**Institute of Architecture of Application Systems**

# Capturing Cloud Computing Knowledge and Experience in Patterns

Christoph Fehling, Thilo Ewald, Frank Leymann,
Michael Pauly, Jochen Rütschlin, David Schumm

Institute of Architecture of Application Systems
University of Stuttgart
Stuttgart, Germany
{fehling, leymann, schumm}
@iaas.uni-stuttgart.de

Daimler AG
Stuttgart, Germany
jochen.ruetschling@daimler.com

T-Systems International GmbH
Frankfurt, Germany
{thilo.ewald, michael.pauly}
@t-systems.com

BibTeX:

```
@inproceedings{Fehling12,
  author    = {Christoph Fehling and Thilo Ewald and Frank Leymann and
               Michael Pauly and Jochen Rütschlin andDavid Schumm},
  title     = {Capturing Cloud Computing Knowledge and Experience in Patterns},
  booktitle = {Proceedings of the 5th IEEE International Conference on
               Cloud Computing, CLOUD 2012},
  year      = {2012},
  pages     = {726--733},
  doi       = {10.1109/CLOUD.2012.124},
  publisher = {IEEE Computer Society}
}
```



**Universität Stuttgart**
Germany

# Capturing Cloud Computing Knowledge and Experience in Patterns

Christoph Fehling[1], Thilo Ewald[3], Frank Leymann[1], Michael Pauly[3], Jochen Rütschlin[2], David Schumm[1]

[1] Institute of Architecture of Application Systems
University of Stuttgart
Stuttgart, Germany
{fehling, leymann, schumm}
@iaas.uni-stuttgart.de

[2] Daimler AG
Stuttgart, Germany
jochen.ruetschling@daimler.com

[3] T-Systems International GmbH
Frankfurt, Germany
{thilo.ewald, michael.pauly}
@t-systems.com

***Abstract*— The industry-driven evolution of cloud computing tends to obfuscate the common underlying architectural concepts of cloud offerings and their implications on hosted applications. Patterns are one way to document such architectural principles and to make good solutions to reoccurring (architectural) cloud challenges reusable. To capture cloud computing best practice from existing cloud applications and provider-specific documentation, we propose to use an elaborated pattern format enabling abstraction of concepts and reusability of knowledge in various use cases. We present a detailed step-by-step pattern identification process supported by a pattern authoring toolkit. We continuously apply this process to identify a large set of cloud patterns. In this paper, we introduce two new cloud patterns we identified in industrial scenarios recently. The approach aims at cloud architects, developers, and researchers alike to also apply this pattern identification process to create traceable and well-structured pieces of knowledge in their individual field of expertise. As entry point, we recap challenges introduced by cloud computing in various domains.**

***Keywords- cloud computing, architecture, patterns***

## I. INTRODUCTION

Through the use of cloud computing, cloud providers and their customers benefit from the fundamental cloud properties: elasticity, pay-per-use, standardization, and resource sharing. Elasticity empowers cloud users to reserve and release cloud resources dynamically and based on the currently experienced workload. Pay-per-use pricing models ensure that only the temporarily used resources are billed to customers. The broad availability of reusable cloud services cause cloud applications, their components, and used middleware to be standardized and homogenized. This significantly reduces the complexity of cloud application runtime environments and, together with the enabled resource sharing between cloud users, makes cloud resources a generally available commodity. For these properties, cloud computing has been recognized as one of the key IT topics for the next years [39] [5] due to (i) its ability to cope with extremely large amounts of users, (ii) its ability to handle vast amounts of data, (iii) its flexibility when these amounts change suddenly, and (iv) its customizability regarding various business requirements. To benefit from this powerful computing environment, application architects and developers need to follow certain architectural and management best practices. The dynamicity of the runtime infrastructure [1], replication of data [14], handling of resource failure [15], tenant isolation [5], and the bridging of different cloud environments [24] have to be incorporated in the applications architecture and runtime behavior. Many of these challenges are also faced in non-cloud applications in a similar manner. However, the strong interrelation of cloud applications with the runtime environment provided as-a-Service makes clouds a very complex and diverse environment. Building and executing cloud applications without considering the specifics of this environment may result in suboptimal resource utilization and may even reduce applications' availability [1].

To tackle the complexity of the cloud environment, we propose to use patterns, a well-established concept to describe pieces of knowledge [2] to capture architectural styles and best practices of cloud computing. We understand these *cloud patterns* as human-readable documents sharing a common format. The patterns are interrelated in a structured manner and ensure an abstraction of implementation details and use case specifics to describe reusable solutions to reoccurring problems. The approach addresses the deficit that the industry-driven evolution of cloud computing technologies often results in an *implicit* application of architectural patterns. Such implicit use of patterns generally makes cloud offerings hard to compare, because the common underlying architectural principles remain non-public or are obfuscated through offering-specific terminology and components. In previous works [3] [4] [25], we have already introduced several patterns capturing *cloud architecture principles* and *management styles* common for different cloud providers. Because these providers offer large portions of the applications' functionality and runtime environments as-a-Service we have used one and the same format to also capture the concepts and behavior of different *cloud types* and *cloud offerings*. Using just one format enables the categorization of cloud providers regarding the patterns they support and sets the context for architectural patterns to be implemented by application developers. The use of a common pattern format, further, eases perception [45] and, thus, increases cloud technology adoption.

