Boris Wehrle
(E-Mail: Boris.Wehrle@aitgmbh.de)
ist Senior Consultant bei der AIT AG. Er berät Unternehmen bei der Softwareentwicklung auf Basis von Microsoft Technologien. Seine Schwerpunkte liegen in der Konzeption von verteilten Anwendungen sowie der Unterstützung bei der Umsetzung von Entwicklungsprozessen auf Basis des Visual Studio mit dem Team Foundation Server.

Gelebte Architekturdokumentation in einem globalen Team

Ein neuer Mitarbeiter steht in der Tür und schon stellt sich die Frage: „Wie integrieren Sie diesen am schnellsten in Ihr Team und bringen ihn auf den aktuellen Projektstand?“ Die Architekturdokumentation wäre ein guter Einstieg. Welche Architekturdokumentation? Und was ist das überhaupt?

Projektübergreifende und projektspezifische Architekturdokumentation

Man kann zwischen projektübergreifenden und projektspezifischen Architekturdokumentationen [MAD] unterscheiden. Eine projektübergreifende Architekturdokumentation trifft allgemeine Festlegungen bezüglich der zu verwendenden Architektur- und Designpatterns sowie Kodierrichtlinien. Ihr Ziel ist die Vereinheitlichung der im Unternehmen zu entwickelnden Software. Ein Wechsel von einzelnen Entwicklern zwischen verschiedenen Teams in einem Unternehmen wird hierdurch vereinfacht. Die Qualität der Anwendung sowie deren Wartbarkeit wird verbessert.

Eine projektspezifische Architekturdokumentation gibt einen Überblick über die Komponenten eines Systems. Sie stellt Zusammenhänge und Abhängigkeiten zwischen diesen dar. Weiterhin werden die Schnittstellen und das Verhalten der Komponenten beschrieben. Häufig werden zu den Komponenten Ansprechpartner definiert, die bei konkreten Fragen konsultiert werden können. Damit bietet diese für neue Teammitglieder einen sehr guten Überblick über das zu entwickelnde System. Bei Anforderungsänderungen kann die Architekturdokumentation zur Analyse von Auswirkungen herangezogen werden.

Beide Dokumentationsformen bilden die Basis für eine erfolgreiche Entwicklung eines Systems. Durch die fortlaufende Verbesserung können auch alle nachfolgenden Applikationen hiervon profitieren. Die Abhängigkeit vom Wissen Einzelner wird reduziert.

Die Notwendigkeit einer projektübergreifenden und projektspezifischen Architekturdokumentation steigt insbesondere in folgenden Umgebungen:

- häufig wechselnde Teamzusammenstellungen
- Vielzahl gleichartiger Projekte, die von unterschiedlichen Teams bearbeitet werden
- häufige Anforderungsänderungen
- hohe Fluktuation der Mitarbeiter
- Entwicklung an verteilten Standorten

Insbesondere die letzten beiden Punkte stellen eine spezielle Herausforderung in globalen Teams dar. Hier kommen zusätzlich Kommunikationsprobleme durch unterschiedliche Sprachen, einen anderen Kulturkreis sowie Zeitverschiebungen hinzu.

Architekturdokumentationen findet man in der Praxis in sehr unterschiedlicher Ausprägung. Sehr häufig existieren diese überhaupt nicht. In anderen Unternehmen umfassen sie wiederum mehrere hundert Seiten, in denen man sich schnell verliert. In wieder anderen sind die Dokumentationen mehrere Jahre alt. Die beschriebenen Systeme haben sich in der Zwischenzeit in eine ganz andere Richtung entwickelt. Damit verfehlen die Architekturdokumente oft ihre Funktion.

Projektübergreifende Architekturdokumentation

Um dies zu verhindern, kann man sich für eine projektübergreifende Architekturdokumentation an einigen Grundprinzipien orientieren. Dabei muss stets der Nutzen hinterfragt werden. Für wen dokumentiere ich? Welchen Nutzen zieht der Lesende aus der Dokumentation? Ein Dokument, das häufig gelesen wird, wird auch gelebt!

