Kein UML ohne Patterns

Max Kleiner

armasuisse
CH-3003 Bern
max.kleiner@armasuisse.ch
max@kleiner.com

Abstract:

Die in vielen Software Projekten enthaltenen Muster (Analyse-, Entwurfs- und Architekturmuster, engl. Patterns), lassen sich konkret in UML modellieren und darstellen. Muster oder Patterns gehören zweifellos zu den nützlichsten und erfolgreichsten Konzepten des Software Engineering.

Anhand einer Zuordnung von Analyse Mustern zu Use Cases und Activity, der Zuordnung von Design Pattern zu Klassen-, Zustand- und Sequenzdiagramm und der Zuordnung von Architektur Pattern zu Paket- und Komponenten Diagramm lassen sich die Pattern in UML notieren. Jeweils einen Vertreter der offiziellen Muster stelle ich im Referat vor. Die UML wird so mit Patterns erweitert.

Damit Patterns in einem definierten Sinne zur Verfügung stehen, braucht es einen Katalog oder ein Archiv der in UML visualisiert wird. Deshalb gilt auch: Keine Patterns ohne UML.

"A Pattern is a proven Solution for a general Problem"

1 Zusammenhang von UML und Patterns aufzeigen

Existierende Designtechniken beschreiben Probleme, die während der Analyse und des Designs auftreten können und bieten dann passende Lösungen an. Sie vernachlässigen jedoch den Bereich der Erfahrung in der Entwicklung, d.h. wie kommt man auf eine Lösung wo noch kein Muster vorliegt. Die durch UML gefundene Lösung lässt sich auch in ein mögliches Standard Pattern einbringen und in einem der nächsten Projekte einsetzen. Aus Patterns entsteht UML und aus UML wiederum entstehen Patterns. Folgende Vorteile lassen sich in die Analyse und Design Phase einbringen:

- Patterns beinhalten Erfahrung von Experten und Praktikern.
- Patterns können (sollten) in UML notiert sein.

- Patterns beschreiben, warum eine bestimmte Alternative, d.h. ein anderes Pattern, verwendet wird, und die sich daraus ergebenden Konsequenzen. Daher lassen sich Entscheidungen später leichter nachvollziehen und erleichtern die Dokumentation in UML.

Patterns bringen flexible und wiederverwendbare Strukturen aus der Designphase in die Analysephase zurück. Dies ist ein interessanter Punkt, denn dadurch läßt sich der iterative Übergang vom Use Case zum Klassendiagramm vereinfachen.

2 UML mit Patterns in Analyse, Design und Implementierung erweitern

Martin Fowler [MF00] zeigt in seinem Buch über *Analysemuster* einen Katalog von Mustern, die bestimmte Anwendungsbereiche untersuchen. Diese Bereiche lassen sich als Kanditaten für Anwendungsfälle (Use Case) und darauf folgende Aktivitäts-diagramme einsetzen.

Entwurfsmuster von Erich Gamma et al. [DP01] ist ein bereits älteres (seit 1994), aber oft empfohlenes Handbuch für die 20 wichtigsten Problemlösungsstrategien in der objektorientierten Softwareentwicklung. Sie lassen sich vor allem in Klassen- und Sequenzdiagrammen darstellen.

Frank Buschmann und seine Kollegen [BU96] teilen in ihrem Buch Patterns nach ihrem Abstraktionsgrad ein. Er beginnt mit der höchsten Abstraktionsstufe und nennt diese Architektur Patterns. Diese beschreiben und erklären die fundamentale Struktur von Softwaresystemen auch aus der Sicht der Implementierung. Sie geben die nötigen Subsysteme an und definieren Funktionen und Beziehungen zueinander. Auch wenn man anschließend kein fertiges Architekturkonzept für die eigene Anwendung hat, bringt es mit Paket- und Komponentendiagrammen neue Lösungsansätze näher.

2.1 Dokumentation und Testbarkeit

Die in UML eingesetzten Muster sind als bekannte Standards in der Notation schon mehrfach diskutiert und in der Praxis ausgiebig getestet worden. Zudem lassen sich aus den Mustern Codestrukturen generieren, die als Referenz auch schon geprüft sind.

Dokumentation, insbesondere die exakte und definierte Spezifikation von Schnittstellen (Interfaces), die lokale Verständlichkeit von Anweisungen resp. von Kommentar im Code und der Parameter sind das Hauptziel einer Wiederverwendbarkeit von Mustern, die man in der UML einsetzt.

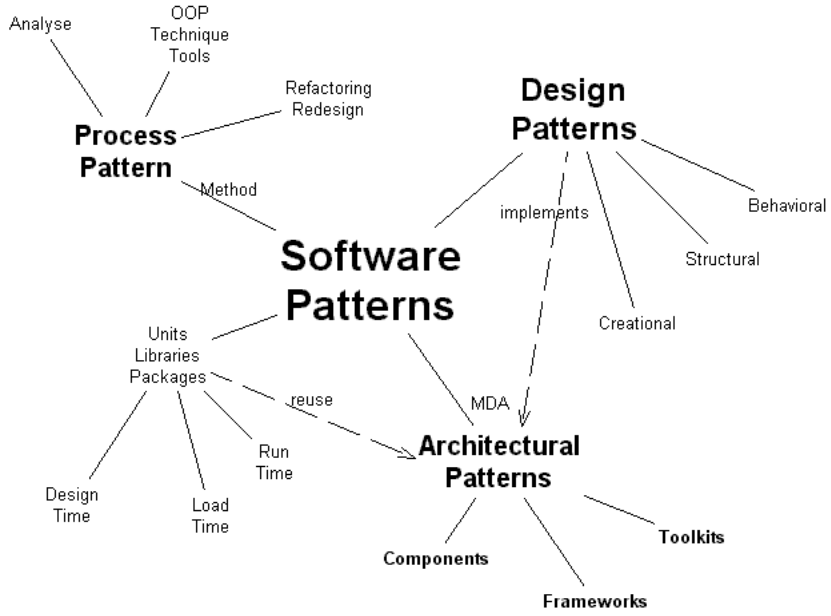

Abbildung 1: Aufzeigen von Patterns in UML

2.2 UML und Simplifikation

Wir stehen noch vor der Frage, wie Prozesse der Simplifikation zu erklären sind. Und welche technischen Strukturen oder Pakete sind in besonderer Weise "anfällig" für Vereinfachungen (Standards)? Um einer Beantwortung dieser Fragen näher zu kommen, sind die Technischen Anforderungen der Schlüssel. Die angewandte Simplifikation, also eine Verdichtung und Vereinfachung der IT-Lösungen, stehen seit Jahren im Interesse vieler Firmen, doch fehlt dann der Mut die Prozesse mit Hilfe von Patterns auch wirklich zu vereinfachen. Ein Lösungsansatz lässt sich mit Patterns in UML zeigen!

Anforderungen, Architektur- oder Designfragen lassen sich vielfach mit Standards wie Patterns, Profile oder Referenzen einfacher lösen, sofern man vom "not invented here" Syndrom wegkommt.

Literaturverzeichnis

[MF00] Martin Fowler: Analysis Patterns: Reusable Object Models, Addison-Wesley,
[DP01] Erich Gamma et al., Entwurfsmuster, Addison-Wesley, München
[BU96] Buschmann et al., Pattern-orientierte Software-Architektur, Addison-Wesley
[MK03] Kleiner et al.: Patterns konkret, 2003, Software & Support Verlag, ISBN 3-935042-46-9
[MK04] Pattern Poster: http://www.softwareschule.ch/download/patternposter.pdf