Sommerloch

Markus Völter, voelter@acm.org, www.voelter.de

Diese Kolumne adressiert praktisch relevante („Der Praktiker") technologisch interessante Themen. Dies bedingt, dass die interessanten Themen auch tatsächlich in der Praxis eingesetzt wurden. Nun habe ich diesen Sommer aber glücklicherweise auch Urlaub gehabt und deshalb nichts interessantes praktisches getan (abgesehen von Segelfliegen in den Schweizer Alpen [MV04], wofür sich im Rahmen dieser Kolumne aber wahrscheinlich niemand interessiert). Ich werde daher in diesem Artikel etwas tun, was ich schon lange einmal vorhatte: Auf interessante Bücher hinweisen, die ich in der letzten Zeit gelesen habe.

Einleitung

Bücherlesen gehört bei uns Technologiegetriebenen ja zu den essentiellen Tätigkeiten. Mir persönlich geht es aber immer öfter so, dass ich nicht so genau weiß, was ich lesen soll; viele Bücher sind so spezifisch für eine Technologie/API/Version, dass sie schon veraltet sind, bevor man sie zu Ende gelesen hat. Andere wiederum sind so allgemeine und wenig konkret, dass ich es oft nicht schaffe, bis zum Ende durchzuhalten. Die Idealform eines Buches, also praktisch relevant mit der richtigen Prise Grundlagen kommt recht selten vor. Ich werde in diesem Artikel also versuchen, auf einige solche Bücher hinzuweisen. Natürlich ist so eine Unternehmung hochgradig subjektiv…

Grundlagen, Architektur und Verteile Systeme

Eine ganze Serie von Büchern möchte ich jedem ans Herz legen der sich mit der Architektur und der Implementierung verteilter Systeme beschäftigt, sowie generell etwas über Softwarearchitektur lernen möchte: Die *Pattern Series* von Wiley. In Summe findet sich in diesen Büchern eine unheimlich mächtige Wissensbasis zu diesem Thema, handhabbar in Patterns aufgeteilt. Die darin bisher zu diesem Thema erschienenen Bücher (*Pattern Oriented Software Architecture* Vol 1 und 2, *Server Component Patterns*) möchte ich hier jetzt nicht noch mal näher erläutern; vielmehr möchte ich auf die beiden kürzlich erschienenen Mitglieder der Serie hinweisen. Zunächst sei da *POSA 3, Patterns for Resource Management* von Michael Kircher und Prashant Jain [KJ04]. Dieses Buch geht auf Strategien zur Resourceverwaltung ein, Beispiele wären Dinge wie Lazy Acquisition, Caching, Pooling, Leasing oder Eviction. Verdeutlich wird das ganze durch zwei recht ausführliche Case Studies. Das andere kürzlich erschienene Buch, *Architecting Enterprise Solutions* von Paul Dyson und Andy Longshaw [DL04] beschäftigt sich mit der Architektur performanter und skalierbarer Internet-/Business Systeme. Das Buch enthält eigentlich alles was der Architekt wissen muss: Lastverteilung, Failover, Persistenz, DMZs, Status Reporting und Logging, Weiterentwicklung des Systems, etc. Sehr gut geschrieben, für e-Business-Architekten unbedingt lesenswert. Die Weiterentwicklung der Serie ist im Übrigen gesichert: Das nächste Buch wird sich Remoting Architekturen (CORBA, .NET, Webservices) widmen.

Ein anderes Buch zu dem Thema rund um verteilte Systeme sei auch noch erwähnt: *Distributed Systems, Principles and Paradigms* von Tanenbaum und Steen [TS02]. In bewährter Tanenbaum-Qualität widmet sich dieses Buch den Prinzipien verteilter Systeme, wobei Aspekte wie Verteilung, Synchronisation, Naming, Replikation, Fehlertoleranz und Security angesprochen werden. Verschiedene Paradigmen werden dann erläutert, unter anderem objektbasierte verteilte Systeme, Message-basierte Systeme oder verteilte Dateisysteme. Illustriert wird all dies mittels bekannter Technologien wie CORBA, Sun NFS oder Lotus Notes. Als Grundlagenbuch unbedingte Pflichtlektüre.

