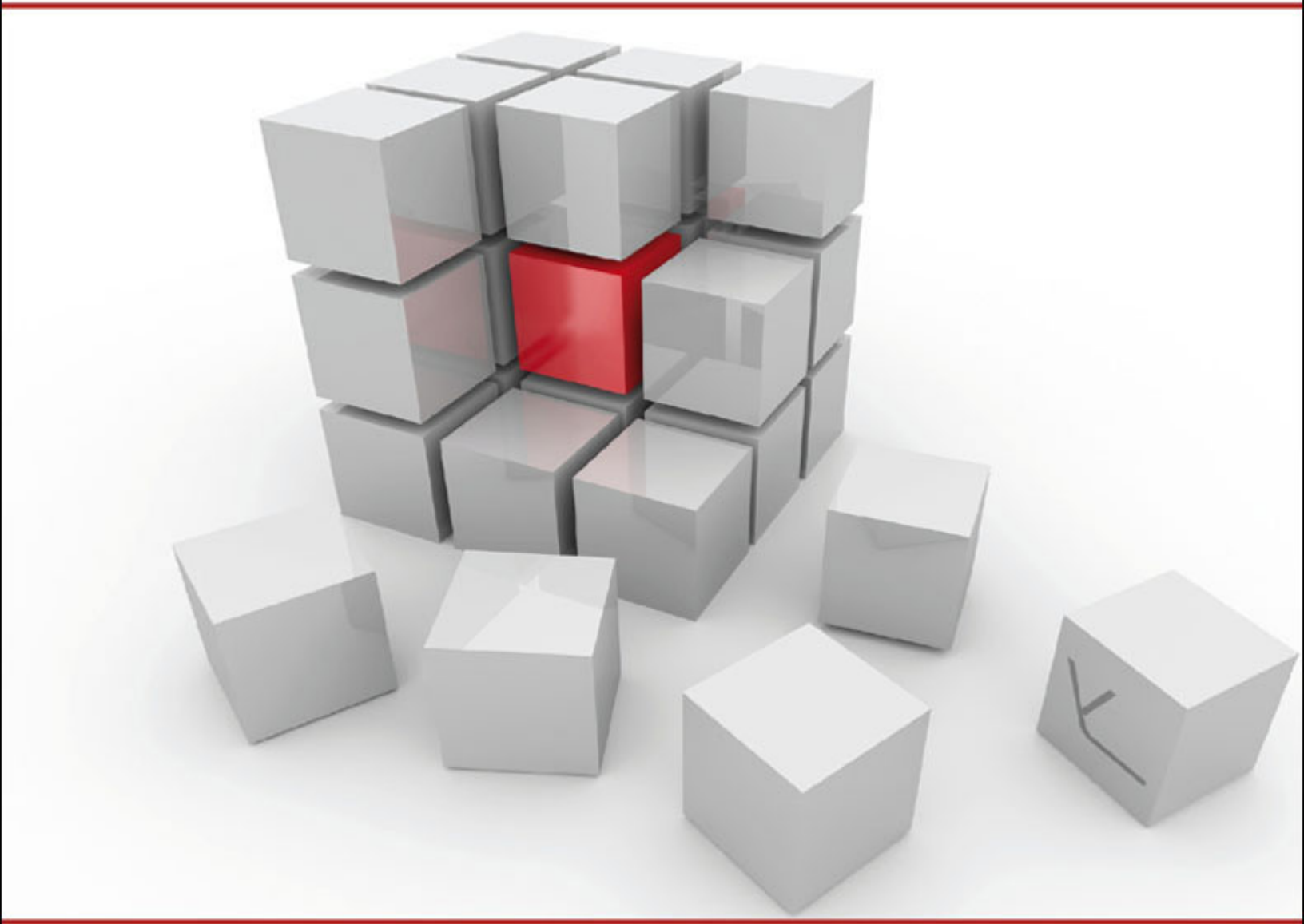
Synomic Academy

TECHNISCHE UNIVERSITÄT MÜNCHEN



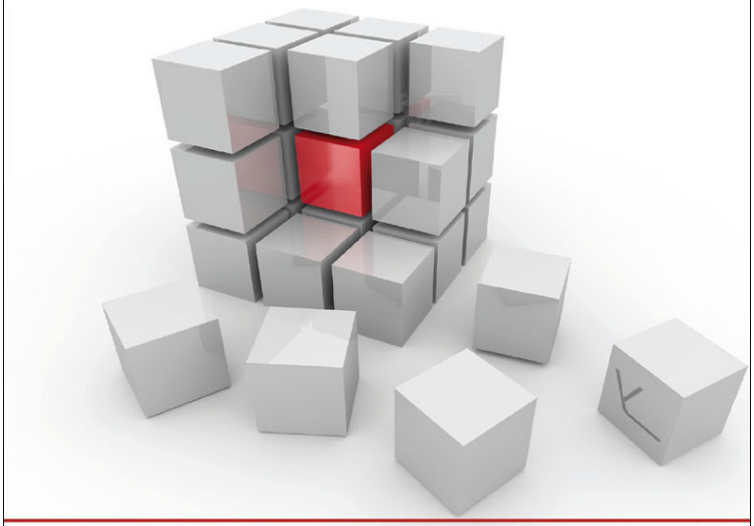
Josef Waltl

Intellectual Property (IP)

Modularity in Software Products and Software Platform Ecosystems

Synomic Academy

TECHNISCHE UNIVERSITÄT MÜNCHEN



Josef Waltl

Intellectual Property (IP)

Modularity in Software Products and Software Platform Ecosystems

TECHNISCHE UNIVERSITÄT MÜNCHEN

Dr. Theo Schöller-Stiftungslehrstuhl für Technologie- und Innovationsmanagement

IP Modularity in Software Products and Software Platform Ecosystems

Josef Waltl

Books on Demand

Vollständiger Abdruck der von der Fakultät für Wirtschaftswissenschaften der Technischen Universität München zur Erlangung des akademischen Grades eines Doktors der Wirtschaftswissenschaften (Dr. rer. pol.) genehmigten Dissertation.

Vorsitzender:
- Univ.-Prof. Dr. Stefan Minner

Prüfer der Dissertation:
1. Univ.-Prof. Dr. Joachim Henkel
2. Univ.-Prof. Dr. Oliver Alexy

Die Dissertation wurde am 24.01.2013 bei der Technischen Universität München eingereicht und durch die Fakultät für Wirtschaftswissenschaften am 12.02.2013 angenommen.

Life is like riding a bicycle. To keep your balance you must keep moving.

Albert Einstein (1879 – 1955)

Acknowledgements

First of all, I would like to thank Prof. Dr. Joachim Henkel for his outstanding support as a thoughtful supervisor. His enthusiasm and energy for the research on IP modularity were always inspiring to me and guided me throughout my whole dissertation.

This research project would never have been successful without the support from the researched software companies. For SugarCRM I would like to thank Elena Annuzzi, Nick Halsey, John Mertic and Clint Oram. For Salesforce.com the credits go to Kimia Poursaleh and for SAP special thanks goes to Dr. Karl-Michael Popp for his excellent support and all the effort he put into our joint research. I would also like to acknowledge the interview partners in a company whose name has to be kept undisclosed for the excellent support and the willingness to share their insights, despite the additional effort to control each statement for non-disclosure of confidential company information.

Furthermore sincere thanks are due to the team of the Dr. Theo Schöller Chair of Technology and Innovation Management for the inspiring time and the pleasant cooperation.

Special credits go to Christoph Krauß and Kristina Schreiner for their help in data collection and their valuable comments on my analysis results.

Finally, this dissertation would not have been possible without the never-ending support from Marianne, Josef, Martin and Claudia.

