Werkzeugunterstützung für iterative Modernisierungsprozesse

Matthias Vianden
Research Group Software Construction
RWTH-Aachen

Frank Berretz
SOPTIM AG

Tobias Rötschke
SOPTIM AG

Zusammenfassung

In dieser Arbeit stellen wir unsere Überlegungen und Erfahrungen bei der Werkzeugunterstützung für iterative Modernisierungsprozesse von formularbasierten Informationssystemen vor. Nachdem wir die Anforderungen für ein Werkzeug skizzieren, stellen wir deren beispielhafte Implementierung im Werkzeug SOPTIM CQM vor. Der zweite Teil beschäftigt sich mit unseren Erfahrungen bei der Einführung des Werkzeugs. Neben den Problemen und deren Lösungen erläutern wir, die Veränderungen durch die Einführung.

1. Einführung

Die SOPTIM AG entwickelte sich vom Anbieter projektbezogener Individualsoftware hin zum Anbieter für Softwareprodukte. Bei den Produkten handelt es sich um Informationssysteme, welche zum Teil aus den Projektentwicklungen stammen. Die Domäne der Produkte ist die Energielogistik – der Vertrieb und Einkauf von Energie sowie die Verwaltung von Energiedaten. Durch die Anforderungen der Projektentwicklung wurden die Anwendungen initial mit dem Rapid Prototyping Werkzeug Delphi entwickelt. Die Produkte bilden mittlerweile eine Produktfamilie mit einem gemeinsamen Kern, wobei der Übergang langsam innerhalb von mehreren Jahren erfolgte. Die verwendeten Sprache und die damit verbundenen Prozesse wurden daher weitgehend beibehalten. Die Entwicklerteams vergrößerten sich mit steigender Vielfallt der Kundenanfragen, begleitet von ständig ändernden rechtlichen Rahmenbedingungen.

Wie in [LL07] beschrieben, *alterten* die Produkte und der gemeinsame Kern – die Wartung und Wiederverwendung wurde immer aufwändiger. Im Sommer 2007 beschloss daher die Soptim AG, einen Prozess zu entwickeln, um gezielt diese Aspekte zu verbessern.

2. Entwurf des Modernisierungsprozesses

Der Prozess sollte sowohl die Architektur der Produkte verbessern, als auch kleine lokale Verbesserungen (Refactorings) erlauben. Um die Verbesserungsmaßnahmen zu steuern, evaluieren und einzuläuten sollten Messungen sowie Regelverletzungen - *BadSmells* - genutzt werden. Der Prozess besteht im Wesentlichen aus den vier Phasen: Planung, Durchführung, Evaluation und Iteration. Eine detaillierte Beschreibung des Prozesses, des Vorgehens in den einzelnen Phasen, sowie zwei beispielhafte Anwendungen des Prozesses finden sich in [Via08].

3. Werkzeugunterstützung

Zur Durchführung des beschriebenen Prozesses sind ständige Messungen unerlässlich. Dabei stehen statische Quelltextanalysen im Vordergrund, da primär eine Verbesserung des Quelltextes angestrebt wird. Die Anwendungen bei der SOPTIM AG werden hauptsächlich in Delphi und PL/SQL entwickelt. Für diese spezielle Konstellation kann nicht auf vorhandene Standardwerkzeuge wie Sotograph [Bis04] oder XRadar [Kva05] zurückgegriffen werden.

Es musste daher ein eigenes, auf die vorhandene Situation optimiertes Werkzeug entwickelt werden. Die folgenden Punkte wurden von uns bei der Entwicklung des Werkzeugs als besonders wichtig empfunden und bauen auf den Erfahrungen von [Roe06] auf.

- Das Werkzeug muss aus drei Komponenten auf gebaut sein, der Datenerfassung, Datenhaltung und der Datenaufbereitung, um die Analyse zeitlich zu entkoppeln.
- Die Analyse muss nicht nur ein breites Spektrum von Metriken messen, sondern auch einen Regelkatalog überprüfen und strukturelle Abhängigkeiten aufschlüsseln, damit eine möglichst breite Grundlage für die Modernisierungsentscheidungen besteht.
- Die Visualisierung muss einen schnellen Zugriff auf die Analysedaten und eine intuitive Navigation ermöglichen, da das System sonst von den Anwendern nicht akzeptiert wird.

4. Das Werkzeug SOPTIM CQM

Die genannten Anforderungen wurden von uns im Werkzeug SOPTIM CQM realisiert. Die Architektur des Werkzeugs besteht dabei im Wesentlichen aus drei Modulen: der in Perl realisierten Datenanalyse, der Datenspeicherung in eine MySQL Datenbank und der Visualisierung in Form einer PHP basierten Webapplikation. Um einen schnellen und dynamischen Zugriff auf die Anwendung zu gewährleisten, wird zum einen die Visualisierung durch AJAX Technologien unterstützt und zum anderen die Analyseergebnisse bereits in optimierter Form in der Datenbank vorgehalten. So wird beispielsweise zu jedem Artefakt ein fertig formatierter Quelltext hinterlegt. Neben klassischen Visualisierungen, wie Listen, Tabellen und

Balkendiagrammen, ist es in SOPTIM CQM außerdem möglich, beliebige Messungen als Treemap anzeigen zu lassen – diese stellen die Messwerte in der Hierarchie der Artefakte dar. Die Kombination von Messwerten und Strukturanalysen ist insbesondere für Architekten und Teamleiter interessant, bietet aber auch dem einzelnen Entwickler die Möglichkeit ein zeitnahes Feedback über die von ihm entwickelten Module zu erhalten.