Und wo wir grade schon bei Grundlagen sind: Neben Verteilungsinfrastrukturen gehören Betriebssysteme und Programmiersprachen zu den für uns wichtigen Grundlagen. Zum Thema Betriebssysteme sei das Buch *Modern Operating Systems* von Tanenbaum [AT01] empfohlen, und zum Thema Programmiersprachen das Buch *Concepts of Programming Languages* von Robert W. Sebesta [RS99]. Es geht auf die wichtigsten Konzepte von Programmiersprachen ein und erläutert diese anhand verschiedenener Sprachen. Insbesondere werden auch funktionale und logische Programmiersprachen adressiert – und es schadet ja nie, etwas über den typischen OO/prozeduralen Tellerrand hinaus zu schauen.

Ein weiteres Buch welches ich jedem der die OO Grundlagen und die GoF-Patterns verstanden hat ans Herz legen möchte ist Eric Evans' *Domain-Driven Design* [EE04]. Es dreht sich primär darum, wie man (UML-) Diagramme und Quellcode ausdrucksstärker macht, die Artefakte also besser mit den Konzepten der Domäne in Einklang bringt, die sie beschreiben sollen. Unter anderem adressiert es , wie man richtig über die Domäne redet (konsistente Verwendung von Sprache) und beschreibt ein generisches Domänenmetametamodell in Form von Entities, Services, Value Objects, Factories und Repositories. Das Refactoring von Code, sodass er die Semantik besser zum Ausdruck bringt nimmt des weiteren großen Raum ein. Auch die Partitionierung eines Systems in verschiedene Teile und deren Koordiniertes Zusammenspiel wird erläutert. Zu beachten ist dabei, dass dieses Buch keine bestimmte Technologie (abgesehen von der Objektorientierung) beschreibt und auch nicht auf modellgetriebene Codegenerierung eingeht (was man bei dem Titel vermuten könnte). Es beschreibt lediglich, wie man in einem Team wartbare, ausdrucksstarke und performante Objektorientierte Systeme baut – dieses Wissen ist essentiell war aber vorher noch nirgends so gut und konsistent beschrieben.

Embedded und C/C++

Practical Statecharts in C/C++ von Miro Samek ist ein wirklich hervorragendes Buch. Es geht um die Verwendung von Statecharts für die Implementierung von Embedded Systemen. Zunächst bringt Samek eine der besten Einführungen in (den wirklich relevanten Teil von) Statecharts. Er beschreibt dann verschiedene Implementierungsalternativen in C und C++. Eine dieser Implementierungsalternativen – der aus seiner Erfahrung beste Kompromiss aus Performance und Wartbarkeit – beschreibt er im Detail; Samek hat für diese Art der Implementierung ein Framework entwickelt, welches auf der beiliegenden CD zu finden

ist. Im 2. Teil des Buches erläutert er die Funktionsweise des Frameworks und dessen Implementierung auf verschiedenen Plattformen (RTKernel32, DOS, Win32). Wenn man dieses Buch liest, bekommt man unweigerlich den Eindruck, dass da ein Praktiker seiner Erfahrungen niedergeschrieben hat und man bekommt direkt Lust, mit diesem Framework Embedded-Systeme zu entwickeln. Es ist sehr gut lesbar geschrieben (auch wenn glaube ich mehr Klammer-Einschübe und Fußnoten in dem Buch zu finden sind, wie in allen anderen Büchern in meinem Regal in Summe).