Table of Contents

List of Appendices

List of Figures

List of Tables

List of Abbreviations

AGPL – Affero GPL
APIs – Application Programming Interfaces
BSD – Berkeley Software Distribution
CDO – Chief Development Officer
CEO – Chief Executive Officer
CIO – Chief Information Officer
CRM – Customer Relationship Management
CTO – Chief Technology Officer
DSM – Design Structure Matrix
FOSS – Free- and Open Source Software
GNU – GNU's Not Unix
GPL – GNU General Public License
IP – Intellectual Property
IPRs – Intellectual Property Rights
ISV – Independent Software Vendor
JAR – Java Archive
MIT – Massachusetts Institute of Technology
OLS – Ordinary Least Squares
OSS – Open Source Software
SaaP – Software as a Product
SaaS – Software as a Service
SCRM – SugarCRM
SFDC – Salesforce.com
SI – System Integrator
VIF – Variance Inflation Factor
VP – Vice President

Zusammenfassung

Die Arbeit untersucht die Auswirkungen einer modularen Softwarearchitektur, die durch die Optimierung von Wertaneignungsmechanismen entstanden ist (IP Modularität), auf Softwareprodukte und Softwareplattform-Ökosysteme. Ein System ist IP-modular, wenn die internen Modulgrenzen so gezogen werden, dass die jeweiligen Module ausschließlich Elemente enthalten, die in Bezug auf geistige Eigentumsrechte identisch behandelt werden können. Diese Rechte können in Form von Lizenzrechten, Urheberrecht aber auch informell, zum Beispiel durch Geheimhaltung von Quellcode, in Erscheinung treten.

Die Ergebnisse dieser Dissertation basieren auf einer detaillierten qualitativen Fallstudienanalyse von Softwareprodukten und -plattformen sowie einer quantitativen Studie in zwei Plattform-Ökosystemen.

Durch das Aufzeigen eines direkten Zusammenhanges von IP-modularer Produkt- oder Plattformarchitektur mit dem entsprechenden Geschäftsmodell zur Wertaneignung erweitern die Ergebnisse der Arbeit die bestehende Literatur zu IP Modularität. Es wird gezeigt, dass eine IP-modulare Architektur eine partielle Offenheit erlaubt, die verteile Wertschöpfung begünstigt. Zusätzlich wurde die frühe Berücksichtigung von Anforderungen mit Bezug auf geistiges Eigentum in der Anforderungsanalyse (Requirements Engineering) von Softwaresystemen als wesentlicher Treiber zur Verhinderung von zeit- und kostenaufwändigen Re-Modularisierungen identifiziert.

Die Ergebnisse der quantitativen Studie zeigen, dass die Ausprägungen IP-modularer Plattformarchitekturen, durch die Möglichkeit zu größerer Offenheit, bei gleichzeitiger Sicherstellung der Wertaneignung, die Plattformattraktivität

für Designer plattformspezifischer Zusatzapplikationen erhöhen können.

Zusammenfassend zeigt die Arbeit die Verbindung von Management geistigen Eigentums, Softwarearchitektur und der jeweiligen Geschäftsmodelle von Softwareprodukt oder -plattform Anbietern auf.

Abstract

This dissertation examines the impact of Intellectual Property (IP) modular architecture on software products and software platform ecosystems. A software system is IP modular when its module boundaries separate parts of a system that have to be treated differently with respect to IP. The IP status is then homogeneous within each module, but may differ between modules. IP rights can be formal IP such as licensing contracts or copyright, but also informal IP like keeping the source code secret.

The presented results in this dissertation are based on a detailed qualitative case study analysis of two software products and two software platforms and on a quantitative study of two software ecosystems.

The results extend the existing literature on IP modularity by demonstrating a direct association between IP modular product or platform architecture and the related business models. The analysis also shows that the early consideration of IP-related requirements in the requirements engineering process of software systems can prevent costly and time-consuming re-modularizations.

The quantitative analysis in two software ecosystems shows that IP modular platform architecture, which can allow increased openness while still maintaining value appropriation, can increase a platform's attractiveness for complementors.

To summarize, this dissertation demonstrates the connections between IP management, software architecture and the respective business models of software product or platform providers.

1 Introduction

This dissertation explores the implications of the new concept of Intellectual Property (IP) modularity (Henkel and Baldwin, 2010, 2011) for the software product and software platform ecosystems domain. A system is IP modular when the module boundaries separate the parts that need to be treated differently with respect to IP. The IP status is accordingly homogeneous within each module, but it can differ between modules. IP rights can be formal IP, such as licensing contracts or copyright, but they can also be informal IP, such as keeping source code secret. This research endeavor focuses exclusively on the software domain because IP is the core asset of each software business, and the modularity of software systems can be adapted to a variety of requirements.