Reduzierung auf das Wesentliche

Der Umfang des Architekturdokuments sollte auf das Notwendige minimiert werden. Dies erhöht die Akzeptanz und reduziert den Pflegeaufwand. Allgemein-

gültige Patterns die anzuwenden sind, sollten z. B. nur genannt und kurz angerissen werden. Auf weiterführende Informationen wird mithilfe von Links verwiesen. Codebeispiele werden direkt aus einer bestehenden Applikation referenziert. Hier kann bei Bedarf nachgeschlagen werden. Veraltete oder redundante Informationen werden aus dem Dokument gelöscht.

Zielorientiertes Schreiben zur besseren Verständlichkeit

Die wesentliche Zielgruppe der Architekturdokumentation sind Entwickler. Dazu gehören sowohl die aktuellen als auch später hinzukommende Teammitglieder. Nur diese müssen das Dokument verstehen. Auf die Aufführung von Grundlagenwissen kann daher meist verzichtet werden. Skizzen oder UML-Modelle geben in der Regel einen schnelleren und besseren Überblick als lange Texte. Dies reduziert zusätzlich den Pflegeaufwand. Eine immer mehr zunehmende Tendenz zu globalen Teams empfiehlt die Dokumentation in englischer Sprache.

Navigierbarkeit

Eine Architekturdokumentation sollte so aufgebaut sein, dass diese nicht linear gelesen werden muss. Eine Übersicht zusätzlich zum Inhaltsverzeichnis hilft, direkt zu den aktuell relevanten Informationen zu navigieren. Innerhalb des Dokuments kann zwischen den einzelnen Bereichen verlinkt werden.

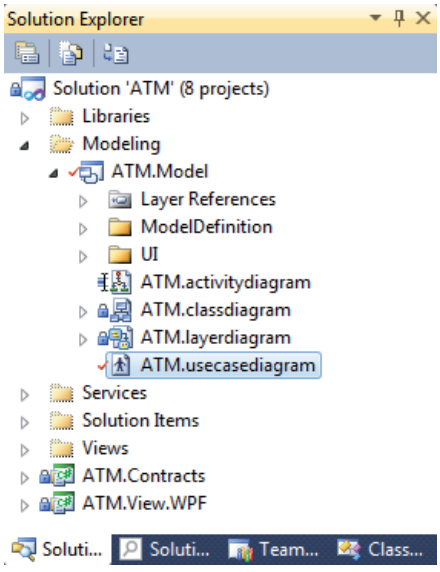

Abb. 1: Modelle werden mithilfe eines Modellierungsprojektes direkt in die Projektmappe integriert.

Aktualität

Die Architekturdokumentation ist nur so lange wertvoll, wie diese sich auf dem aktuellen Stand befindet. Eine Aktualisierung sollte deshalb zeitnah erfolgen. Darüber müssen alle Entwickler informiert werden. Eine veraltete Dokumentation verliert schnell an Akzeptanz und wird nicht weitergepflegt. Die Erstellung der Dokumentation kostet Zeit und diese muss im Projektplan reserviert werden. Leider fällt dieser Task sehr oft den Streichungen bei Termindruck zum Opfer. Dabei sollte man sich stets den Wert für die Wartung der Applikation sowie für alle zukünftigen Weiterentwicklungen wieder ins Gedächtnis rufen.

Versionierung / Zugriff

Architekturdokumentationen unterliegen Veränderungen. Sie bilden mit den entwickelten Applikationen eine Einheit. Eine Ablage in der Quellcodeverwaltung parallel zum Code ermöglicht eine Versionierung und einen einfachen Zugriff durch die Zielgruppe und erinnert nebenbei an die regelmäßige Pflege.

Format / Werkzeuge

Für das Erstellen der Dokumentation sollten Werkzeuge verwendet werden, die jedem Entwickler zur Verfügung stehen. Sehr häufig kommen MS Word oder Visio zum Einsatz. Auch eine Verwendung eines Wikis ist denkbar. Bei Berücksichtigung dieser Prinzipien entsteht ein für das Unternehmen wertvolles Dokument.

