I-Pattern for Gap Analysis

Philipp Gringel, Matthias Postina

Software Engineering for Business Information Systems
OFFIS - Institute for Information Technology
Escherweg 2
26121 Oldenburg, Germany
{philipp.gringel, matthias.postina}@offis.de

Abstract: Patterns are an effective way to express solutions to constantly occurring problems in software engineering. Books like Erich Gamma's *Design Patterns* ([Gam94]) or Martin Fowler's *Analysis Pattern* ([Fow96]) helped to develop a common understanding of problem solving in this field. The pattern catalog developed by members of the chair Software Engineering for Business Information Systems (sebis) of TU-Munich ([BELM08]) addresses patterns related to reoccurring problems in enterprise architecture management (EAM) and the authors are still working on this catalog. [BELM08] mentions several concerns related to SOA but is missing patterns concerning the initiation of SOA. This contribution will elaborate on an I-Pattern related to SOA-initiation on the basis of [EHH+08]. The focus lies on the problem of gap analysis on application landscapes as stated in [PSS09] and will further add metrics for gap analysis to this I-Pattern.

1 Introduction

The paradigm of service-oriented architecture (SOA) has undergone a remarkable change. Considered as a technical solution for a long time, SOA is today regarded as a conceptual approach able to align enterprise IT systems with the business strategies and processes they are supposed to support. SOA success stories in large enterprises indicate that service orientation is an effective way to integrate legacy systems as reusable building blocks into a flexible enterprise architecture. However, especially for small and medium sized enterprises, SOA is a far cry from being already implemented as mainstream. So strategies for initiating SOA in enterprises (like [EHH+08], [AGA+08]) are still subject of interest. The pattern catalog ([BELM08]) addresses concerns related to SOA (C-51, C-61, C-62, C-64, C-65, C-66, C-67, C-71 for example) but is missing patterns concerning the initiation of SOA. Our contribution will elaborate on an I-Pattern supporting a specific aspect of the process of SOA-initiation on the basis of Quasar Enterprise ([EHH+08]). We focus on the problem of gap analysis on application landscapes as stated in [PSS09] and will further add metrics for gap analysis to the I-Pattern.

1.1 Intended Audience

This article about I-Pattern for Gap Analysis is intended for people concerned with planning the enterprise evolution - like an enterprise architect for instance. The described pattern focuses on the problem of performing a gap analysis between two states of the application landscape, where the current state has a pre-SOA status and the envisioned state should be designed according to the principles of Quasar Enterprise.

2 Gap Analysis Concern

The term gap analysis is used in context of enterprise architecture as a name for the comparison between two architectures or strictly speaking two states of the same architecture. The Open Group Architecture Framework (TOGAF) uses the terms of baseline- and target-architecture for these two states ([The09]).

In our case we follow the methodology of [EHH+08] and focus on SOA initiation, so we compare the current (non-SOA) landscape of an organization to an envisioned landscape (Ideal-SOA). The concern of gap analysis in this context means to understand the architectural differences between current and ideal landscape in order to get an idea of the characteristics which have to be improved on the way towards an ideal service-oriented landscape. When we use the term ideal, we mean ideal in terms of being designed corresponding to the principles of Quasar Enterprise.

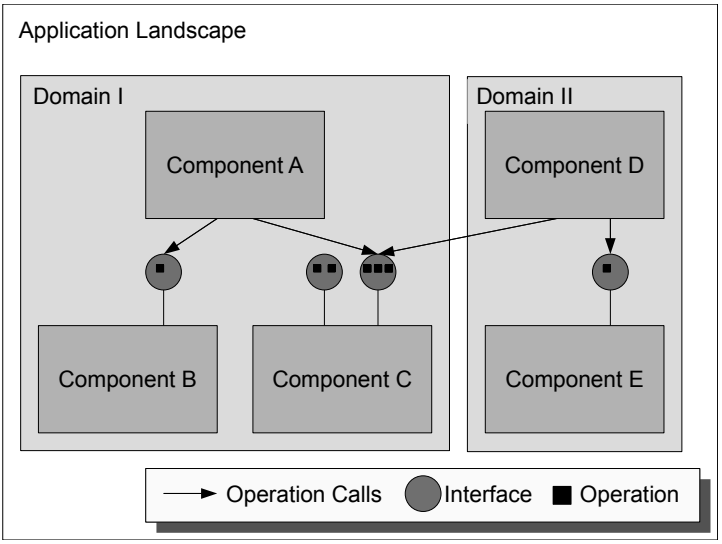

Figure 1: Ideal Application Landscape - Schematic Overview

Figure 1 provides a schematic overview of an ideal application landscape designed in ac-

cordance to [EHH+08]. Thereafter, service capabilities of an organization are provided by application components as operations encapsulated by services and described by interface descriptions. Application components are grouped by domains and domains are designed by functional decomposition of the enterprise. This means that [EHH+08] consider SOA initiation as a business driven process. Concerning gap analysis, we are observing the structural deviation between the current and ideal state of the application landscape.

