoren < 1 ausfallen. Wenn der Verbesserungsfaktor für Wartbarkeit auf 1,2 und der Verbesserungsfaktor für Performanz auf 0,8 geschätzt werden, gleichen sie sich aus. Der

Gesamtverbesserungsfaktor wird 1, d.h. keine Verbesserung. Da einzelne Eigenschaften wie Zuverlässigkeit wichtiger sein können als Andere muss es eine Möglichkeit geben die Eigenschaften zu gewichten. Der Gesamtverbesserungsfaktor ist dann die Summe der gewichteten Verbesserungsfaktoren.

Der Nutzen des Migrationsprojektes ist der neue Nutzwert – der alte Nutzwert. Dies ist der Nutzen der in die Wirtschaftlichkeitsrechnung einfließt.

3 Tutorium Ablauf

Das Tutorium ist eine Mischung aus Vorträgen, Übungen und Demonstrationen. Es beginnt mit einer Einführung in die Migrationswirtschaftlichkeitsanalyse. Danach folgt eine kurze Lektüre über Risikoanalyse, eine Demonstration der Risikoanalyse und eine Übung in Risikoanalyse eines kleinen Migrationsprojektes. Auf die Risikoanalyse folgt die Kostenschätzung mit Vortrag, Demonstration und Übung. Die Teilnehmer bekommen eine Codegröße, einen Komplexitätsmaß, einen Qualitätsmaß und einen Produktivitätsmaß um den Migrationsaufwand zu schätzen. Danach folgen ein Vortrag, eine Demonstration und eine Übung in Migrationsnutzenanalyse. Zum Schluss wird am PC-Arbeitsplatz eine Tool-gestützte Wirtschaftlichkeitsanalyse für ein Musterprojekt vorgeführt.

4 Abbildungen

Die Teilnehmer des Tutoriums erhalten zwei MS-Windows Werkzeuge die sie mitnehmen können:

- SofAudit für die Prüfung und Messung von Legacy Systeme in Assembler, C, COBOL, PL/I, Natural und VisualAge.
- SoftCalc für die Schätzung der Migrationsaufwände nach Anweisungen (COCOMO), Data-Points, Function-Points und Object-Points

Für die Anwendung dieser Tools bekommen die Teilnehmer auch zwei Fallbeispiele aus der IT-Praxis die sie nachvollziehen können.

5 Tutorium Ergebnisse

Nach dem Besuch dieses Tutoriums sind die Teilnehmer in der Lage ihre Legacy Systeme zu messen und die Ablösemöglichkeiten nicht nur zu schätzen sondern auch im Hinblick auf ihren Nutzen zu bewerten. Es geht darum dass sie für jede Situation sie den geeigneten Ausweg finden. Alte Systeme lassen sich neuentwickeln, neu-

implementieren, konvertieren und kapseln. Es komme darauf an die richtige Lösung zu finden. Dies wird in diesem Seminar vermittelt.

Literaturverzeichnis

[BG99] Bisbal, J.; Grimson, J; Lawless, D.; Wu, B.: Legacy Information Systems: Issues and Directions. IEEE Software, Vol. 16, No. 5, S. 103-111, 1999.

[Bo99] Boehm, B. W. u.a.: Software Cost Estimation with COCOMO-II, Prentice-Hall, Upper Saddle River, N.J., 1999

[DF08] De Lucia, A./Francese, R.: “Developing System Migration Methods and Tools for Technology Transfer”, in Journal of Systems and Software, Vol. 70, No. 1, 2008, s. 3

[Di89] Dietrich, W.: “Saving a Legacy System with Objects”, Proc. of OOPSLA-88, ACM Press, New York, 1989, p. 5

[Eb04] Ebert, J.: Software-Reengineering - Umgang mit Software-Altlasten. In: Informatiktage 2003. Konradin-Verlag, Grasbrunn, S. 24-31, 2004.

[Me07] Meyer, D. : "Modernisierung von Legacy Systemen - Risiken und Kosten senken durch Automation", Objektspektrum, Nr. 5, Sept. 2007, S.33.

[SP03] Seacord, R./Plakosh, D./Lewis, G. (2003): Software Modernization, Addison-Wesley, Boston, 2003

[Sn84] Sneed, H.M.: “Software Renewal – A case study”, IEEE Software, Vol. 1, No. 3, July, 1984, s. 56

[Sn91] Sneed, H.M: Softwaresanierung, Rudolf Müller Verlg Köln, 1991

[Sn91] Sneed, H.M.: "Economics of Software Reengineering", Journal of Software Maintenance, Vol. 3, No. 3, Sept. 1991, s. 163

[Sn95] Sneed, H.M.: Planning the Reengineering of Legacy Systems. IEEE Software, Vol. 12, No. 1, S. 24-34, Jan. 1995, S. 24.

[Sn96] Sneed, H.M.: “Encapsulating Legacy Software for Reuse in Client Server Systems” Proc. Of 3rd WCRE, IEEE Computer Society Press, Monterey, CA., Nov., 1996, p. 104

[Sn97] Sneed, H.M.: „Metriken für die Wiederverwendbarkeit von Softwaresystemen“, in Informatikspektrum, Vol. 6, S. 18-20, 1997.

[Sn99] Sneed, H. M.: “Risks involved in Reengineering Projects” Proc. of WCRE-1999, IEEE Computer Society Press, Atlanta, Oct. 1999, p. 204-211

[Sn99] Sneed, H.M.: Objektorientierte Softwaremigration. Addison-Wesley, Bonn, 1999

[Sn01] Sneed, H.M.: “Wrapping Legacy COBOL programs behind an XML Interface”, Proc. of Working Conference on Reverse Eng., IEEE Computer Society Press, Stuttgart, Oct. 2001, p. 189.

[Sn05a] Sneed, H.M.: Software-Projektkalkulation - Praxiserprobte Methoden der Aufwandsschätzung für verschiedene Projektarten. Hanser, München/Wien, 2005, S. 159.

[Sn05b] Sneed, H.M.: "Estimating the Costs of a Reengineering Project", IEEE Proc. of 12th WCRE, Computer Society Press, Nov. 2005, Pittsburgh, PA., s. 111

[SW10] Sneed, H., Wolf, E., Heilmann, H.: “Software Migration in der Praxis”, dpunkt.verlag, Heidelberg, 2010.

[TB02] Tortorella, M./ Bodhum, T./ Albanese, C.: “ A Toolkit for Applying a Migration Strategy”, IEEE Proc. of 6th CSMR, Computer Society Press, Budapest, March 2002, S. 154.

[Vo05] Von Hahn (2005): Werterhaltung von Software – Planung und Bewertung von Reengineering Maßnahmen am Beispiel von Standard Software, Deutscher Universitätsverlag, Wiesbaden, 2005, S. 173

[Wa99] Warren, I.: The Renaissance of Legacy Systems: Method Support for Software-System Evolution. Springer, London, 1999.