Für die projektspezifische Dokumentation eignet sich Word nur sehr bedingt. Ein von der Entwicklungsumgebung losgelöstes Dokument bleibt stets ein Fremdkörper. Zusätzliche, wertvolle Funktionen wie eine automatische Prüfung der Architektur stehen damit nicht zur Verfügung.

Projektspezifische Architekturdokumentation

Das Visual Studio 2010 stellt für die Architekturmodellierung und -dokumentation neue Werkzeuge zur Verfügung. In erster Linie sind hier die UML-Diagramme zu nennen. Diese werden, wie Abbildung 1 zeigt, mithilfe eines Modellierungsprojektes direkt in die Projektmappe integriert. Damit stehen sie jedem Entwickler zur Verfügung und werden automatisch im Rahmen der Anwendung versioniert.

Ein Auseinanderlaufen der Dokumentation und des Quellcodes wird hierdurch wirkungsvoll verhindert. Bei Verwendung von Namenskonventionen, wie z. B. „Pro-

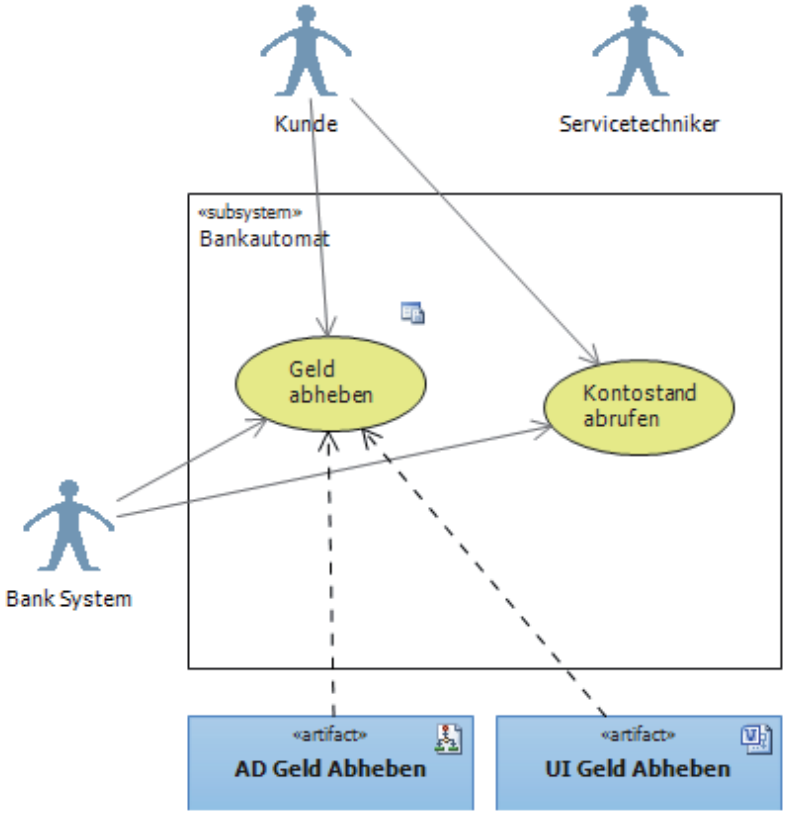

Abb. 2: Ein Use Case Diagramm gibt einen Überblick und verlinkt zu weiter detaillierten Dokumenten.