More broadly, this dissertation links research on IP modularity with research concerning software platforms (Gawer and Cusumano, 2002; West, 2003; Boudreau, 2010), multi-sided markets (Eisenmann *et al.,* 2006), software ecosystems (Jansen and Cusumano, 2012), software business models (Osterwalder, 2004; Weill *et al.,* 2005) and software requirements engineering (Wiegers, 2003; Chung and do Prado Leite, 2009).

To motivate the research on IP modularity Henkel and Baldwin (2010) consider, among others, the case of the video game Counter-Strike (Jeppesen and Molin, 2003). When the software publisher Valve Software released the video game Half-Life in 1998, it divided its codebase into two different modules. Valve Software put the game engine under a proprietary license and kept its source code secret, whereas it made the remaining application source code available to users under a broad license that allowed users to modify and share the code. Within approximately one and

a half years of the original release of Half-Life, users generated the game Counter-Strike, which surpassed the success of the original game and created significant additional revenue for Valve Software, given that Counter-Strike players had to license and reuse the Half-Life game engine. This example shows the potential benefits that IP modular design could have for companies in the software domain, where module boundaries are flexible and technological entry barriers for value co-creators are low.

Already initial interviews in the early stages of this research project with industry practitioners have revealed that IP modular design is of special relevance in software platforms, which by nature face the challenge of optimizing value creation in the whole ecosystem of complementors and value appropriation for the platform providers. The exchange with managers from software platform companies confirmed the link between their IP modular platform designs and their business strategies; the managers also confirmed the need for a better understanding of these IP mechanisms on a more conceptual level.

In the software domain, there are many examples of IP modularity, but little is known about the exact reasoning that led to those IP modular designs. To my knowledge this dissertation presents the first empirical study to shed light on the effects of IP modularity in the software domain. The main research objective of this dissertation is formulated as follows:

Research objective: *What are the effects of IP modularity on software products and software platform ecosystems?*

This dissertation aims to answer the main research objective through three different perspectives. First, there is the perspective of the providers of software products. Second, there is the perspective of the providers of software platforms. Third, there are the complementors who generate

additional applications for software platforms. Based on these perspectives, the structure of this dissertation is as follows:

Section 2 introduces IP modularity as the main theoretical concept of this dissertation. This section also presents the basics of system design and introduces the concept of modularity and the main drivers of modularizing technical systems. The section concludes by presenting IP modularity in software systems with concrete examples.

Section 3 presents the research methodology applied in this dissertation. First, in section 3.1 a hybrid research approach is identified as the most suitable method, given the current progress in the research field. In section 3.2 a detailed description of the case study methodology builds a solid methodological foundation for the in-depth qualitative research conducted in this dissertation. This section not only describes the applied research methodology, but it also guides the reader through the case selection process. A thorough understanding of this section is therefore vital for the interpretation of the case results. Finally, section 3.3 describes the quantitative analysis.

Section 4 addresses IP modularity from the perspective of a software product provider. First, the section presents the basics of software product design and software business models. The analysis of an engineering software case for outgoing IP modularity and of a data management software case for incoming IP modularity build the empirical foundation to answer the research questions regarding IP modularity in software products:

- Why are software products modularized with regard to IP considerations?
- How do the intended effects of IP modular product design relate to the real effects?

Finally, in this section, a cross-case comparison of the identified effects uncovers the similarities of both cases and leads to the formulation of additional propositions about the impact of IP modular software product design.

Section 5 uncovers the impact of IP modular design on software platform design from a software platform provider's perspective. Based on a review of the core concepts of software platforms and related ecosystems, two case studies on popular software platforms form the empirical basis to answer the research questions of this section:

- Why are software platforms modularized with regard to IP considerations?
- How does IP modular platform design influence the cooperation between platform providers and complementors?

The findings from the first case on SugarCRM confirm prior findings by Henkel and Baldwin (2010) and lead to the formulation of the research hypotheses for later quantitative tests. The second case on SAP NetWeaver PI suggests how the IP modular design can increase the attractiveness of a proprietary software platform. Finally, the cross-case analysis uncovers the common effects of IP modular platform design on platform development, platform attractiveness for complementors and platform attractiveness for end-users.