To enable other experts to capture their knowledge about cloud computing in this form, we cover in this paper how cloud architectural patterns can be identified, structured,

organized, and provided as teaching materials. We aim at supporting the following groups and their concerns with a concise cloud pattern-based approach. First, presentation of information in a pattern format enables *application architects* to quickly absorb a large body of knowledge and to understand the impact that cloud computing, associated technologies, and products have on application architectures. The patterns, thus, capture how architectural decisions impact the functional and non-functional properties of cloud applications. The interrelations between patterns for the cloud environment and those for cloud applications describe cross-cutting concerns between application architecture and the runtime environment. Second, cloud patterns provide abstract knowledge about cloud products to *application developers*. Quickly changing and evolving cloud application implementations and available cloud offerings may be easier coped with, because abstract template solutions, contained in the pattern descriptions, may be applied to different cloud technologies. A pattern catalog with a well-defined semantic of pattern relations additionally helps developers during the discovery and use of patterns applicable in their concrete use cases. Third, abstract pattern-based descriptions of concepts can be used in *education* and ensure a longer persistence and relevance of obtained knowledge, because of their focus on abstracted information rather than on concrete implementations and offerings.

Regarding the research design and methodology, we provide in this paper a blueprint for the identification and compilation of cloud patterns in different architecture domains to guide experts. By following a verified pattern identification guide that was peer-reviewed by the pattern community [3], the process result is traceable and the identification re-doable. We, further, present an expert-reviewed cloud pattern format to readily express cloud architectural concepts to speed up adoption of the pattern approach in the cloud community. Finally, we give an overview of key cloud architectural domains in which pattern identification is promising. For each domain, we point to existing cloud architectural patterns and give pointers to other patterns discovered in them. The major contributions of this paper are (i) a clear process for the traceable identification of cloud patterns, (ii) a flexible and concise pattern format, (iii) two patterns capturing our recent findings about cloud application architecture, and (iv) an entry point to target domains where the approach may be applied by experts.

The remainder of the document is structured as follows: Section II covers our motivation from a practical point of view, describing two architectures from different cloud providers which do not share a common pattern approach. Section III presents the do-it-yourself identification process for cloud patterns. Two patterns we abstracted from the motivating examples are given in Section IV. Section V points to further domains we identified as key for future cloud pattern research. Related work is discussed in Section VI. Section VII concludes the paper and lists the lessons learned.

## II. MOTIVATION

In industry, cloud providers already make use of pattern-based descriptions and provide vendor-specific pattern formats and graphical notations. These patterns can be used to model cloud applications hosted on these providers' clouds. In the following, we use the graphical notation of patterns for Microsoft Windows Azure [6] and Amazon AWS [8] to describe one and the same exemplary scenario for both providers.

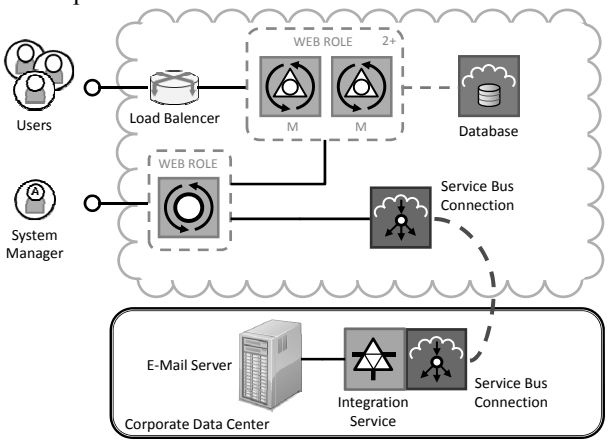



Figure 1: Exemplary scenario modeled for Windows Azure

Consider a simple cloud application with a web-based user interface to access data. This application shall scale elastically with changing workloads. Additionally, if a certain number of resources have been provisioned, a human system manager shall be notified via e-mail. The administrator has to approve further resource provisioning via a management user interface to control the costs generated by the application.

Figure 1 depicts this scenario modeled using the pattern icons for Windows Azure [48]. Users access a so-called *combined web role* hosting the user interface. The term "combined" is used, because in our scenario, this *user interface web role* additionally offers a Web service interface through which its utilization may be obtained. Another web role hosting the management interface accesses this Web service interface to determine an adequate number of user interface web roles, which it provisions. When the number of user interface web roles has reached a threshold, the *management interface web role* accesses a corporate e-mail server to inform a system manager. Since there is currently no e-mail service directly available in Windows Azure, this e-mail is sent through a *service bus connection* to a local data center where the corporate email server is hosted (currently, there is no icon for such servers).

In Figure 2, the exemplary scenario has been modeled with graphical icons provided for Amazon AWS [46]. Whereas Amazon does not explicitly use the term 'pattern' to refer to these icons, their use is very similar to the patterns provided for Windows Azure. In our motivating example, users access a set of EC2 instances (virtual servers) [8], which host the user interface, through a load balancer. The EC2 instances access Amazon's SimpleDB [8] to access provided data. The

EC2 instances are further monitored by CloudWatch [8]. Using this monitoring service, an alarm can be defined using an Amazon-specific rule language. The rule states that the system administrator shall be notified via the e-mail functionality provided by the Amazon Simple Notification Service (SNS) [8] when a certain number of EC2 instances have been started. The administrator can then access Amazon's Auto Scaling [8] service to adjust the maximum number of allowed EC2 instances.