Ich möchte noch auf drei weitere Bücher rund um die C/C++-Embedded Welt eingehen. Zum einen Schmidt und Huston's *C++ Network Programming Vol. 1* und *Vol. 2*. Die Bücher sind zwar recht anstrengend zu lesen (insbesondere sollte man POSA 2 vorher gelesen haben), aber sie zeigen sehr schön wie man in C/C++ portable, skalierbare Frameworks für verteilte Anwendungen bauen kann, illustriert am Beispiel des bekannten ACE Frameworks. *Volume 1, Mastering Complexity with ACE and Patterns* [SH02] zeigt die grundlegenden Aspekte wie Multithreading, Connection Management und Plattformunabhängigkeit. *Volume 2, Systematic Reuse with ACE and Frameworks* [SH03] zeigt die ACE Implementierungen der POSA 2 - Patterns Reactor, Proactor, Acceptor-Connector und Service Configurator. Interessant an den beiden Büchern ist nicht nur der eigentliche Inhalt, sondern auch die Art und Weise wie C und C++ verwendet werden – sehr lehrreich.

Als letztes Buch im Abschnitt C/C++ und Embedded sei noch das Buch *Real-Time Systems and Programming Languages* von Burns und Wellings [BW97] erwähnt. Es erläutert die wichtigsten Konzepte und Grundlagen von Echtzeitsystemen und zeigt, wie Echtzeitderivate bekannter Programmiersprachen funktionieren.

Generative Programmierung und AOP

AspectJ in Action von Ramnivas Laddad [RL03] ist – wie der Name schon sagt – ein Buch über AspectJ. Auch wenn es spezifisch ist für AspectJ ist es ein sehr gutes Buch zum Thema AOP allgemein, und mit Abstand das Beste zum Thema AspectJ. Das Buch zeichnet sich vor allem durch nicht-triviale, aber trotzdem verständliche Beispiele aus. Dabei werden verschiedene relevante Java Technologien berücksichtigt, unter anderem erläutert ein Beispiel die Verwendung von JAAS und AspectJ zum Bau einer Security-Lösung. Andere Beispiele adressieren die Implementierung von effizientem Resourcenmanagement (Pooling, Caching) mittels AspectJ, oder das Thema Threadsicherheit. Abgerundet wird das ganze durch eine Reihe von Best Practices und Patterns. Ein anderes Buch aus demselben Dunstkreis ist *The Art von the Metaobject Protocol* von Kiczales, des Rivieres und Bobrow [KRB91]. Metaobjektprotokolle stellen historisch gesehen ja die Basis für Aspektorientierung dar (wie man an dem Namen Kiczales auf der Autorenliste unschwer erkennen kann). Das Buch enthält zwar jede Menge CLOS Code, ist aber gut geschrieben und extrem lesenswert auch und vor allem um Aspektorientierung zu verstehen. Um die Kategorie „Abstruse Syntax" vollends abzuschließen, sei noch das Buch *Modern C++ Design* von Andrei Alexandrescu [AA01] erwähnt. Es adressiert templatebasierte generische Programmierung in C++, auch unter Templatemetaprogrammierung bekannt.

Auch wenn das Thema für die allermeisten sicherlich harte Kost ist, ist das Buch sehr schön geschrieben und absolut lesenswert. Insbesondere die statische (weil templatebasierte) Umsetzung einiger der bekannten GoF-Design Patterns ist faszinierend.

Ein anderes Buch welches ich jedem empfehle *Generartive Programming* von Eisenecker und Czarnecki [EC00]. Entgegen dem oft gehörten Vorurteil, das Buch wäre voller komischer C++ Template Metaprogramme (nur ein oder zwei Kapitel enthalten diese wirklich) enthält es eine ganze Menge nützlicher Themen rund um generative und modellgetriebene Entwicklung, darunter Domänenanalyse, Variabilitätsanalyse und Featuremodellierung.

Last but not Least möchte ich in diesem Abschnitt noch das Buch *Executable UML* von Mellor und Balcer [MB02] erwähnen. Es beschreibt, wie man mittels UML präzise, vollständige, und damit ausführbare (in dem Sinne, dass man sie automatisiert in etwas ausführbares transformieren kann) Modelle beschreibt. Unter anderem kommt dabei sehr viel OCL und Action Semantics zum Einsatz. Als Nebeneffekt beinhaltet das Buch auch eine sehr pragmatische, auf das Wesentliche fokussierte Einführung in die UML.

