Produktkernel in der Systemintegration (Erfahrungsbericht aus der Praxis)

Dr.-Ing. Ingo Elsen, Asma Hawari, Uwe Johnen

T-Systems GEI GmbH
Systems Integration Automotive and Manufacturing Industries
Pascalstraße 51
52076 Aachen
{ingo.elsen, asma.hawari, uwe.johnen}@t-systems.com

Abstract: In der Vergangenheit basierten große Systemintegrationsprojekte in der Regel auf Individualentwicklungen für einzelne Kunden. Getrieben durch Kostendruck steigt aber der Bedarf nach standardisierten Lösungen, die gleichzeitig die individuellen Anforderungen des jeweiligen Umfelds berücksichtigen. T-Systems GEI GmbH wird beiden Anforderungen mit Produktkerneln gerecht. Neben den technischen Aspekten der Kernelentwicklung spielen besonders organisatorische Aspekte eine Rolle, um Kernel effizient und qualitativ hochwertig zu entwickeln, ohne deren Funktionalitäten ins Uferlose wachsen zu lassen. Umgesetzt hat T-Systems dieses Konzept für Flughafeninformationssysteme. Damit kann dem wachsenden Bedarf der Flughafenbetreiber nach einer effizienten und kostengünstigen Softwarelösung zur Unterstützung Ihrer Geschäftsprozesse entsprochen werden.

1 Die Idee des Produktkernels

Große Systemintegrationsprojekte im Umfeld der Individual-Software liefen in der Vergangenheit oft so ab, dass, sofern der Typ der Anwendung bekannt war, Wiederverwendung in Form von Kopieren von Source Code betrieben wurde. Mit dem Aufkommen der Enterprise Java Plattform entstanden technische Frameworks, die zwar keinerlei fachliche Funktionalität bereitstellen konnten, den Entwicklern aber sehr viel Arbeit abnahmen und –nehmen. Ein Beispiel hierfür ist das Hibernate-Framework [HIB10]. Neben technischer Funktionalität besteht aber vielfach der Bedarf, über immer wiederkehrende fachliche Funktionalität verfügen zu können.. In der Systemintegration sind Produktlinien in der Regel zu unflexibel, da die Umgebungen, in die die Systeme integriert werden, für einen solchen Ansatz zu heterogen sind. Hier kommt die Idee des Produktkernels zum Tragen: Fachliche Grundfunktionalität, die allen Anwendungen gemeinsam ist, wiederverwendbar bereitzustellen.

In der Literatur [BAS06, BUS01, CLE07, FOW03] finden sich keine Einordnungen von Produktkerneln in die Taxonomie der Elemente einer Softwareanwendung. Wir haben daher einen Vorschlag hierzu entworfen, der sich in Abbildung 1 findet.

Beim Übergang von Bibliotheken zu Frameworks, Produktkerneln, Produktlinien und Produkten steigt jeweils die Komplexität des Elements, während die Flexibilität für den Einsatz in verschiedenen Anwendungsdomänen sinkt. So sind z.B. die Java-Standard-bibliotheken für praktisch jede Anwendungsdomäne einsetzbar; Java-Frameworks für Web-Anwendungen eigenen sich jedoch nicht für reine Client-Anwendungen.

Im Gegenzug sinkt mit steigender Komplexität der Elemente der Aufwand, um einen produktiven Einsatz zu erreichen. Während Off-the-Shelf-Produkte in der Regel sofort einsetzbar sind - maximal ist eine Einstellung der Konfigurationsdateien erforderlich - gilt es, aus Produktkerneln zunächst Anwendungen zu erstellen.