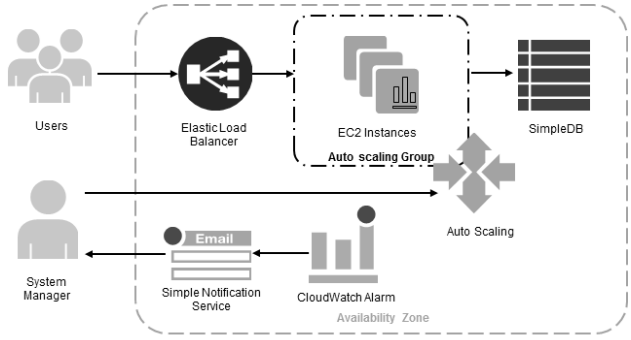



Figure 2: Exemplary scenario modeled for Amazon AWS

Both scenarios describe the same architectural concept: they tightly integrate human management interaction into the automated management of the application to ensure that the dynamicity of the cloud is not hindered by lengthy offline management tasks. We argue that this inherent architectural concept appears very different depending on the cloud provider for which it is expressed. This is due to a number of limitations of the provider-specific models. First, they always consider *concrete services* of providers, but the available functionality may differ. Second, providers employ *different level of abstraction* regarding the cloud service model they offer, i.e., *IaaS*, *PaaS*, or *SaaS*. We see this as an important motivation to provide a common, abstract pattern format to capture common architectural concepts abstracting from the models and implementations of different cloud providers.

## III. IDENTIFICATION OF PATTERNS

Based on existing pattern descriptions from other domains, cloud patterns we identified earlier [4], and a pattern format reviewed and improved through the pattern community [3], we have compiled a pattern identification process, depicted in Figure 3. It formalizes the pattern identification we followed in our work and assists experts to identify, capture and distill implicit knowledge into reusable cloud patterns. The step-by-step process ensures traceability of the identification efforts, and incorporates a continuous evolution and improvement of discovered patterns. We support this process with a *pattern authoring toolkit*[1]. It includes an Excel template for capturing information sources, a Word template containing the pattern format, and a PowerPoint template guiding the creation of pattern icons and sketches by providing a set of reusable shapes.

[1] http://cloudcomputingpatterns.org/authoringtoolkit.zip

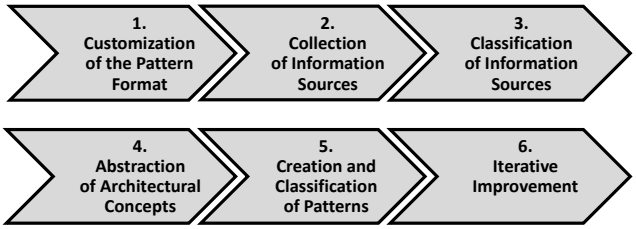



Figure 3: Overview of the Pattern Identification Process

### A. Step 1: Customization of the Pattern Format

Based on available works on patterns and the formats used for describing them, we defined the pattern format contained as a Word template in the pattern authoring toolkit. This format allows the omission of optional sections, which are then summarized with mandatory sections as indicated in the Word template. Further customization, for example, the inclusion of video material, code samples etc. is incorporated as annotations. This enforces a common format to ease adoption and comprehension while allowing arbitrary customization for a specific use case.

Each pattern is identified by a unique *name* and has an *icon* associated with it. This icon is used as graphical representation for the pattern in architectural diagrams. Then, a *driving question* states the problem solved by the pattern to enable readers to quickly check if the pattern fits to problems at hand. The following *context* section describes the environment in which the pattern may be used, going into greater detail. In addition to a textual description, other patterns may be referenced here that form the environment. In this scope, the expression of cloud types and cloud offerings in the pattern format [4] has proven to be very valuable. The context section is followed by stating the explicit *problem* solved by the pattern. This section is optional, because the driving question and the context section may already be sufficient. Then, a brief *solution* section provides the pattern reader with a list of steps to follow. It is supported by a graphical *sketch* depicting the architectural elements employed by the pattern. Especially, this sketch may contain icons of other patterns. The outcome after the application of the pattern is described in detail in the *result* section. In this section, new challenges may be covered that arise in the resulting environment and, in doing so, patterns may be referenced addressing these challenges. In the optional *variations* section different styles to apply the pattern may be described, if the differences between the variants do not justify their description as individual patterns. Additional *references* to other patterns than those referenced by the context or result section are covered next. Especially, the following references used in many pattern formats [9] [10] [2] should be covered: *composition* – the pattern uses another pattern; *refinement* – the pattern provides a more detailed version of another more abstract pattern; *generalization* – the pattern abstracts from a more concrete pattern, which is especially useful to refer to provider-specific models if they implement the pattern, for example, like those of the exemplary scenario described in Section II. *Known uses* of the pattern are provided to give practical examples in which the pattern has been applied, as well as

pointers to information sources from which it was abstracted. Finally, a section of *annotations* may be used to point to arbitrary documents, code samples, videos etc. supporting the pattern reader during the comprehension and implementation of the pattern.