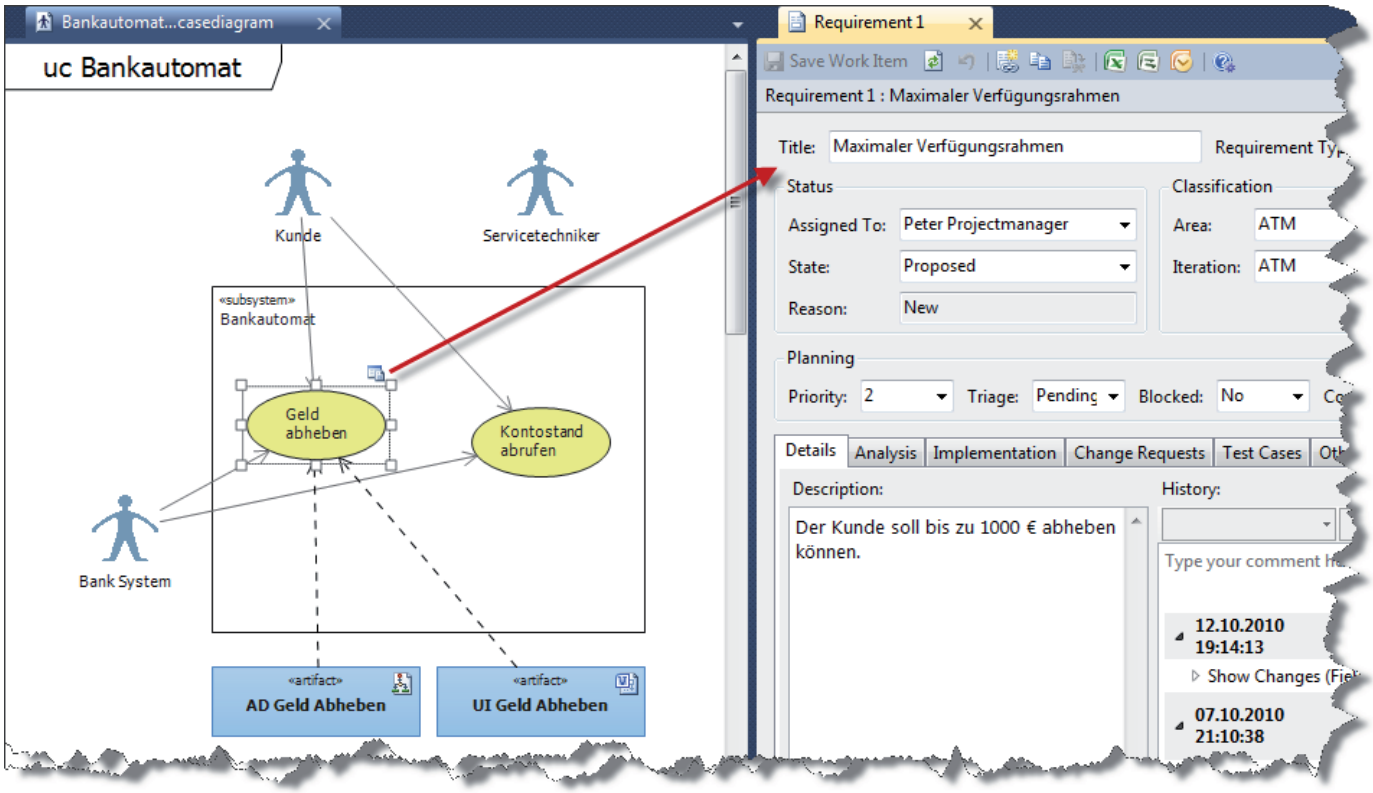

Abb. 3: Zu Use Cases können innerhalb des Team Foundation Servers Anforderungen erfasst werden.

jektmappenname + Model“ in allen Projekten, ist ein Auffinden, auch in großen Projektmappen, schnell möglich.

Ein Use Case Diagramm bietet sich, wie in Abbildung 2 gezeigt, als Übersicht und zugleich Einstieg an. In diesem können die wichtigsten Szenarien zusammengefasst werden.

Zu diesen können, wie in Abbildung 3 dargestellt, konkrete Anforderungen durch die Integration des Team Foundation Servers erfasst werden.

Zudem ist eine Navigation in beide Richtungen gegeben. Auch ein Zugriff auf das Modell aus einer Anforderung (Requirement) heraus ist möglich. Auswirkungen von Änderungsanforderungen können somit schnell analysiert werden.