5. Erfahrungen

Im folgenden Abschnitt erläutern wir die Hauptprobleme, die bei der Einführung des Werkzeugs aufgetreten sind, sowie unsere Lösungsansätze.

Das erste Problem, dem wir begegnen mussten war, die Akzeptanz des Werkzeugs im Unternehmen zu stärken. Hierbei war es wichtig insbesondere den Entwicklern ein Verständnis für die Notwendigkeit von Messungen zu vermitteln, ohne Angst vor Überwachung zu schüren. Durch den modularen Aufbau des Werkzeugs konnten wir nach der Einführung schnell auf Nachfragen bezüglich neuer Messungen, weiterer Regeln oder Layout-Anpassungen reagieren. Den Mitarbeitern wurde dadurch deutlich, dass ihre Wünsche und Anforderungen an das Werkzeug ernst genommen werden.

Die Integration der Möglichkeiten des Werkzeugs in vorhandene Wartungsprozesse war ebenfalls nicht selbstverständlich. Neben den positiven Ergebnissen der beispielhaften Durchführung des Modernisierungsprozesses [Via08] bildeten hier insbesondere die Teamleiter die führende Kraft. Diese akzeptierten aus oben genannten Gründen das Werkzeug sehr schnell und förderten dadurch dessen Benutzung auch in Wartungsprozessen.

Ein weiteres Problem war das Verständnis der Mitarbeiter bezüglich der im Werkzeug verwendeten Metriken zu stärken. Als erste Maßnahme, um diesem Problem zu begegnen, wurden gezielt Schulungen und Workshops angeboten – zusätzlich wurde dadurch auch Refactoring Knowhow vermittelt. Um das Verständnis der Metriken zu erhöhen, implementierten wir neben Standardmetriken wie LOC, NCSS, CycComp und WMpC auch selbst entwickelte Metriken. Eine dieser Metriken misst beispielsweise die für eine Methode notwendigen Testfälle für gegebene Überdeckungskriterien, um deren Komplexität greifbar zu machen. Andere messen Eigenschaften der Delphi-Formulare, um gezielt Fragestellungen an das Benutzerinterface beantworten zu können.

6. Fazit

In der täglichen Arbeit manifestiert sich die Einführung des Werkzeugs hauptsächlich darin, dass das Bewusstsein für problematischen Quelltext und die damit häufig einhergehenden Wartungsproblemen gestärkt wurde. Den Teamleitern und dem Management wurde bewusst, dass ein gezieltes Vorgehen an dieser Stelle sinnvoll ist und entsprechend Zeit für die Behebung der Probleme eingeräumt werden muss – bei der Definition neuer Arbeitspakete wird dieses schon berücksichtigt.

Die Architektur der Produkte und die Interaktion der einzelnen Komponenten wurden durch die Einführung des Werkzeuges transparenter. Durch die sprachübergreifenden Abhängigkeitsanalysen können die Entwickler nun nahtlos in einem Werkzeug von Delphi- zu PL/SQL-Artefakten wechseln und umgekehrt.

Insbesondere durch den letzten Punkt ist das Arbeiten mit dem Werkzeug nun seit knapp einem Jahr fest etabliert. Entwickler nutzen es insbesondere zur Abhängigkeitsanalyse und sprachübergreifenden Suche; Teamleiter nutzen es, um gezielt Modernisierungsmaßnahmen zu initiieren. Auch das Beheben von, durch das Werkzeug gemeldeten, Regelverstößen gehört mittlerweile zur täglichen Arbeit. Die Teamleiter und das Architekturteam nutzt daher Regeln gezielt, um spezielle Defizite im Quelltext zu beheben.

Um eine endgültige Aussage über den Erfolg des Werkzeuges und des Modernisierungsprozesses zu ermöglichen, müssen allerdings noch mehr Daten erhoben werden. Insbesondere für eine Evaluierung des Prozesses ist die Implementierung von Trendanalysen und Vergleichen im Werkzeug unerlässlich.

7. Literatur

[Bis04] Bischofberger W., et. al., Sotograph - A Pragmatic Approach to Source Code Architecture Conformance Checking. In: Proceedings of the First European Workshop on Software Architecture (EWSA 2004), 2004

[Kva05] Kvam, K. et. al., Legacy system exorcism by Pareto's principle. In: Companion to the 20th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2005), 2005

[LL07] Ludewig, J.; Lichter, H.: Software Engineering. dpunkt.verlag, 2007

[Roe06] Rötschke, T.: Metamodellbasierte Generierung von kundenspezifischen Software-Leitständen. In: INFORMATIK 2006 - Informatik für Menschen - Band2. Bonn: Gesellschaft für Informatik, 2006

[Via08] Vianden, M.: Entwurf und Realisierung eines Ansatzes zur Modernisierung der Architektur eines Formularbasierten Informationssystems. RWTH-Aachen, Diplomarbeit, 2008