### B. Step 2: Collection of Information Sources

In this step, different information sources are collected. To support this step, we provide an Excel spreadsheet contained in the pattern authoring toolkit. The toolkit further contains example spreadsheets filled with the information sources used to extract the patterns presented in this paper. For each information source, general information, such as document names, hyperlinks etc. are collected. Further, the cloud computing relevant information obtained from the sources is summarized. This summarization should be expressed respecting the context of the information source and its domain specific terms. There should *not* be any interpretation or abstraction in this step to make the pattern that is abstracted from this information source traceable to its unaltered origin. Examples for information sources are provider guidelines, knowledge about existing applications, books, and journals.

### C. Step 3: Classification of Information Sources

The summarized knowledge extracted from the information sources, is classified regarding the architecture domain and challenge it addresses. The pattern authoring toolkit includes the domains of existing patterns [4] and suggests challenges, for example, availability, automation, scalability, pay-per-use, or tenant-isolation. During the classification process the summaries created for information sources may be split up into multiple entries to allow a fine-granular classification. For example, Amazon EC2 gives the guideline [1] [11] to divide application functionality among distributed virtual machines and availability zones, addressing the architectural challenge of availability.

### D. Step 4: Abstraction of Architectural Concepts

Until now, the summarized knowledge extracted from the various information sources remains provider-specific or application-specific. It, therefore, needs to be abstracted to inherent architectural concepts. During this step, summaries may again be split up. For example, the above mentioned redundancy of virtual machines is also suggested by IBM for WebSphere Clusters [33]. Both sources can be abstracted to the concept of 'distribution of application components among different and redundant cloud resources'. In the corresponding column of the Excel sheet, abstract concepts are reused as much as possible. This allows a grouping of summarized knowledge regarding the abstracted concepts using Excel's filtering functionality.

### E. Step 5: Creation and Classification of Patterns

Based on the abstract concepts, patterns are now compiled from the information sources. Since patterns are not invented, but identified and discovered from existing solutions, ensuring a suitable level of abstraction is always a challenge. For this task, the pattern writer reviews information sources with the same abstracted architectural concept addressing the same architectural challenge. While pattern writing itself cannot be automated, we argue that the prior classification and abstraction of information sources eases this step. Further, it ensures an adequate level of abstraction of the created patterns by filtering product-specific information.

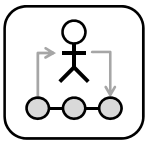
### F. Step 6: Iterative Improvement

Our industry collaboration and discussions with pattern experts (see acknowledgements) have shown that patterns evolve continuously through discussion in a community. This community should be heterogeneous, thus, should contain experts and beginners alike to ensure that obtained patterns contain all necessary information while remaining comprehensible. As patterns are created by many different participants, an alignment of graphical elements can be pursued during the revision for a more a consistent look-and-feel. For example, an icon to depict a virtual machine should look the same or at least similar in different pattern catalogs. According to Moody [12], one of the key success factors of a graphical notation is 'cognitive effectiveness', which he refers to as the speed, ease and accuracy with which humans read diagrams of a notation. Moody describes many further aspects of good notations, for example, he argues that cognitive and perceptual limits need to be respected for diagram complexity management. Therefore, during iterative improvement, graphical designers are involved to obtain easy to understand graphics. This has been proven useful, as exemplified for learning patterns described by [13].

## IV. Exemplary Patterns

We provide two exemplary patterns from different domains that we identified using the pattern authoring toolkit. The information sources for these patterns are included in the pattern authoring toolkit. The following pattern descriptions are based on the pattern format of the Word template also included in the pattern authoring toolkit.

### A. Human Integration in Automated Systems Management



*How can the dynamicity and flexibility of the cloud be ensured even if IT management tasks have to be performed by humans?*

**Context**: hardware virtualization and the standardization of runtime platforms have enabled very flexible provisioning capabilities of clouds. New resources like virtual servers can be started and stopped within minutes. However, a company may have to handle many additional tasks associated with this provisioning and deprovisioning of cloud resources, such as obtaining management approvals, budgeting, documentation etc. If these tasks are not sped up in a similar fashion as the resource provisioning and deprovisioning, the beneficial effects of cloud computing are reduced drastically.

**Solution**: human interaction is reduced as much as possible. If human interaction is required, it is integrated in automated management flows via various communication channels. The pattern sketch is depicted in Figure 4.

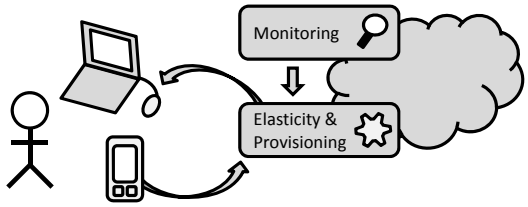


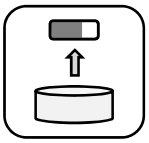

Figure 4: Sketch of the Systems Management with Human Integration

**Result**: to take advantage of cloud computing, a company has to minimize the human interactions required in its IT management processes. For example, budget approval tasks should not consider single servers, but a certain amount of money for which arbitrary offerings of a cloud provider may be obtained. Approving every start of a new virtual machine in the cloud would be inefficient. If human interactions are required, for example, to provide critical encryption keys, administrators and business personnel should be integrated into automated management processes. Depending on the importance of the task at hand, different interaction styles and communication channels should be used. For example, informational messages about a cloud application may be propagated via RSS or Twitter. If a critical action is required, a system administrator may provide commands via a text message from a cell phone.