Zur Strukturierung der Anwendung bietet sich ein Schichtendiagramm, wie Abbildung 4 zeigt, an. Abhängigkeiten zwischen Komponenten werden verdeutlicht.

Durch eine Zuordnung von Visual Studio Projekten zu Komponenten ist zusätzlich eine Validierung der Beziehungen untereinander möglich. „Abkürzungen“ werden schnell erkannt und können damit umgehend behoben werden. Das Diagramm bietet sich ebenfalls als Diskussionsgrundlage bei der Betrachtung von Systemerweiterung an.

Abbildung 5 zeigt, wie Entitäten in einem Klassendiagramm in Beziehung gesetzt werden können. Zusammenhänge können so einfach visualisiert und damit im Schichtendiagramm grob umris-

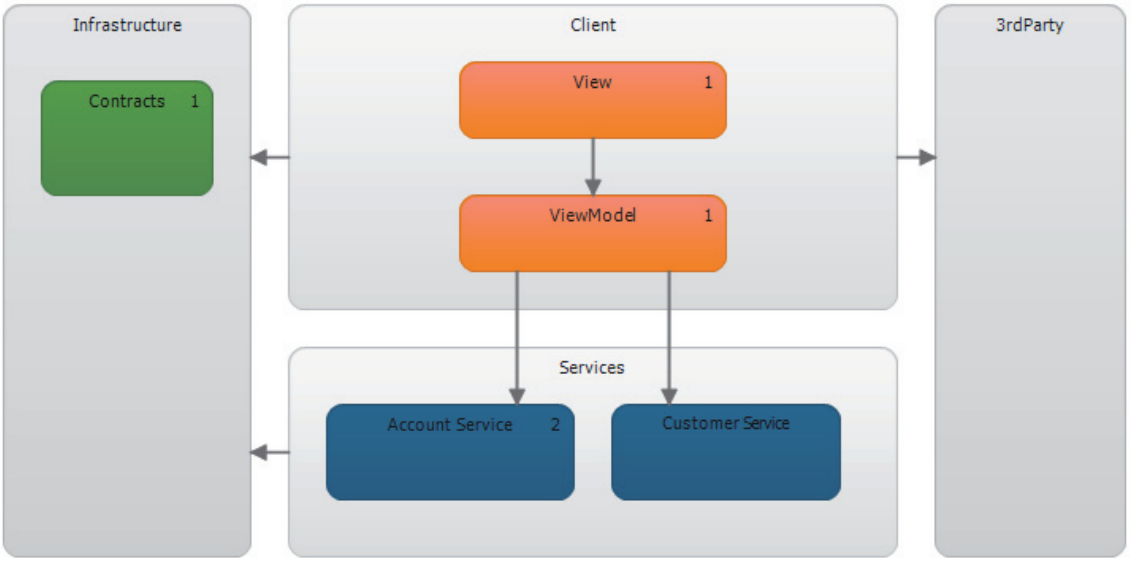

Abb. 4: Modellierung von Service- und schichtenorientierten Architekturen.

