Schätzung Reengineering Alternative

Harry M. Sneed
SoRing Kft. H-1221 Budapest
ZT-Prentner, A1220 Wien
Harry.Sneed@T-Online.de

Abstrakt: Bei der Auswahl einer geeigneter Migrationsstrategie müssen u.a. die Kosten berücksichtigt werden. Das macht es erforderlich den Aufwand für jedes Alternativ zu schätzen und zu vergleichen. In dem Projekt, das hier geschildert wird gab es drei Alternative, den alten Code direkt 1:1 in Java konvertieren, den alten Code erst zu sanieren und dann in Java transformieren, und den Code komplett neu zu schreiben in Java. Dieser Beitrag beschreibt worum es ging, wie vorgegangen. Da das Migrationsprojekt noch aussteht, ist es zu früh zu sagen welchen Migrationspfad eingeschlagen wird, geschweige denn ob der geschätzte Aufwand auch stimmt. Es geht hier allein um die Schätzmethode. .

Schlüsselwörter: Migration, Reengineering, Software Aufwandsschätzung, Productivity, Risikoanalyse, COCOMO, Data-Point, Function-Point.

1 Hintergrund des Migrationsprojektes

Eine mittelständische Speditionsfirma in Österreich steht vor der Notwendigkeit ihr bestehendes AS-400 bzw. I-Series, System für die Steuerung von Transportgütern durch Europa abzulösen. Das alte System ist in der 4GL Sprache Synon von Computer Associates implementiert aus der RPG Code generiert wird. Synon gehört zu den CASE Tools der 80er Jahren und ist eigentlich sehr effektiv. Synon Entwickler können bis zu 100% produktiver als Java Entwickler sein weil sehr viele Business-Funktionen in der Sprache eingebaut sind. Synon baut auf einer Repository auf und verbindet viele Software Entitätentypen mit einander – Bildschirmmasken, Berichte, Schnittstellen, Dateien und Business-Logik Module mit einander. Sie ist auf die AS-400 Maschine zugeschnitten.
Nur Synon hat Grenzen was das Mengengerüst anbetrifft und das Zielsystem ist an diese Grenzen gestoßen. Außerdem hat die Sprache auch Grenzen was die Integrierbarkeit und Portierbarkeit anbetrifft. Schließlich ist sie eine Exote in der heutigen IT-Welt. Es gibt kaum noch Entwickler die sie beherrschen. In Anbetracht dieser Defizite wurde entschieden sie abzulösen.

2 Migrationsalternative

Aufgrund einer Voruntersuchung wurden drei alternative Migrationsstrategien festgestellt – 1:1 Konversion, Reengineering und Neuentwicklung. Für Konversion und Reenginerering wurden zwei Pilotprojekte mit Ausschnitten aus dem Gesamtsystem durchgeführt um a.) die Machbarkeit und b.) den Aufwand zu ermitteln. Aus dieser Erfahrung wurden die Produktivitätsdaten entnommen. Die Produktivität für die Neuentwicklung in Java wurde aus einem Schwesterprojekt entnommen. Die wichtigsten Codegrößen – Anweisungen, Data-Points, Ojbect-Points und Function-Points wurden mittels einer statischen Analyse aus dem Java Source-Code abgeleitet und justiert nach Komplexität und Qualität. Diese Kenngrößen wurden dann den Aufwandsdaten gegenübergestellt um die mittlere Produktivität von 1,1 Function-Points pro Personentag festzustellen.

3 Zerlegung des Zielsystem Source-Codes

Ehe mit der Analyse des Zielsystems begonnen wurde, musste es in 13 Einzelsysteme zerlegt werden. Der Anwender wollte die Möglichkeit haben die Migration schrittweise durchzuführen, also ein Teilsystem nach dem Anderen. Eine betriebswirtschaftliche Analyse ergab dass es im Gesamtsystem 13 einzelne Systeme gab, die funktional voneinander unabhängig waren. Dennoch gab es eine Menge Dateien und Code-Module die in mehreren dieser Teilsysteme vorkamen. Um die Systeme unabhängig voneinander zu migrieren wurden diese Elemente vervielfältigt. In jedes Teilsystem kam eine Kopie.
Das hatte zur Folge, dass auch die Source-Bibliotheken reorganisiert werden mussten. Die eine große Source-Bibliothek wurde in 13 einzelne Bibliotheken aufgeteilt und die Duplikate in jene Bibliotheken hineinkopiert zu denen sie gehörten. Dadurch wurde die gesamte Codemenge erhöht aber jedes Teilsystem war vollständig.und konnte als eigenes Projekt geschätzt werden.

4 Messung der Zielsystemgröße

Die Größe des Synon Systeme musste indirekt gemessen werden, da es kein Tool gibt die den Synon Source analysieren kann. Im Grunde genommen gibt es gar keine Synon Source, sondern nur eine Repository mit verschiedenen Artifaktentypen. Deshalb wurde der generierte RPG Source für die Messung herangezogen und durch einen Expansionsfaktor von 3:1 in Java Größen umgesetzt, d.h. aus den 7,8 Millionen RPG Statements wurden 2,6 Java statements abgeleitet. Function-Points und Data-Points sind unabhängig von der Programmiersprache. Es folgten aus der Analyse

130.954 Data-Points und 32.383 Function-Points. Diese Größen wurden mit den Aufwandsdaten aus der Zielsystementwicklung verglichen um eine Produktivität von 1,9 Function-Points pro Personentag zu errechnen. Diese Produktivitätsrate wurde verifiziert gegen die gebuchten Aufwände anderer Synon Projekte.