**Relations to other patterns**: in [52], we describe different interaction styles how humans may be integrated in management tasks. For example, 'user notification' describes how informational data can be provided to humans. 'User response required' describes how an input may be obtained from humans and optionally escalates the communication channels the longer the system has to wait for input.

**Known uses**: Amazon CloudWatch [8] allows the descriptions of systems states to inform a systems manager via e-mail, text messages etc.

### *B. Eventually-consistent User Interface*



*How can the user interface of an application cope with eventual consistency of used data stores?*

**Context**: eventual consistency of cloud storage offerings is of vital importance for the availability and performance of a cloud application [14]. By allowing inconsistencies between data replicas, the performance and availability i.e., tolerance towards network partitioning, may be enabled [15]. However, this affects the state information that an application can provide, which is problematic if the business case expects consistent data.

**Solution**: design the functions supporting the business case so that they do not depend on knowledge about the consistent overall system state. Therefore, provide data abstractions, approximations, and tendencies whenever possible. The pattern sketch is depicted in Figure 5.

**Result**: traditionally, many business cases assumed consistent information to be handled by application. However, in many cases this quality of information is unnecessary.

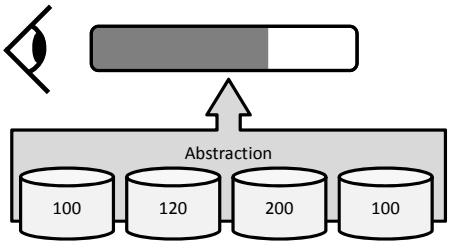



Figure 5: Sketch of the Eventually-consistent User Interface Pattern

Instead, users may be provided with approximations or abstractions, for example, in the form of progress bars, traffic lights, or other categories.

**Relations to other patterns**: if the business case is designed to incorporate and support the application of the *eventual-consistency* pattern [4], data stores implementing this pattern, for example *No-SQL storage* [4], may be employed in the application.

**Known Uses**: SFPark [22] provides information about parking space availability in San Francisco. Since timely information is not always available and hard to retrieve from garage management systems, availability is only described as *high*, *medium*, or *low*. WallMart's online store [23] abstracts the availability of products to *in stock*, *limited quantity*, and *out of stock*. In this case, the abstraction is better suited for the business case: if absolute numbers were provided to customer, he or she would need to know how many items were sold in the past in order to interpret them.

## V. ARCHITECTURE DOMAINS RELEVANT FOR CLOUD COMPUTING

Cloud applications have to follow three fundamental architectural principles: *componentization* – the functionality of a cloud application is divided into loosely coupled components; *distribution and redundancy* – multiple instances of application components are distributed among a set of redundant cloud resources; *automated management* – resource provisioning and deprovisioning are automated to increase beneficial effects of the cloud's elasticity. Resiliency towards failing resources is ensured by handling resource failures automatically. These architectural concepts have already been captured as cloud patterns [4], but there are still undiscovered architectural concepts out there addressing the challenges of cloud computing. Given the pattern identification process described in Section III, we now present a collection of challenging domains we consider qualified for cloud pattern capturing. We identified the relevance of these domains based on various information sources: (i) literature on enterprise and software architectures and middleware [10] [16] [17], (ii) existing cloud applications [18] [19] [20] and (iii) internal use cases we worked on in our industry collaboration. The classification of challenges into the following domains is based on this information and is presented in alphabetical order. We do not claim that this is a complete list, but we argue that each domain has problems with corresponding good solutions to be discovered and captured for the advancement of cloud technology.

### A. Accounting & Controlling

The use of pay-per-use billing is inherent to cloud computing. Additionally, providers may offer long term reservation of resources at a lower price, such as Amazon Reserved Instances [8]. These are intended for the static workload of customers. Variable pricing depending on the overall utilization of a cloud are also available, such as Amazon Spot Instances [8]. These billing models should be reflected in the application architecture to reduce runtime costs. Companies will also be faced with accounting challenges to keep track of the expenses at different cloud providers and the various payment models used.

### B. Application Migration

Many existing applications could benefit from cloud computing. Especially, business-noncritical applications providing information to the public often have little legal obligations to overcome. However, the environment found in the cloud differs significantly from internal environments. To move applications to the cloud, the applications' infrastructure requirements and how they are respected by the application architecture need to be reconsidered.

### C. Cloud Integration

When using cloud computing, a company often faces the challenge that different computing environments have to be integrated. For example, legacy applications may reside in on-premise data centers, whereas others use a public cloud. A cloud component gateway [36] may be used in this scope to bridge different environments. Other technologies to integrate cloud environments with private data centers or to ensure a consistent name resolution, such as Amazon VPC [8], WSO2 Enterprise Service Bus [21], or Microsoft Azure Connect [47], currently remain provider-specific and have not been abstracted into an abstract re-usable pattern format.