To establish this understanding, we define in section 3 the data-structures describing the relationship between current and envisioned landscape as an I-Pattern. Based on this pattern, we present structural metrics which are used to quantify the differences between landscape states. Basically, we follow the definitions of [EHH+08] and partially extend the metrics defined there.

2.1 Forces

The problem of gap analysis on application landscapes described in this pattern could be summarized by the following question:

How do I need to restructure the current application landscape to converge towards an ideal application landscape designed according to the principles of Quasar Enterprise?

The following forces influence gap analysis on application landscapes:

1. **Qualitative assessment and quantitative assessment** The outcome of the gap analysis is an action list for possible measures to overcome the structural deviation between current and ideal application landscape. It deserves mentioning that this list reflects the quantitative assessment only and qualitative assessment could possibly complement this.

2. **Considering entire application landscape or focus on selective components** The gap analysis focuses on operations as atomic building blocks to detect structural deviations between current and ideal application landscapes. This is a detailed level of description when it comes to modeling both landscapes. The amount of needed data has to be considered carefully.

3. **Regarding state snapshots for gap analysis or monitor the distance over time** When performing gap analysis, two states - current and ideal application landscapes - are compared. Measures of the action list correspond to the outcome of this comparison. The architect needs to consider the circumstances of architectural work. Is the gap analysis part of a strategic enterprise architecture initiative or is it executed for a single comparison only?

3 I-Pattern

”An I-Pattern contains an information model fragment including the definitions and descriptions of the used information objects” ([BELM08]). According to this definition we wrote down an I-Pattern to support the gap analysis concern.

Following TOGAF and Quasar Enterprise, the **ideal application landscape** (ideal AL) should be designed according to the future / ideal business architecture. So the top level of an ideal AL is organized by domains grouping business functionality. Domains are nestable and are hosting application landscape components. These components can be regarded as - possibly only logically existing - application systems. A component provides public interfaces bundling a number of operations in order to implement a number of application services which are used by business services inside a business process dealing with business objects. Furthermore, operations are assigned to categories, which are ordered as follows:

1. *Interaction* services dealing with user interaction or interaction with other ALs.
2. *Process* services supporting business processes.
3. *Function* services supporting business processes and having an algorithmic character.
4. *Data* services accessing stored data.

This categorization is a typical means for structuring service-oriented landscapes and can also be found in other service-oriented approaches (see e.g. [KBS06]). Over the application service interfaces, the categorization can also be propagated down to the operations' level and is, by definition of the term ideal AL in [EHH+08], also applicable for components. The ideal AL includes only components which are constructed in a way that they solely bundle services of a certain category. Quasar Enterprise postulates that interfaces of a specific category are only callable by components of the same category-level or by components of higher categories, with ”interaction” being the highest and ”data” being the lowest category. So, a component categorized as function component, hosting only application services of the function category, can legally call function services or data services for instance.

The **current application landscape** (current AL) is also divided into application components which in practice usually means that these are tantamount to existing physical applications. In contrast to the ideal AL, current ALs are usually not structured in the sense of business domains. Instead, they are either structured by organizational units (e.g. corresponding to departments) or are not structured at all. Due to this absence of a domain structure, we refrain from modeling domains in current ALs. Just like ideal components, current components bundle functionality organized as operations, which are again the atomic functional building blocks. Since service is a term used in the context of the ideal AL only, we will further focus on operations as least common comparable units when developing metrics to perform the gap analysis.

At a certain point in planning the evolution of the current AL, the architect needs to define intermediate states of the AL as steps of the migration road map. Being in between the current AL and the ideal AL, such an intermediate state is referred to as **target application landscape** (target AL).