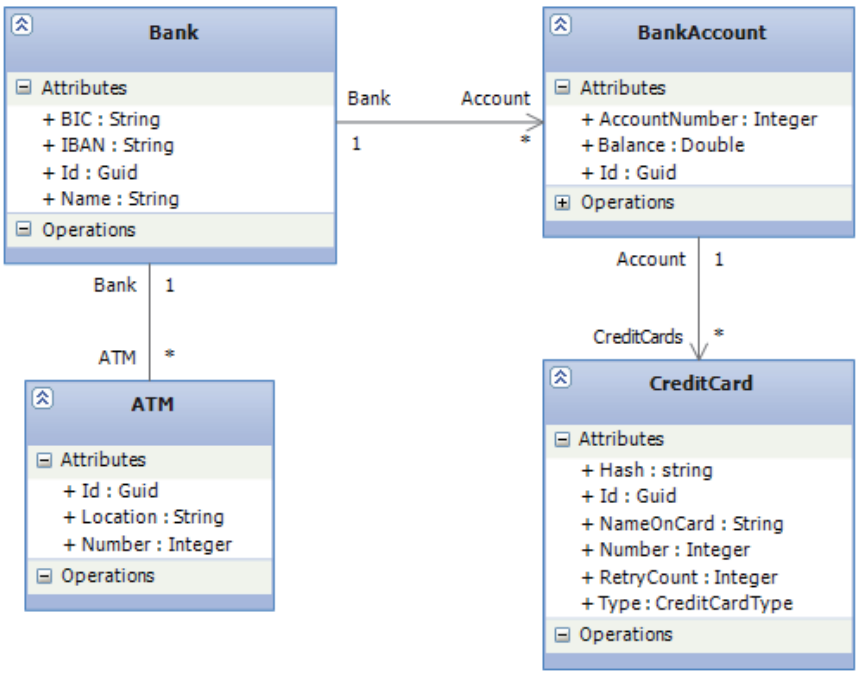

Abb. 5: Modellierung von Entitäten in Klassendiagrammen.

sene Komponenten weiter verfeinert werden.

Zusätzlich ist es möglich, aus dem Modell flexibel mithilfe des Text Transformation Toolkits Quellcode (siehe Abbildung 6) zu generieren. Hierdurch gewinnt das Modell eine neue Bedeutung. Zusätzlich zur Analyse- und Dokumentationsfunktion wird eine Basis für den Quellcode zur Verfügung gestellt. Eine modellgetriebene Entwicklung wird hierdurch unterstützt und gefördert.

Für die Dokumentation stehen weitere Diagrammtypen, wie z. B. das Komponentendiagramm, das Sequenzdiagramm und das Aktivitätsdiagramm, zur Verfügung. Detaillierte Schnittstellenbeschreibungen werden im Quellcode hinterlegt. Aus diesem kann bei Bedarf ein Dokument generiert werden. Ein Befüllen einer Dokumentenschablone, wie sie zum Beispiel von arc42 [arc] zur Verfügung gestellt wird, ist ebenfalls denkbar. Es erfordert jedoch die Implementierung entsprechender Werkzeuge die auf dieses Modell aufsetzen.

```
AIT.DataCont...tTemplate.tt
GeneratedTextTransformation
TransformText()
24  namespace <#= TransformationContext.DefaultNamespace #>
25  {
26
27      using System;
28      using System.Collections.Generic;
29      using System.Runtime.Serialization;
30      using System.ComponentModel;
31
32      /// <summary>
33      /// <#= this.ModelClass.Description #>
34      /// </summary>
35      [DataContract]
36      public partial class <#= this.ModelClass.Name #> : INotifyPropertyChanged
37      {
38          DataMembers
53
54          INotifyPropertyChanged Members
78      }
79  }
80  <#+
81          return this.GenerationEnvironment.ToString();
82      }
```

Abb. 6: Code Generierung mithilfe des Text Transformation Toolkits (T4).

Fazit

Die verschiedenen Diagrammtypen sowie zusätzliche Analysefunktionen im Visual Studio 2010 vereinfachen eine projektspezifische Architekturdokumentation. Weitere Elemente, wie Anforderungen und User Interface Skizzen, werden an den relevanten Stellen verlinkt. Dadurch entsteht ein vielseitig einsetzbares, vernetztes, virtuelles Dokument. Dabei erfolgt die Dokumentation nicht im Nachhinein, sondern wird aktiv zur Unterstützung oder sogar als Basis der Entwicklung genutzt. So bekommt eine vermeintlich zusätzliche Belastung – als welche die Dokumentation oft gesehen wird – eine produktivitätssteigernde Funktion, die durch den intensiven Gebrauch von standardisierten Modellen auch in globalen Teams verstanden wird. Die Architekturdokumentation wird gelebt! ■

Referenzen

[MAD] MAD. [Online] Wikipedia. http://de.wikipedia.org/wiki/Meta-Architektur-Dokument.

[arc] *arc42*. *arc42* Template. [Online] arc52. http://www.arc42.de/template/template.html.