### D. Data Storage

The most influential property of cloud computing regarding handled data is eventual consistency [14]. While inconsistency of data replicas is well-researched in distributed systems [27], cloud computing has shown how this concept may be employed to increase the availability and performance of an application [15] to handle large amounts of data and users efficiently. This behavior of data stores, however, needs to be respected in the data access layer of the application. Best practices, made reusable and easy to grasp as patterns, are required in this scope.

### E. License Management

Licenses issued per CPU [41] executing a software are hard to measure or enforce in the cloud where hardware virtualization is inherent. This renders the use of many software products employing this licensing complicated to impossible in the cloud. Network-based licensing models [41] consider concurrently running software instances. Since virtual servers may be started and stopped dynamically in the cloud, best practices for license management are required to ensure a compliant deployment of cloud resources, for example, during automated scaling processes.

### F. Monitoring, Analysis, and Reporting

Cloud computing is often considered for the aspect of cost savings. The introduced sharing of resources, however, can also be exploited for energy savings. In either case, information about cloud resources has to be collected and analyzed. For example, the use of environmental resources may have to be correlated to products of a company since the public demands to know about the product's environmental footprint [38]. Sharing resources between different projects and products significantly complicates such computations. In scope of this domain, best practices and architectural concepts are required to address these cross-cutting concerns in cloud application monitoring and reporting systems.

### G. Multi-Tenant Cloud Middleware

Sharing cloud resources between multiple cloud users, also called tenants, is a very important concept to benefit from economies of scale [5]. Hardware virtualization introduced this sharing for physical hardware, but beneficial effects increase significantly if middleware is also shared [43]. However, cloud middleware will have to be redesigned to ensure isolation of tenants, for example, on the database layer [5]. Patterns are, therefore, required in this domain to address challenges, such as tenant-isolation, version management, customer-specific application customization, migration of tenants between software versions and runtime environments etc.

### H. Organizational Structures

The introduction of cloud computing to a company may face 'political' challenges. Consuming IT services from a private cloud may feel like a loss of control, which may motivate employees to hinder the adoption of cloud computing. Even though, this domain is not directly application architecture relevant, we argue that cloud patterns for organizational change would be helpful to raise awareness for these organizational challenges as they can be a limiting factor for cloud computing [40]. Similar patterns have been generally defined by [26]. Especially in large companies, provisioning new IT resources or registering new IT services often require human management approval hindering automated management. If the organization structure and processes of a company are not adjusted to respect cloud computing, the beneficial effects can be significantly reduced.

### I. Security and Compliance

Many of the security issues found in non-cloud applications are also arising in a cloud environment. These concepts have been captured in a pattern format [29] and may be applicable in cloud computing as well. New security threats arise from malicious use of dynamic cloud resources or from other cloud users. Regarding compliance, cloud computing introduces the challenge that a company's IT has to compete with public cloud services. Methods how to address these issues have been investigated [49] but there are currently no practical solutions to extract patterns from.

## VI. Related Work

The background knowledge used in this paper is based on key works on patterns and their identification and structuring [28] [9]. The pattern format is largely inspired by Gamma et al. [2] and Holpe and Wolf [10].

Pattern-based descriptions have been used frequently to describe good solutions to reoccurring problems during the architectural design of applications. Patterns for object-oriented programming are given by [2]. In [10], patterns for the message-based integration of enterprise applications are covered. In [29] significant research in the area of security patterns is described. In [31] we described patterns on how IT reconsolidation may reduce the environmental impact of the IT architecture by restructuring application architectures and supported processes. Some of these patterns likely address challenges similar to those faced in cloud computing. However, to be applicable in the context of cloud computing each pattern would have to be evaluated and possibly needs to be revised. In contrast to the cloud pattern format that we propose, many of these patterns do not incorporate pattern icons to be used in architectural diagrams.

In the field of cloud computing, patterns are gaining momentum. In industry, they are often used in a less formal way to present architectural guidelines and advice [50] [51]. In research, patterns have the same intention but are discussed more formally. Binz et al. [24] researched patterns for application migration. Regarding an application moved to the cloud, it covers best practices how to handle differences of the runtime environments in a pattern format. Further, [44] introduces a classification for strategies how to migrate applications to the cloud. Misuse of cloud resources leading to new security threads has been covered by [30]. This misuse ranges from using cloud resources for distributed attacks to hijacking the user accounts of other cloud customers [32]. The two patterns presented in Section IV have been identified during our industry collaboration and from our own experience with cloud applications. Daimler AG has also used patterns previously to homogenize IT landscapes [34] and offers cloud applications for car fleet management [18] [35]. In collaboration with T-Systems we worked on patterns to enable the customization of cloud applications' functionality, data elements, and deployment [36]. The requirements for such patterns came from an application used in the German city of Friedrichshafen to manage the availability and assignment of places in Kindergartens for approximately 59.000 inhabitants [37].

## VII. Conclusion

Current industry-driven approaches to model application architecture diagrams are provider-specific, have different composition semantics, and use different levels of abstraction. We argue that a common pattern format should be used for the abstraction of common underlying principles inherent to multiple cloud providers. We provided a pattern identification process and pattern authoring toolkit to make pattern identification structured and traceable. Challenging domains introduced by cloud computing were covered, where the pattern identification can be applied.