5 Ermittlung der Produktivitätsdaten

Die Ermittlung der Produktivitätsdaten ist eine der Haupthindernisse zur Schätzung von Aufwand in der Industrie. Die wenigsten Anwenderbetriebe haben sie. Das zwingt die Aufwandschätzer auf Daten aus der Literatur auszuweichen, z.B. die Capers Jones Daten für Function-Point Produktivität [Jone00]. Diese Praxis darf jedoch nur eine Notlösung sein. Die Produktivität variiert so erheblich zwischen Ländern und Anwendern, dass sie kaum geeignet ist. Bereits in einer Studie aus dem Jahr 2000 hat Kathrin Maxwell erwiesen dass es große Unterschiede gibt zwischen Branchen. Banken und Versicherungen sind 100% weniger produktiv als Handelsunternehmen [Maxw2000]. Bei derartig großer Unterschiede ist auf die Produktivitätsdaten aus der Ferne wenig Verlass.
Zum Glück konnten die Schätzungen hier auf heimische Produktivitätsdaten aufbauen. Die Produktivität bei einer Neuentwicklung wurde aus dem anderen Java Projekt übernommen. Die Produktivität bei einer 1:1 Konversion wurde aus dem Konversionspilotprojekt übernommen. Sie betrug in etwa das Doppelte der Produktivität in einer Neuentwicklung, d.h. 2,2 Function-Points pro Personentag. Die Produktivität bei dem Reengineering Projekt betrug 60% der Produktivität in einer Neuentwicklung, d.h. 1,75 Function-Points pro Personentag.

6 Aufbau einer Schätzdatenbank

Jetzt konnte mit der eigentlichen Kostenkalkulation begonnen werden. Die Größen, Komplexitäten und Qualitäten aus der Codemessung wurden aus der Codemessungsdatenbank in die Schätzdatenbanken übernommen. Da jedes Teilsystem als eigenständiges Projekt zu betrachten war gab es eine eigene Schätzdatenbank für jedes Teilsystem. Die Größenkennzahlen wurden auf der Modulebene geliefert, d.h. es gab eine Zeile in der Schätztabelle für jeden Modul.

```
< Entity = "Component"
 EName = "BORKOR3"
 EType = "MODU" >
  <Component_Attributes>
    <Nr_Locs>001640</Nr_Locs>
    <Nr_Stmts>000829</Nr_Stmts>
    <Nr_Functs>000042</Nr_Functs>
    <Nr_Variables>000254</Nr_Variables>
    <Change_Rate>0.000</Change_Rate>
  </Component_Attributes>
</Entity>
```

Man muss bedenken, dass es 5603 Module gab in dem gesamten System. Hinzu kamen die 2426 Bildschirmmasken, die 147 Druckberichte, die 1541 Physical Files und die 115 SQL Datenbanktabellen. Sie und ihre Attributanzahl wurden alle in die Schätzdatenbanken geladen.
Anschließend wurden sie aggregiert und durch die Komplexität und Qualität des einzelnen Systems justiert.
Zu den reinen Produktdaten kommen in der Schätzdatenbank die Risikodaten, die Produktivitätsdaten und die Einflussfaktoren hinzu. Die Risikodaten wurden aus der Risikoanalyse übernommen. Die Produktivitätsdaten kamen von den Benchmark-Projekten. Lediglich die Einflussfaktoren mussten vom Schätzer eingegeben werden. Damit war die Schätzdatenbank für die Schätzung selbst aufbereitet.

7 Schätzung der Migrationskosten

Nachdem die Schätzdaten aufbereitet waren liefen die einzelnen Schätzungen automatisch ab, eine für jede der drei Migrationsvarianten für jedes der 13 Teilsysteme plus das System als Ganzes. Das machte 42 Schätzungen insgesamt. Jede Schätzung wurde nach drei verschiedenen Methoden durchgeführt: COCOMO, Data-Point und Function-Point. Sie konnten alle innerhalb eines Tages durchgezogen werden. Falls etwas geändert werden musste, konnte die Schätzung leicht wiederholt werden.
In dem Schätzbericht gibt es neben dem Aufwand auch die Kosten, die erforderliche Personenanzahl und die geschätzte Dauer. Für das gesamte System gab es folgende Schätzungen für 1115 KDSI, 130594 Data-Points und 32380 Function-Points.:

- Konversion nach COCOMO = 788 PMs
- Konversion nach DataPt = 772 PMs
- Konversion nach FunctPt = 846 PMs
- Reengineering nach COCOMO = 1011 PMs
- Reengineering nach DataPt = 985 PMs
- Reengineering nach FunctPt = 1111 PMs
- Redevelopment nach COCOMO = 1673 PMs
- Redevelopment nach DataPt = 1694 PMs
- Redevelopment nach FunctPt = 1835 PMs

Eine Untermenge dieser Schätzungen gab es für jedes Teilsystem aufgrund der Größen des Subsystems.
Mit diesen Daten wurde der Anwender in die Lage versetzt die alternativen Migrationsansätze nach ihrer Kosten zu vergleichen. Es folgte eine weitere Untersuchung sie nach ihren Nutzen zu vergleichen. Der Nutzen einer Neuentwicklung ist natürlich viel größer als der einer reinen Konversion. Ob er den Kosten ausgleicht ist eine andere Frage.

Literaturhinweise:

[Maxw00] Maxwell, K./Forseluus, P.: Survey of Software Productivity in Europe, IEEE Software, Jan. 2000, S. 80

[Jone00] Jones, C.: Software Assessments, Benchmarks and best Practices, Addison-Wesley, Reading, MA., 2000