Section 6 follows up on the findings of the previous section with a quantitative research approach. It tests the findings from the qualitative case analysis on the levers of platform attractiveness. In this section, the perspective switches from the platform providers to the complementors. The analysis aims to answer the following research question:

- How does IP modularity influence the attractiveness of a software platform from a complementer's perspective?

The analysis is based on a platform attractiveness model[1] that describes the variables that influence the platform attractiveness in the platform provider setting and in the complementor setting. With reference to the qualitative findings from Section 5 and propositions on outgoing IP modularity from Henkel and Baldwin (2010) two hypotheses are formulated and tested with a regression analysis.

Finally, Section 7 draws conclusions for two target audiences. The first audience is the scientific community, for which this dissertation embeds the results in the broader discussion on related streams of research and makes suggestions for further research. The second audience are practitioners, such as general managers in software companies, ecosystem managers or software architects, for whom the managerial implications of IP modular design are discussed.

All results presented in this dissertation are based on my own work unless stated otherwise. All results from other researchers are carefully referenced. However, it is my deepest belief that creative ideas do not only sparkle in the mind of a single researcher. To reflect this belief and to recognize the efforts of my co-authors in earlier publications on the topic and other members of the research community to critically review my results, I purposely use "we" to present the results of this dissertation.

---

[1] The model is based on the Master's thesis of Schreiner (2012) that I initiated and supervised. For the analysis in this dissertation the original model was adapted and simplified.

2 The concept of IP modularity

In this chapter we introduce the concept of IP modularity as the central topic of this dissertation. We start with a basic description of design and introduce modularity as a design concept. Subsequently, we show the impact of modular design on value appropriation with a special focus on the software domain. Finally, we present IP modularity as a means to combine the benefits of modularity with the goal of value appropriation.

2.1 The basics of design

To understand modular design, it is important to first understand the basics of design. According to Baldwin and Clark, design is a complete description of an artifact (Baldwin and Clark, 2000, p. 21). The artifact can be a physical object or, as in the context of this dissertation, a virtual object such as a software source code. The design specifies all parameters of an artifact. All interdependencies between the design parameters define the design structure of an artifact (Baldwin and Clark, 2000, p. 21).

This design structure can be visualized with a tool named design structure matrix (DSM), invented by Steward (Steward, 1981) and refined by Eppinger (Eppinger, 1991). The use of design structure matrices can be illustrated with the simple example of the design of a mug (Baldwin and Clark, 2000, p. 21). The design parameters are named from P1 to P10, and the interdependencies are displayed with *X* marks in the DSM. These marks refer to an *is input to* relationship and mean that the specification of one column design parameter affects the design of the corresponding row design parameter (Eppinger, 1991, p. 285).

Figure 1 a) shows such a hierarchical relationship in the mug example. The manufacturing process influences the possible height but not vice versa. For the design process of the mug, this influence implies that manufacturing process can be specified first, regardless of the height. Once the manufacturing process is defined, the height cannot be chosen without restrictions. The design process is strictly sequential.

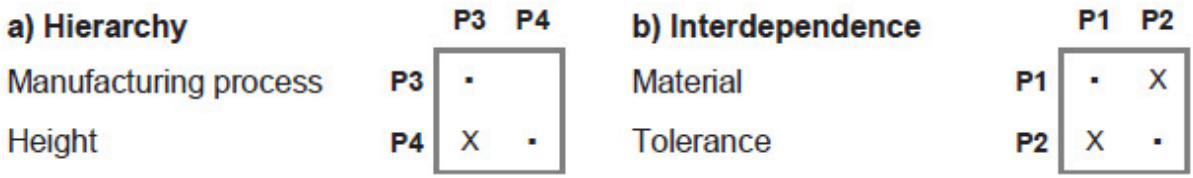


Figure 1 – Design structure (based on Baldwin and Clark, 2000)

If the design parameters mutually depend on each other, the relationship is interdependent as shown in Figure 1 b). Here, the material and the tolerance depend on each other. A change in the specification of one parameter requires the designer to adapt the specification of the other parameter. Design parameters can also be fully independent from one another, as in the case of the height and tolerance in the mug example. The full design structure matrix of the mug example is shown in Figure 2 which presents an input-output table of design parameter choices (Baldwin and Clark, 2000, p. 41).