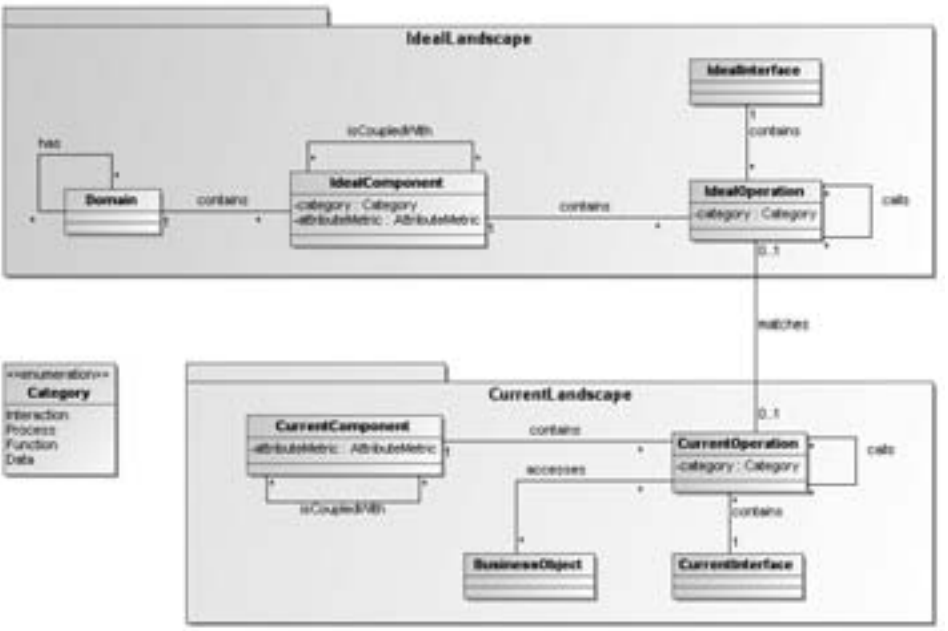

Figure 2: I-Pattern for Gap Analysis (according to [Sec09]).

The corresponding I-Pattern to perform gap analysis consists of the following entities (see also Figure 2):

Ideal AL:

1. **Domain -** Domains are grouping high level business functionality of an ideal AL. They describe the top level work performed by an organization or an organizational unit and - when nested - also high level functionality. Typically enterprises organizational structures are organized by domains. Examples for domains include: Customer Relationship Management, Partner Relationship Management, Sales, Marketing, Product Management. The set of all domains builds the ideal application landscape.

2. **Ideal Component -** Ideal components belong to a specific domain and are hosting the atomic building blocks of an AL - operations. Hence, ideal components do also group business functionality but on a more fine grained basis than domains. Components are often considered as logical applications which reflects their granularity quite well. Ideal components only bundle operations of a specific category (*interaction, process, function, data*) and are classified corresponding to the category of

their contained operations.

3. **Ideal Interface -** Ideal interfaces describe services offered by ideal components on a logical level. Synonymously to web-services on the physical level, interfaces offer a public description of one or multiple operations.

4. **Ideal Operation -** An ideal operation is the atomic entity of an ideal AL. It is the fine grained logical description of business functionality on a method or function level (CRUD operations for instance). An operation is classified according to a specific category (*interaction, process, function, data*).

Current AL:

1. **Current Component -** Current components describe the business functionality of the current AL. In practice, this means they describe the functionality provided by existing applications. In contrast to the ideal AL, current components are not grouped by domains, since domain is a term to express the future (ideal) high level functionality of an enterprise.

2. **Current Interface -** Current interfaces describe services offered by current components on a logical level.

3. **Current Operation -** Like ideal operations, a current operation is the atomic entity of the current AL and is also the fine grained logical description of business functionality on a method or function level. A current operation is classified according to a specific category (*interaction, process, function, data*).

4. **Business Object -** Business objects represent the entities relevant in the business domain. Examples for business objects include: customer, contract, order, invoice. Only current operations of category *data* should have writable access to business objects - ideally exactly one operation per business object (see metric *Data Sovereignty* for details).

4 Metrics

We identified metrics to measure the distance between the current application landscape and the ideal landscape. We subdivide metrics into quantitative and qualitative metrics. Quantitative metrics consider structural differences and lead to concrete recommendations for restructuring the current application landscape towards the ideal landscape. Qualitative metrics complement quantitative metrics and highlight critical components of the current application landscape. In a nutshell, quantitative metrics show what needs to be restructured, whereas qualitative metrics focus on what needs to be given special attention to by the architect when the restructuring takes place. Qualitative metrics are important but they are predominantly expressed by attributes (see [PSS09] for more details) and are not directly connected to the I-Pattern, so we discard them here. The following quantitative metrics are based on [EHH$^+$08] and directly applicable to the I-Pattern.