### A. Lessons Learned

**Patterns for Education**: we experienced patterns to be well-perceived as an entry point to cloud computing. The feedback of readers new to the cloud domain was especially useful during step 6 of the pattern identification process (iterative improvement), because we found it difficult for experts to find an adequate level of abstractions during the initial identification and description of patterns.

**Transparency of Pattern Identification**: when presenting patterns to other persons, especially non-members of the pattern research community, we often had to explain how patterns were identified. The presented pattern identification process enables transparency of this task and allows other to retrace the steps undertaken to identify a pattern. We found this increased the credibility of cloud patterns.

**Pattern-based Classification of Providers**: cloud providers are hard to compare, because the implications of services provided in the cloud on application using them are often obfuscated. We found that the cloud patterns enabled a classification of providers regarding the patterns supported by them. Thus, they become easier to compare with each other as well as with existing IT environments.

**Patterns for Documentation**: Generally, patterns can be used in software documentation to reduce its size and increase its quality [42]. We found that the use of patterns in documentation increased readability, because the well-known concept of a pattern could be perceived easier and quicker while reducing the amount of introductory text required in documents. Adoption of the pattern approach in industry confirms this, see motivation in Section II.

**Pattern Vocabulary**: the defined cloud patterns [4] have been used as a common vocabulary in discussions with Daimler AG and T-Systems employees. In this scope, the abstraction from concrete products proved to be very useful to bridge different product user communities.

### B. Outlook

Capturing cloud architectural principles in provider-independent patterns is a first step to overcome the limitations of provider-specific models and notations. In the future, the used graphical representations of patterns are revised to improve readability. A pattern composition language is also required for architecture diagrams with well-defined semantics. During the design of icons and the composition language, cognitive requirements will be considered to ensure 'cognitive effectiveness' [12].

With a growing size of a pattern catalog, advanced tooling is needed to organize patterns and make their relationships explicit in a searchable catalog data model. We started work on pattern recommendation systems in [3]. With a larger catalog, these methods will have to be made more user-friendly. One approach could be the integration with Web 2.0 technologies to enable comments on patterns, ratings, or recommendations. We would like to motivate experts to follow the introduced pattern identification process to also capture solutions to the challenges introduced by cloud computing. Further, the revision and discussion of existing patterns would also be an important step to establishing a knowledge- and expertise-sharing community.

## VIII. ACKNOWLEDGEMENTS

Many thanks go to Robert Hanmer, Gregor Hohpe, Ralph Johnson, and Joseph Yoder for the helpful discussions. The authors further thank all participants and reviewers of the PLoP 2011 for their support and constructive feedback: Celina Gibbs, Sebastian Günther, Ralph Johnson, Donna Kaminskyj Long, Mehdi Mirakhorli, Pedro Monteiro, and Ernst Oberortner. David Schumm would like to thank the German Research Foundation (DFG) for financial support of the project within the Cluster of Excellence in Simulation Technology (EXC 310/1) at the University of Stuttgart.

## REFERENCES

[1] J. Varia: "Cloud Architectures," Technical Report, Amazon, 2010.

[2] E. Gamma, R. Helm, R. Johnson, J. Vlissides, "Design Patterns: Elements of Reusable Object-oriented Software," Addison-Wesley, 1995.

[3] C. Fehling, F. Leymann, R. Retter, D. Schumm, W. Schupeck, "An Architectural Pattern Language of Cloud-based Applications," Proceedings of the Conference on Pattern Languages of Programs (PLoP), 2011.

[4] C. Fehling, F. Leymann, R. Mietzner, W. Schupeck, "A Collection of Patterns for Cloud Types, Cloud Service Models, and Cloud-based Application Architectures," Report No. 2011/05, University of Stuttgart, 2011.

[5] F. Chong, G. Carraro, "Architecture Strategies for Catching the Long Tail", Microsoft Whitepaper, 2006.

[6] Microsoft, Windows Azure. (*http://www.windowsazure.com/)*

[7] U. Zdun, P. Avgeriou, "Modeling Architectural Patterns Using Architectural Primitives," ACM SIGPLAN conference on Object oriented programming systems languages and applications, 2005.

[8] Amazon.com, Amazon Web Services. *(http://aws.amazon.com/)*

[9] G. Meszaros, J. Doble, "A Pattern Language for Pattern Writing," Pattern Languages of Program Design, 1998.

[10] G. Hohpe, B. Wolf, "Enterprise Integration Patterns: Designing, Building, and Deploying," Addison-Wesley, 2004.

[11] Amazon.com, Amazon EC2 Service Level Agreement. *(http://aws.amazon.com/ec2-sla/)*

[12] D. Moody, "The Physics of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering," IEEE Transactions on Software Engineering, 2009.

[13] T. Iba, T. Miyake, M. Naruse, N. Yotsumoto, "Learning Patterns: A Pattern Language for Active Learners," Conference on Pattern Languages of Programs (PLoP), 2009.

[14] W. Vogels. "Eventually Consistent," Communications of the ACM, 2009.