Prozesse, Soft Skills

Für alle die die nicht gar so technologiegetrieben sind sondern eher die klassische Beraterrolle einnehmen seinen vier herausragende Bücher erwähnt. Zum einen sei da der Klassiker von Alistair Cockburn: *Agile Software Development* [AC02]. Cockburn beschreibt in diesem Buch die Prinzipien agiler Softwareentwicklung, und er tut das in einem sehr gut lesbaren Stil. Ich hoffe, Sie haben das Buch schon gelesen… wenn nicht, nachholen!

Wenn es darum geht, Agile Softwareentwicklung in größeren Projekten anzuwenden, kommt man mit den im Großteil der Literatur geschilderten Praktiken nicht unbedingt weiter; ganz anders Jutta Eckstein's *Agile Software Development in the Large* [JE04]. Es skaliert agile Softwareentwicklung für große Projekte und kommt dabei wirklich aus der Projektpraxis (ich kann's beurteilen, ich war teilweise dabei ☺).

Ein anderes Buch welches sich sehr amüsant liest, aber einen sehr ernsten Hintergrund hat ist *Secrets of Consulting* von Jerry Weinberg [GW85]. Er beschreibt das Zusammenspiel zwischen Beratern und deren Kunden. Aus Sicht des Beraters ist es zum einen ein Spiegel für sich selbst, und eine „Bedienungsanleitung" für den Kunde; aus Sicht des Kunden ist es eine „Handhabungsrichtlinie" für den Berater.

Als viertes Buch möchte ich *Project Retrospectives* von Norm Kerth [NK01] erwähnen. Projektretrospektiven ist eine Technik, Projektteams und den Projekterfolg im Nachhinein zu analyisieren um daraus für zukünftige Projekte zu lernen. Was sich trivial anhört ist in der Praxis höchst-kritisch, da sehr viel Psychologie und Befindlichkeiten im Spiel sind. Norm Kerth beschreibt diese Herausforderungen und bewährte Lösungen leicht verdaulich und angenehm zu lesen. Pflichtlektüre für jeden, der erwägt, das Mittel der Retrospektive einzusetzen.

Dokumentation und Typographie

Abschließen möchte ich mit zwei Büchern zu einem relativ ungeliebten Thema: Dokumentation. Zunächst sei das Buch *Agile Documentation* von Andreas Rüping [AR03] empfohlen. Es enthält eine Sammlung von Patterns (und jede Menge Beispiele!!) wie man leichtgewichtige und nützliche Dokumentation schreibt. Es beschreibt, wie man dokumentationsrelevante Themen findet, wie man einzelne Dokumente strukturiert, es sagt einiges zum Thema Layout und Typographie sowie zu den Themen Technische Aspekte und Qualitätssicherung. Aufgrund der Aufteilung in kleine Portionen (Patterns) und den vielen Beispielen liest sich das Buch sehr gut – auch wenn es ein aus Sicht vieler recht trockenes Thema behandelt.

Für diejenigen die den gestalterischen Aspekt von Dokumenten (oder Büchern) etwas genauer verstehen wollen, sei das Buch *Mut zur Typographie* von Gulbins und Kahrmann [GK00] empfohlen. Es behandelt die wichtigsten Konzepte und Begriffe rund um Schrift und Satz, erläutert Regeln des Tabellensatzes, die richtige Gestaltung und Platzierung von Abbildungen sowie den technischen Prozess wie ein Buch entsteht (sowie dessen Konsequenzen für den Autor). Meines Erachtens sollte jeder der etwas publiziert dieses Buch zumindest überfliegen, um die gröbsten Fehler zu vermeiden – die Leser werden's danken!