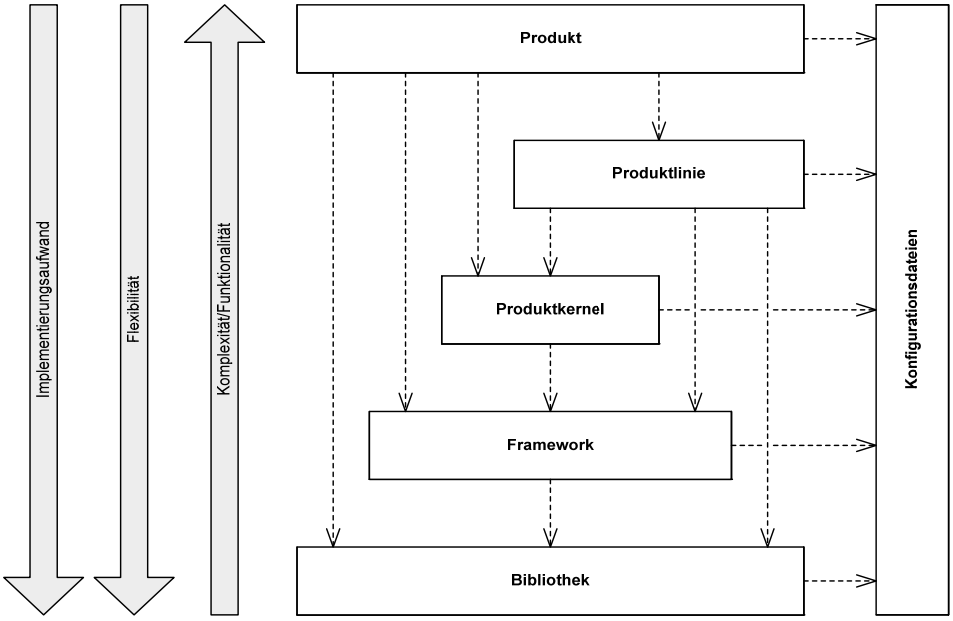

Abbildung 1: Einordnung der Produktkernels in die Taxonomie von Anwendungskomponenten

Produkte werden in ihrer Codebasis nicht von Installation zu Installation verändert. Variabilität in der Funktionalität erzielt der Administrator einer Anwendung durch Konfiguration von z.B. XML Dateien [PAR10]. Hier ist mittlerweile durch Deskriptive Sprachen (z.B. für GUI) eine Anwendungsadaption in weiten Grenzen möglich [QT10]. Hierbei sind die funktionalen Abläufe innerhalb der Anwendung vorgegeben und nicht veränderbar. Inhalte sind konfigurierbar, indem aus einer Gesamtmenge für eine konkrete Anwendung Teilmengen ausgewählt und dem Anwender zur Verfügung gestellt werden [SPR10].

Im Systemintegrationsgeschäft unterscheiden sich die Anforderungen von Kunde zu Kunde, basieren aber zumeist auf einer gemeinsamen Grundfunktionalität der Anwendungsdomäne. Die Abläufe können jedoch unterschiedlich sein, und die Inhalte sind nicht vollständig aus einer Obermenge ableitbar, da hier der Systemkontext, der in aller Regel auf gewachsenen individuell geprägten Anwendungslandschaften und –prozessen basiert, zu berücksichtigen ist.

Ein *Produktkernel* enthält bereits solche Grundfunktionalitäten wie z.B. eine Datenbank, in der sich ein Datenmodell findet, das die Gemeinsamkeiten der Geschäftsprozesse der Anwendungsdomäne enthält. Gleiches gilt für die Geschäftsfunktionalität in den Komponenten des Produktkernels. Auch hier sind komplette Workflows in ihrer Grundfunktionalität bereits implementiert und können durch externe Erweiterungen, z.B. via Shared Libs, unter Anwendung des Dependency Injection Musters [FOW04, NYG07] um kundenspezifische Funktionen ergänzt werden. Hier sind insbesondere Schnittstellen zu Bestandssystemen zu nennen.

Frameworks, die zwar bestimmte Prozessschritte modellieren können [BUS01], aber normalerweise keine Fachlichkeit eines Geschäftsprozesses beinhalten, sind wegen des zu leistenden Aufwands nur dann attraktiv, wenn eine einmalige Anwendungserstellung und Systemintegration gefordert ist. Der Aufwand zur Realisierung einer Fachlichkeit im Vergleich zu einem Produktkernel ist deutlich höher. Auf der anderen Seite eigenen sich Frameworks für die Anwendung in verschiedenen Fachgebieten wie z.B. bei unterschiedlichen Geschäftsprozessen.

In spezifischen Geschäftsdomänen gibt es viele fachliche Gemeinsamkeiten und wiederkehrende Prozesse. Daher ist es sinnvoll, über die Flexibilität von Frameworks hinaus Fachlichkeit so umzusetzen, dass sie in verschiedenen Anwendungsszenarien wiederverwendet werden kann. Gleichzeitig muss die Flexibilität bewahrt werden, so dass eine Anpassung an geänderte prozessuale Abläufe und fachliche Inhalte sowie eine Integration in unterschiedlichste Systemlandschaften gewährleistet ist.