- *Completeness* means, that the functionality provided by the current application landscape is measured and compared to the functionality provided by the ideal application landscape. Such comparison is performed regarding operations of both, current and ideal landscape. Thus, the ratio is used as a key indicator for completeness calculated as follows:

$$\frac{\#\text{OP}_{tALnotImpl}}{\#\text{OP}_{tAL}},$$

in which $\#\text{OP}_{tALnotImpl}$ describes the number of ideal landscape operations not implemented in the current landscape, and $\#\text{OP}_{tAL}$ describes the overall number of ideal AL operations

- *Purity of the Domain* enunciates the degree of unambiguousness of the mapping of current AL components to ideal AL domains. The performance indicator is:

$$\frac{\#\text{cComp}_{gtOneDom}}{\#\text{cComp}},$$

in which $\#\text{cComp}_{gtOneDom}$ describes the number of current AL components hosting AL operations mapped to more than one ideal domain. The overall number of current AL components is summarized by $\#$cComp.

- *Purity of Categories* describes the degree of unambiguousness of the mapping of current AL operations to a certain service category. The corresponding performance indicator is:

$$\frac{\#\text{cComp}_{OPdiffCat}}{\#\text{cComp}},$$

in which the number of current AL components hosting AL operations of different categories is described by $\#\text{cComp}_{OPdiffCat}$ and the overall number of current AL components is decribed by $\#$cComp respectively.

- *Data Sovereignty* measures the degree of unambiguousness of the mapping of business objects to current AL components having writable access to their data representation. The performance indicator is:

$$\frac{\#\text{BO}_{nw} + \#\text{BO}_{w}}{\#\text{cComp}},$$

in which the number of business objects being writable by non data components is described by $\#\text{BO}_{nw}$. The number of business objects being writable by more than one component is summarized by $\#\text{BO}_{w}$. $\#$cComp represents the overall number of current AL components.

- *Correct Category Dependencies* measures the relative amount of illegal service calls between components of different categories according to the definition of callable as stated in [PSS09]. A metric is indicated by:

$$\frac{\#\text{ISC}_{\text{cComp}}}{\#\text{SC}_{\text{cComp}}},$$

in which the number of illegal service calls between current AL components is represented by $\#ISC_{cComp}$. The overall number of service calls between current AL components is described by $\#SC_{cComp}$.

Quantitative metrics can be used to quantify violations described by violation patterns of structural deviation between current and ideal AL. Each violation pattern implies measures to solve the deviation. To give an example, we refer to Figure 3.

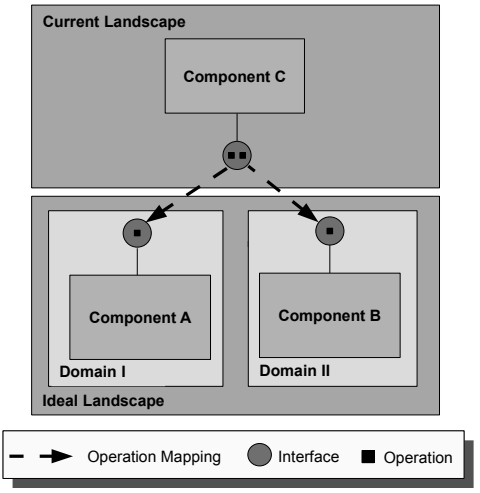

Figure 3: Purity of the Domain Violation.

It shows the violation of domain-purity quantified by the the metric *Purity of the Domain*. As part of the design phase of the ideal AL, the functional decomposition of operations from Component C of the current AL leads to a situation, where existing service functionality should be split in order to separate functional concerns for instance. In the planning phase of the ideal AL, both operations of Component C are assigned to new components of different domains. The I-Pattern allows a mapping of operations between two states of the application landscape. So this structural deviation between current and ideal AL can automatically be detected by algorithms on data having the structure of the I-Pattern, when gap analysis is performed. The metric *Purity of the Domain* is able to quantify the domain purity violations and can be used as indicator for this aspect of structural deviation. The suggested measure to overcome the deviation in this case would lead to the suggestion to initiate a project to split Component C.