Sonstiges

Für Eclipse-Entwickler möchte ich das Buch *Contributing to Eclipse* von Gamma und Beck empfehlen. Es ist meines Erachtens die mit Abstand beste Einführung in die Innereien von Eclipse und *die* Grundlage für Leute, die Eclipse durch eigene Plugins selbst erweitern wollen. Es ist ausserdem extrem gut geschrieben, weil es oft Beck's testgetriebenen Ansatz verfolgt. Es werden also immer auch gleich Tests mit entwickelt, wenn ein Feature vorgestellt wird. Außerdem ist dieses Buch das einzige zu Eclipse, welches nicht die Hälfte des Umfanges damit verbringt, die Eclipse Java-IDE zu erläutern…

Das aus meiner Sicht beste Buch zum Thema Concurrency und Multithreading (vor allem, aber nicht nur für den Java Programmierer) ist Doug Lea's *Concurrent Programming in Java*. Es beschreibt die Herausforderungen von multithreaded Programmierung sehr anschaulich und beschreibt, wie man diese Probleme vernünftig in den Griff bekommt. Er erreicht dies mit einer Bibliothek von Hilfsklassen (die ja nun seit Java 1.5 Teil des JDK ist) deren Verwendung und Implementierung er zeigt. Für jeden der denkt, Multithreading sei in Java kein Problem ist dieses Buch Pflichtlektüre.

Abschließen möchte ich mit einem Buch welches für die meisten sicherlich jenseits des Tellerrands liegt, aber zumindest als Hintergrundwissen sehr nützlich ist: *Essentials of Artificial Intelligence* von Matt Ginsberg [MG93]. Er beschreibt Techniken zur Suche, zur Wissensrepräsentation (z.B. Prädikatenlogik) sowie typische Kategorien von AI Systemen (Planung, Lernen, Expertensysteme). Das Buch ist so geschrieben dass es auch von Lesern wie mir verstanden werden kann, die nicht 10 Semester theoretische Informatik studiert haben.

Referenzen

AA01	Andrei Alexandrescu. Modern C++ Design. Addison-Wesley 2001
AC02	Alistair Cockburn. Agile Software Development. Addison-Wesley 2002
AR03	Andreas Rüping. Agile Documentation. Wiley, 2003
AT01	Andrew S. Tanenbaum. Modern Operating Systems. Prentice Hall, 2001
BW97	Burns, Wellings. Real-Time Systems and Programming Languages. Addison-Wesley 1997
DL04	Dyson, Longshaw. Architecting Enterprise Solutions. Wiley 2004
DL99	Doug Lea. Concurrent Programming in Java. Addison-Wesley 1999
EC00	Eisenecker, Czarnecki. Generative Programming. Addison-Wesley 2000
EE04	Eric Evans. Domain-Driven Design. Addison-Wesley 2004
GB04	Gamma, Beck. Contributing to Eclipse. Addison-Wesley 2004
GK00	Gulbins, Kahrmann. Mut zur Typographie. Springer, 2000
GW85	Gerald M. Weinberg. Secrets of Consulting. Dorset House, 1985
JE04	Jutta Eckstein. Agile Software Development in the Large. Dorset House, 2004
KJ04	Kircher, Jain. Pattern Oriented Software Architecture, Vol. 3. Wiley, 2004
KRB91	Kiczales, des Rivieres, Bobrow. The Art von the Metaobject Protocol. MIT Press, 1991
MB02	Mellor, Balcer. Executable UML. Addison-Wesley 2002
MG03	Matt Ginsberg. Essentials of Artificial Intelligence. Morgan Kaufmann, 1993
MS04	Miro Samek. Practical Statecharts in C/C++. CMP Book, 2002
MV04	www.voelter.de/flying/sm04.html
NK01	Norman L. Kerth. Project Retrospectives. Dorset House,
RL03	Ramnivas Laddad. AspectJ in Action. Manning, 2003
RS99	Robert W. Sebesta. Concepts of Programming Languages. Addison-Wesley 1999
SH02	Schmidt, Hustom. C++ Network Programming, Vol. 1, Addison-Wesley 2002
SH03	Schmidt, Hustom. C++ Network Programming, Vol. 2, Addison-Wesley 2003
TS02	Tanenbaum, Steen. Distributed Systems. Prentice Hall 2002