Ein Produktkernel, wie ihn T-Systems einsetzt, besteht im Wesentlichen aus

- fachlichen Elementen, deren Inhalte aber in weiten Grenzen konfigurierbar sind.
- Konfigurationsschnittstellen zur Anpassung des Systemverhaltens in der Integrationslandschaft.
- Schnittstellen zu Frontends verschiedener Hersteller
- Schnittstellen zur Adaption der fachlichen Elemente und Workflows. Hier findet sich primär Dependency Injection (DI) als verbreitetes Architekturmuster.

In Tabelle 1 sind die Nutzen der verschiedenen Ansätze bezogen auf deren organisatorische Umsetzung und wirtschaftliche Anwendung dargestellt.

Kriterium	Bibliothek	Framework	Produktkernel	Produktlinie	Produkt
Realisierungsaufwand bis Inbetriebnahme[1]	--	-	+	+	++
Wartungsaufwand bei großer Basis installierter Anwendungen	--	-	+	+	++
Flexibilität in der Anwendung in verschiedenen Domänen	+	o	o	-	--
Flexibilität in der Anwendung innerhalb einer Domäne bei verschiedenen Szenarien	o	+	++	o	--

Tabelle 1: Eignung der verschiedenen Ansätze aus organisatorischer Sicht.

2 Flughafeninformationssysteme bei der T-Systems GEI GmbH

Die Luftfahrtindustrie hat sich in den vergangenen Jahren rasant verändert. Die Herausforderungen sind gewachsen durch eine steigende Anzahl von Akteuren und Geschäftsprozessen sowie viele neue Leistungen und Angebote, die den Service für den Fluggast verbessern und effizienter gestalten. Um diese zunehmende Komplexität beherrschen zu können und gleichzeitig in der Lage zu sein, sich auf ihre Kernkompetenzen zu konzentrieren, benötigen Flughafenbetreiber eine anspruchsvolle Informations- und Kommunikations-Technologie (ICT).

Zu den Anforderungen der Flughäfen an die ICT gehören u. a. die Bereitstellung auf voll integrierte Lösungen, standardisierte Anwendungen und Systemlösungen, die das zukünftige Wachstum unterstützen.

Das Total Airport Management System (TAMS) der T-Systems GEI ist auf diese Anforderungen ausgerichtet und unterstützt alle wesentlichen Prozesse und Abläufe zum Flughafenbetrieb. Von der Ressourcenplanung über die Disposition bis hin zur Simulation von Passagierströmen ist TAMS modular den Kundenanforderungen angepasst.

[1] Unter der Annahme, dass ein Produkt existiert

Zentraler Bestandteil eines TAMS ist die Airport Operational Database (AODB). Sie stellt sicher, dass alle Informationen, die über die Partnersysteme einfließen oder manuell eingepflegt werden, automatisch eingearbeitet werden und anschließend dem Bedienpersonal und den Kunden des Flughafens adressatengerecht zur Verfügung stehen. Für die Planung von immobilen Ressourcen wie z.B. Check-In-Counter, Abstellpositionen für Flugzeuge u.v.m. bietet T-Systems ein integriertes Resource Management System (RMS). Für übersichtliche Informationsanzeigen sorgt das neue Passagier- und Dienstinformationssystem, Flight Information Display System (FIDS). Dieses System bringt nicht nur großen Nutzen für die Passagiere, sondern auch für die Flughafenbetreiber, die die Anzeigetafeln zusätzlich für Werbung in den öffentlichen Terminalbereichen anbieten können.

Die genannten Lösungen stellen lediglich einen Ausschnitt des gesamten Lösungsportfolio der T-Systems zu TAMS dar (s. folgende Abbildung).