5 Example

To clarify the abstract I-Pattern of gap analysis, we would like to give brief inside into the case study this pattern stems from. We would like to introduce our prototype for

gap analysis and the underlying data structure. For more information about the entire methodology and the actual case study we refer to [PGS09].

The case study is taken from the utility domain, where the current AL needed to be compared to an ideal AL. To manage such complex task, a prototype was developed to support the architect. This prototype is able to read a formalized representation of both, ideal AL and corresponding current AL (see Figure 4) to allow a manual semantic mapping of operations and to perform a gap analysis on these data structures.

Figure 4: Prototype to support gap analysis.

After importing the current and the ideal AL, the enterprise architect can see a graphical representation of the ALs inside the tool, and he is able to start the gap analysis by weighting the quantitative metrics according to enterprise specific preferences (Figure 4 shows the visualization of both ALs inside our prototype). These preferences can be stored by the tool and hence be reused in future analysis. Based on these metric weights, the tool calculates the structural deviation between current and ideal AL and highlights gaps inside the graphical representation of the landscapes if required. The prototype has the ability to calculate measures to resolve the structural deviation. Such measures are summarized as an action list inside the tool and a detailed description is available. Each measure is bound to a certain deviation observed by a specific metric and the position in the action list is determined by the weight of the metrics which assures an adequate prioritization of measures. Considering the metric *purity of the domain* a suggested action might be to split or to merge components of the current landscape to achieve a higher *purity of the domain*.

We use XML to store the elements of our ALs. The left side of figure 5 depicts a representation of the ideal AL whereas the current AL is described by the XML-file on the right side of the figure. The partitionig of ALs as introduced in section 3 is considered by the XML-file's structure. Component-elements can be used as child-elements of domain-elements for instance.

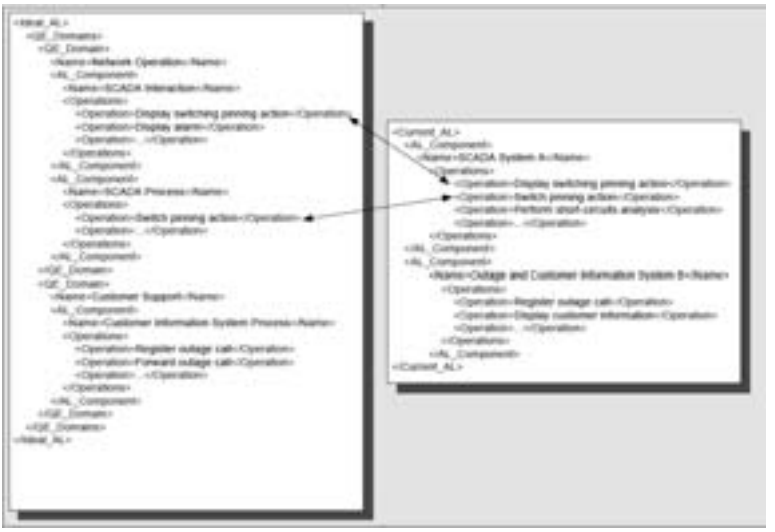

Figure 5: Rough data structure of ALs as used for structural mapping and gap analysis inside the prototype.

All quantitative metrics are calculating indicators for structural deviation between two data structures. Figure 6 illustrates the metric *Completeness*. The ideal AL-operations not implemented in the current AL are highlighted in red (*Display alarm*, *Register outage call*). The number of such current AL-operations divided by the number of the overall Ideal operations lead to the indicator of completeness.

According to [PSS09] all quantitative metrics form the overall distance between current and ideal AL. The tool is able to calculate the overall distance value and can record distance values over time for long term analysis (for example to indicate progress when the current AL evolves and is compared to the ideal AL again).

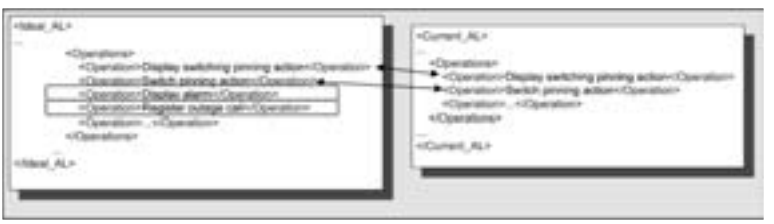

Figure 6: Violation of the completeness metric.

6 Consequences

We would like to consider some consequences of applying the gap analysis pattern on the basis of the forces described in section 2.1.