[15] E. A. Brewer, "Towards Robust Distributed Systems," PODC Keynote, 2000.

[16] D. Krafzig, K. Banke, D. Slama, "Enterprise SOA," Prentice Hall, 2004.

[17] M. Fowler, "Patterns of Enterprise Application Architecture," Addison-Wesley, 2003.

[18] Daimler AG, car2go. *(http://www.car2go.com)*

[19] Peecho. Print as a Service. *(http://peecho.com)*

[20] Kununu. Job rating site. *(http://kununu.com)*

[21] WSO2, Enterprise Service Bus. *(http://wso2.com)*

[22] SFMTA, SFpark. *(http://sfpark.org)*

[23] Wallmart, Wallmart Online Shop. *(http://wallmart.com)*

[24] T. Binz, F. Leymann, D. Schumm, "CMotion: A Framework for Migration of Applications into and between Clouds," IEEE International Conference on Service-oriented Computing and Applications (SOCA), 2011.

[25] C. Fehling, F. Leymann, J. Rütschlin, D. Schumm, "Pattern-based Development and Management of Cloud Applications, " *Furture Internet*, 2012. *(http://www.mdpi.com/1999-5903/4/1/110/)*

[26] L. Rising, "Fearless Change: Patterns for Introducing New Ideas: Introducing Patterns to Organizations," Addison-Wesley, 2004.

[27] A.S. Tanenbaum, M. Van Steen, "Distributed Systems: Principles and Paradigms," Prentice-Hall, 2003.

[28] R.C. Martin, "Design Principles and Design Patterns", Object Mentor, 2000.

[29] M. Schumacher, E. B. Fernandez, D. Hybertson, F. Buschmann, P. Sommerlad, "Security Patterns: Integrating Security and Systems Engineering," Wiley, 2006.

[30] K. Hashizume, N. Yoshioka, E.B. Fernandez, "Misuse Patterns for Cloud Computing," Proceedings of the Asian Conference on Pattern Languages of Programs (AsianPLoP), 2011.

[31] A. Nowak, F. Leymann, D. Schleicher, D. Schumm, S. Wagner, "Green Business Process Patterns," Proceedings of the Conference on Pattern Languages of Programs (PLoP), 2011.

[32] J. Somorovsky, M. Heiderich, M. Jensen, J. Schwenk, N. Gruschka, L. Lo Iacono, "All Your Clouds are Belong to us – Security Analysis of Cloud Management Interfaces," Proceedings of the ACM Cloud Computing Security Workshop (CCSW), 2011.

[33] IBM, WebSphere MQ, (http://ibm.com/software/websphere/mq/)

[34] DaimlerChrysler TSS GmbH, "MDA Success Story ePEP successful with Model Driven Architecture," Whitepaper, 2005.

[35] Daimler FleetBoard GmbH, FleedBoard. (*http://www.fleetboard.com*)

[36] C. Fehling, R. Konrad, F. Leymann, R. Mietzner, M. Pauly, D. Schumm, "Flexible Process-based Applications in Hybrid Clouds," IEEE International Conference on Cloud Computing (CLOUD), 2011.

[37] Deutsche Telekom, T-City Friedrichshafen. *(http://www.t-city.de)*

[38] G. Peter, "Carbon Accounting: Beyond The Calculation and Looking To The Future," Green Economy, 2010.

[39] Gartner, "Gartner Identifies the Top 10 Strategic Technologies for 2011," Press Release, 2010.

[40] D. Townsend, "What are the biggest organizational challenges in adopting cloud," ServiceMesh.com, 2011.

[41] D. Ferrante, "Software Licensing Models: What's Out There?," IT Professional, 2006.

[42] A. Rüping, "Agile Documentation," Wiley, 2003.

[43] C. Guo, W. Sun, Y. Huang, Z. Wang, B. Gao, "A Framework for Native Multi-Tenancy Application Development and Management," IEEE CEC/EEE, 2007.

[44] Y. Chee, N. Zhou, F.J. Meng, S. Bagheri, P. Zong, " A Pattern-based approach to Cloud Transformation," IEEE International Conference on Cloud Computing (CLOUD), 2011.

[45] M. Petre, "Why Looking isn't Always Seeing," Communications of the ACM, 1995.

[46] Amazon.com, AWS Simple Icons for Architecture Diagrams. *(http://aws.amazon.com/architecture/icons/)*

[47] Microsoft, Windows Azure Conncect. *(http://www.windowsazure.com/en-us/home/tour/virtual-network/)*

[48] D. Pallmann, "Windows Azure Design Patterns". (*http://www.azuredesignpatterns.com*)

[49] I. Brandic, S. Dustdar, T. Anstett, D. Schumm, F. Leymann, R. Konrad, "Compliant Cloud Computing (C3)" IEEE International Conference on Cloud Computing (CLOUD), 2011.

[50] S. Guest, "Patters for Cloud Computing", 2009.

[51] S. Riley, "How to Think Cloud Architectural Design Patterns for Cloud Computing".

[52] D. Schumm, C. Fehling, D. Karastoyanova, F. Leymann, J. Rütschlin, "Processes for Human Integration in Automated Cloud Application Management," Report No. 2012/02, University of Stuttgart, 2012.