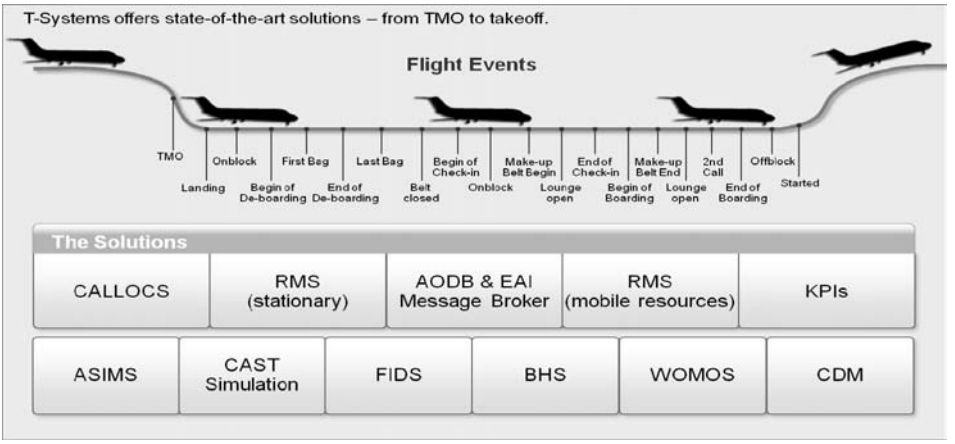

Abbildung 2: Lösungsportfolio der T-Systems zum Total Airport Management System

3 Entwicklung von Produktkerneln für Flughafensysteme

Grundlage für den Einsatz von Produktkerneln ist die Standardisierung. Dabei kann zwischen der **Prozess-Standardisierung**, d. h. der Standardisierung der Entwicklungsmethoden (z. B. Softwareentwicklungs-Prozess, Projekt Management-Prozess/ Projekt-Handling, Entwicklungs-Tools) und der **Technologie-Standardisierung**, d. h. der Standardisierung der technischen Lösung unterschieden werden.

3.1 Standardisierung der Softwareentwicklung (Prozess-Standardisierung)

Die Prozess-Standardisierung bildet die Grundlage für den Wandel von der projektorientierten zur Produktkernel-orientierten Softwareentwicklung. Dabei stehen besonders die Anwendung standardisierter Methoden zum Projekt Management und zum Software-Engineering im Fokus [SEB10, PMB10]. Weitere Aspekte der Prozess-Standardisierung sind in Abbildung 3 dargestellt.

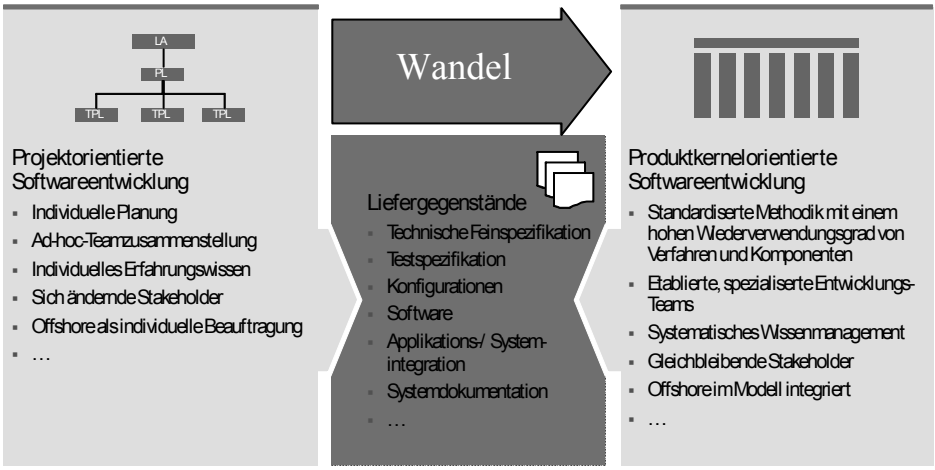

Abbildung 3: Vom Projekt- zum Produktmodell

Zu beachten ist, dass die Produktkernel-orientierte Softwareentwicklung keineswegs im Widerspruch zu der Berücksichtigung kundenindividueller Anforderungen steht.

Die Produktkernel-orientierte Softwareentwicklung führt zur Transparenz im Entwicklungsprozess und der Prozessreife und somit zu einer besseren Vorhersehbarkeit und Messbarkeit der Ergebnisse sowie einem deutlichen Zuwachs an Qualität und Effizienz. Dies ist im Interesse jedes einzelnen Kunden.

3.2 Standardisierung der Lösungen (Technologische Standardisierung)

Der Grad der Prozess-Standardisierung bei der Entwicklung von Flughafensystemen ist in der T-Systems bereits sehr hoch. Zur Bewältigung der oben genannten aktuellen Marktanforderungen der Flughäfen besteht jedoch zusätzlich die Notwendigkeit einer Technologie-Standardisierung.

Dabei ist es nicht das Ziel von T-Systems, das Flughafengeschäft zu einem hundertprozentigen Produktgeschäft zu entwickeln. Vielmehr soll weiterhin auf spezielle und individuelle Kundenanforderungen reagiert werden können. Deshalb bietet sich der Einsatz von Produktkerneln in Kombination mit der Anwendung einer standardisierten Softwareentwicklung für kundenindividuelle Systemanpassungen und Erweiterungen an. Damit werden die Vorteile beider Standardisierungsarten genutzt und gleichzeitig wird die Flexibilität der Systeme garantiert.