1. **Qualitative assessment and quantitative assessment**
   A quantitative assessment of the differences between the as-is and ideal state produces manageable key figures and directly leads to a first list of possible needs for actions to be taken in order to converge towards the ideal state. Such assessment should be complemented by a qualitative assessment comprising more general IT strategic considerations, for example with regard to strengths, weaknesses, chances or risks of the current landscape. The results of the quantitative and qualitative assessment are weighed against the short-term operative goals to finally come to a workable list of actions to be taken. When performing the gap analysis, all these considerations need to be taken into account by an architect as part of an overall EA development process.

2. **Considering entire application landscape or focus on selective components**
   The amount of data necessary to perform gap analysis on the entire application landscape on a detailed level considering operations is overwhelming. Unless data is automatically collectible - which has never been the case for pre-SOA landscapes in our daily work observations- it is not realistic to model the entire application landscape. However, this pattern is beneficial for partial consideration of the application landscape, when single systems need to be restructured, for instance.

3. **Regarding state snapshots for gap analysis or monitor the distance over time**
   Gap analysis is a data intensive analysis and the manual detection of structural deviations is a time consuming task. Whenever gap analysis is a reoccurring task for an enterprise architect, tool support is advisable. This is especially true, when progress of architectural work should be measured continually. In this case, single comparison of two states is insufficient.

7 Conclusion and Outlook

In our contribution, we introduced the task of gap analysis and elaborated on an I-Pattern for gap analysis on application landscapes. We described how a gap analysis between a current non SOA AL and service oriented ideal AL can be performed based on a formal I-Pattern and supported by a software prototype in order to derive a suitable target application landscape.

In this contribution we focused on the shift from non SOA toward SOA. However, a growing number of enterprises is going to introduce or has already established SOAs. Gap analysis is also of relevance at evolution on established SOAs. Ongoing research is aiming in this direction and we are currently working on evolution patterns for SOA.

References

[AGA+08] Ali Arsanjani, Shuvanker Ghosh, Abdul Allam, Tina Abdollah, Sella Ganapathy, and Kerrie Holley. SOMA: A method for developing service-oriented solutions. *IBM Systems Journal*, 47(3):377–396, 2008.

[BELM08] Sabine Buckl, Alexander M. Ernst, Josef Lankes, and Florian Matthes. Enterprise Architecture Management Pattern Catalog. Technical Report TB 0801, Software Engineering for Business Information Systems (sebis), Technische Universität, February 2008.

[EHH+08] Gregor Engels, Andreas Hess, Bernhard Humm, Oliver Juwig, Marc Lohmann, Jan-Peter Richter, Markus Voß, and Johannes Willkomm. *Quasar enterprise: Anwendungslandschaften serviceorientiert gestalten*. dpunkt.verlag GmbH, Heidelberg, 1. edition, 2008.

[Fow96] Martin Fowler. *Analysis Patterns Reusable Object Models*. Pearson Education (US), 1996.

[Gam94] Erich Gamma. *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley, Reading Mass. [u.a.], 2. print.. edition, 1994.

[KBS06] Dirk Krafzig, Karl Banke, and Dirk Slama. *Enterprise SOA: Service-oriented architecture best practices*. The Coad series. Prentice-Hall, Upper Saddle River, NJ, 6. print edition, 2006.

[PGS09] Matthias Postina, José Gonzalez, and Igor Sechyn. On the Architecture Development of Utility Enterprises with Special Respect to the Gap Analysis of Application Landscapes. In Ulrike Steffens, Jan Stefan Addicks, Matthias Postina, and Niels Streekmann, editors, *Proceedings of the 3rd Workshop "MDD, SOA, and IT Management"*, Oldenburg, Germany, September 2009. GITO.

[PSS09] Matthias Postina, Igor Sechyn, and Ulrike Steffens. Gap analysis of application landscapes. In Vladimir Tosic, editor, *Enterprise Distributed Object Computing Conference*, pages 274–281. IEEEXplore Digital Library, 2009.

[Sec09] Igor Sechyn. *Komponentenentwurf für Anwendungslandschaften am Beispiel eines Energiekonzerns. Ein werkzeuggestützter Abgleich zwischen Ist-Landschaft und Ideal*. Diplomarbeit, Carl von Ossietzky Universität, Oldenburg, 2009.

[The09] The Open Group. *TOGAF Version 9*. van Haren Publishing, 2009.