Die Entwicklung von Produktkerneln für Flughafensysteme unter Berücksichtigung der laufenden Kundenprojekte und bestehender Lösungen bedarf einer strategischen Planung.

Folgende Maßnahmen sind im vorliegenden Fall für eine Migration vom reinen Projektgeschäft zum Einsatz von Produktkerneln notwendig:

- Definition von Standard Delivery Elementen (SDE), die als Produktkernel angeboten werden.
- Prüfung des aktuellen Abdeckungsgrads (Current Mode of Operations) dieser SDEs bzgl. typischer Kundenanforderungen (funktional und nicht-funktional).
- Aufzeigen eines Ziel-Szenarios (Future Mode of Operations) und der Maßnahmen zur Entwicklung dieses Ziel-Szenarios.
    - Welche Systeme werden zu einem Produktkernel ausgebaut?
    - Welche Systeme werden in Produktkerneln integriert?
    - Welche Systeme werden stillgelegt?
- Definition einer Referenzarchitektur [SUM07] für jeden Produktkernel. Hierzu gehört z. B. die Festlegung von:
    - Entwicklungsplattformen
    - Betriebsplattformen
    - Kernel-Funktionalitäten
    - Nicht-funktionalen Eigenschaften
- Definition eines Life-Cycle-Management für Produktkernel.

3.3 Organisatorische Auswirkungen auf die Zusammenarbeit im Team

Die Bereitstellung von Produktkerneln hat Auswirkungen auf die Organisation und Zusammenarbeit von Projekt- und Entwicklungsteams [MCC96].

Folgende Abbildung stellt eine mögliche Rollenverteilung dar, die die Grundlage einer Zielorganisation sein sollte.

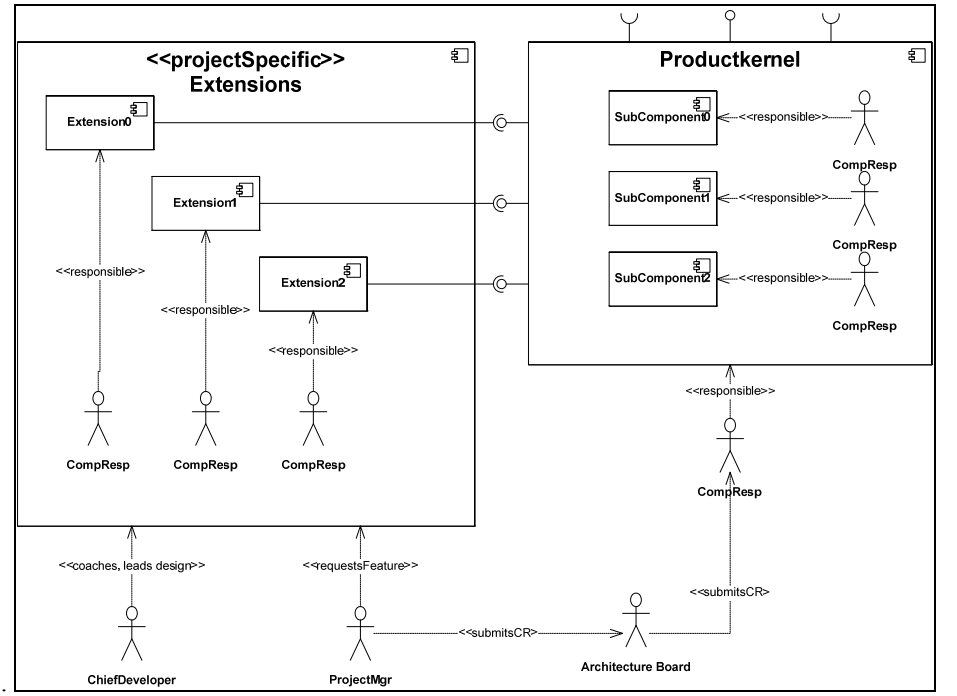

Abbildung 4: Rollenverteilung für die Bereitstellung von Produktkerneln

Relevant ist, dass für jeden Produktkernel ein Entwicklungsteam existiert, das für die domänenspezifische Entwicklung der Grundfunktionalitäten verantwortlich ist. Für die kundenspezifische Integration sind andere Projektteams zuständig.

Diese Trennung der Verantwortlichkeiten führt dazu, dass die Pflege des Source-Code eines Produktkernel nur an einer Stelle stattfindet. Alle Änderungswünsche der Projektteams an diesen Source-Code müssen über ein Architektur-Board bewertet und beim Produktkernel-Entwicklungsteam eingereicht werden. Das Architektur-Board dient somit als Eingangstor für alle Anforderungen an den Source-Code eines Produktkernel.

4 Zusammenfassung und Ausblick

Die Autoren sehen durch die Evolution der Systemintegration hin zu Lösungen auf Basis von Produktkerneln einen essentiellen Beitrag zur Wettbewerbsfähigkeit. Durch den hohen Reifegrad der Lösung kann der Kunde einen höheren Nutzen erzielen. Dies liegt insbesondere daran, dass ein Produktkernel in Abgrenzung zu einem Framework Domänenwissen und somit mehr Fachlichkeit und eine höhere Flexibilität beinhaltet. In Abgrenzung zu einem Produkt bzw. zu einer Produktlinie besteht der Vorteil in der höheren Flexibilität und in der Möglichkeit der Modelarisierung.

Die Mitarbeiter des Anbieters - sowohl im Entwicklungsteam für den Produktkernel als auch im individuellen Projektteam zur Erarbeitung der Kundenlösungen - haben zusätzliches Potential für den Wissensaustausch und den Wissensaufbau. Damit ist die Fokussierung auf ein Lösungsangebot mit Produktkerneln zukunftsträchtig.

Ausgehend von der Idee des Produktkernels und den Anforderungen an Flughafeninformationssysteme hat die T-Systems Systems Integration ein Modell entwickelt, bei dem Produktkernel erfolgreich in Projekten eingesetzt werden. Kernbestandteil sind die technische Standardisierung in Form des Produktkernels und die Standardisierung der Entwicklungsprozesse für ähnlich gelagerte Projekte.

Die positiven Auswirkungen des Kernelansatzes – insbesondere durch die Möglichkeit der Wiederverwendung - auf die Qualität, die Laufzeit und das Budget der Projekte im Flughafenbereich dienen als Referenz für weitere Geschäftsdomänen.

Somit wird dieses erfolgreiche Konzept in Zukunft auf weitere Fachbereiche der T-Systems Systems Integration ausgedehnt. Darüber hinaus sind zukünftig die Messung des Standardisierungsgrades und der technischen und prozessualen Wiederverwendung Bestandteil der Unternehmensentwicklung.

Auch wird durch dieses Konzept eine gute Voraussetzung für ein intelligentes und globales Sourcing- und Delivery-Model gelegt, so dass aktuellen Anforderungen zum Offshoring zur Sicherung der Wettbewerbssituation von Unternehmen begegnet werden kann.

Literaturverzeichnis

[BAZ06] L. Bass, et al., Software Architecture in Practice, Addison-Wesley, 2006.

[BUS01] F. Buschmann et al., Pattern Oriented Software Architecture Vol. 1, Wiley, 2001.

[Az99] Azubi, L. et.al.: Die Fußnote in LNI-Bänden. In (Glück, H.I.; Gans, G., Hrsg.): Formattierung leicht gemacht – eine Einführung. Format-Verlag, Bonn, 1999; S. 135-162

[CLE07] P. Clements, L. Northrop, Software Product Lines, Addison-Wesley, 2007.

[FOW03] M. Fowler, Patterns of Enterprise Architecture, Addison-Wesley, 2003.

[FOW04] M. Fowler, Inversion of Control Containers and the Dependency Injection pattern, http://martinfowler.com/articles/injection.html, 2004

[HIB10] Hibernate Relational Persistence for Java & .NET, http://www.hibernate.org/, 2010.

[MCC96] S. McConnel, Rapid Development, Microsoft Press, 1996.

[NYG07] M. T. Nygard, Release It!, Raleigh, 2007.

[PAR10] T. Parr, Language Implementation Patterns, Raleigh, 2010.

[PMB10] PMBook, T-Systems, 2010.

[QT10] Nokia, Qt Reference Documentation, doc.qt.nokia.com, 2010.

[SPR10] Spring Framework, www.springsource.org/documentation, 2010.

[SEB10] SEBook, T-Systems, 2010.

[SUM07] I. Summerville, Software Engineering, Addison-Wesley, 2007.