Taxonomy of Deployment Patterns for Cloud-hosted Applications: A Case Study of Global Software Development (GSD) Tools

Laud Charles Ochei, Julian M. Bass, Andrei Petrovski

School of Computing Science and Digital Media
Robert Gordon University
Aberdeen, United Kingdom
Emails: {l.c.ochei,j.m.bass,a.petrovski}@rgu.ac.uk

***Abstract*—Cloud patterns describe deployment and use of various cloud-hosted applications. There is little research which focuses on applying these patterns to cloud-hosted Global Software Development (GSD) tools. As a result, it is difficult to know the applicable deployment patterns, supporting technologies and trade-offs to consider for specific software development processes. This paper presents a taxonomy of deployment patterns for cloud-hosted applications. The taxonomy is composed of 24 sub-categories which were systematically integrated and structured into 8 high-level categories. The taxonomy is applied to a selected set of software tools: JIRA, VersionOne, Hudson, Subversion and Bugzilla. The study confirms that most deployment patterns are related and cannot be fully implemented without being combined with others. The taxonomy revealed that (i) the functionality provided by most deployment patterns can often be accessed through an API or plugin integrated with the GSD tool, and (ii) RESTful web services and messaging are the dominant strategies used by GSD tools to maintain state and exchange information asynchronously, respectively. We also provide recommendations to guide architects in selecting applicable deployment patterns for cloud deployment of GSD tools.**

Keywords–Taxonomy; Deployment Pattern; Cloud-hosted Application; GSD Tool; Plugin.

I. INTRODUCTION

Collaboration tools that support Global Software Development (GSD) processes are increasingly being deployed on the cloud [1][2]. The architectures/patterns used to deploy these tools to the cloud are of great importance to software architects because it determines whether or not the system's required quality attributes (e.g., performance) will be exhibited [3][4][5].

Collections of cloud patterns exist for describing the cloud, and how to deploy and use various cloud offerings [6][7]. However, there is little or no research in applying these patterns to describe the cloud-specific properties of applications in software engineering domain (e.g., collaboration tools for GSD, hereafter referred to as GSD tools) and the trade-offs to consider during cloud deployment. This makes it very challenging to know the deployment patterns (together with the technologies) required for deploying GSD tools to the cloud to support specific software development processes (e.g., continuous integration (CI) of code files with Hudson).

Motivated by this problem, we propose a taxonomy of deployment patterns for cloud-hosted applications to help software architects in selecting applicable deployment patterns for deploying GSD tools to the cloud. We are inspired by the work of Fehling et al. [6], who catalogued a collection of patterns that will help architects to build and manage cloud applications. However, these patterns were not applied to any specific application domain, such as cloud-hosted GSD tools.

The research question this paper addresses is: **"How can we create and use a taxonomy for selecting applicable deployment patterns for cloud deployment of GSD tools."** It is becoming a common practice for distributed enterprises to hire cloud deployment architects or "application deployers" to deploy and manage cloud-hosted GSD tools [8]. Saleforce's Continuous Integration systems, for example, runs 150000 + test in parallel across many servers and if it fails it automatically opens a bug report for developers responsible for that *checkin* [9].

We created and applied the taxonomy against a selected set of GSD tools derived from an empirical study [10] of geographically distributed enterprise software development projects. The overarching result of the study is that most deployment patterns are related and have to be combined with others during implementation, for example, to address hybrid deployment scenarios, which usually involves integrating processes and data in multiple clouds.

The main contributions of this paper are:
1. Creating a novel taxonomy of deployment patterns for cloud-hosted application.
2. Demonstrating the practicality of the taxonomy by applying it to: (i) position a selected set of GSD tools; and (ii) compare the cloud deployment requirements of GSD tools.
3. Presenting recommendations and best practice guidelines for identifying applicable deployment patterns together with the technologies for supporting cloud deployment of GSD tools.

The rest of the paper is organized as follows: Section 2 gives an overview of the basic concepts related to deployment patterns for Cloud-hosted GSD tools. In Section 3, we discuss the research methodology including taxonomy development, tools selection, application and validation. Section 4 presents the findings of the study focusing on positioning a set of GSD tools within the taxonomy. In Section 5, we discuss the lessons learned from applying the taxonomy. The recommendations and limitations of the study are in Sections 6 and 7 respectively. Section 8 is reporting the conclusion and future work.

 ISBN: 978-1-61208-388-9

II. Deployment Patterns for Cloud-hosted GSD Tools

A. Global Software Development

Definition 1: **Global Software Development.** GSD is defined by Lanubile [11] as the splitting of the development of the same software product or service among globally distributed sites. Since there are many stakeholders in GSD, there is need to have tools that support collaboration and integration among the team members involved in software development [12]. The work of Portillo et al. [13] focused on categorizing various tools used for collaboration and coordination in Global Software Development.

Definition 2: **Cloud-hosted GSD Tool.** "Cloud-hosted GSD Tool" refers to collaboration tools used to support GSD processes in a cloud environment. We adopt the: (i) NIST Definition of Cloud Computing to define properties of cloud-hosted GSD tools; and (ii) ISO/IEC 12207 as our classification frame for defining the scope of a GSD tool. Portillo et al. [13] identified three groups of GSD tools for supporting ISO/IEC 12207 processes - tools to support Project Processes (e.g., JIRA), tools to support Implementation Processes such as requirements analysis, integration process (e.g., Hudson) and tools for Support Processes (e.g., Subversion). These GSD tools, also referred to as Collaboration tools for GSD [13], are increasingly being deployed to the cloud for Global Software Development by large distributed enterprises.

B. Cloud Deployment Patterns

Definition 3: **Cloud Deployment Pattern.** We define a "Cloud deployment pattern" as a type of architectural pattern which embodies decisions as to how elements of the cloud application will be assigned to the cloud environment where the application is executed. Architectural and design patterns have long been used to provide known solutions to a number of common problems facing a distributed system [4][14].

Our definition of cloud deployment pattern is similar to the concept of design patterns [14], (architectural) deployment patterns [4], collaboration architectures [3], cloud computing patterns [6], cloud architecture patterns [15], and cloud design patterns [7]. These concepts serve the same purpose in the cloud (as in many other distributed environments). For example, the generic architectural patterns- client-server, peer-to-peer, and hybrid [4] - relates to the following: (i) the 3 main collaboration architectures, i.e., centralized, replicated and hybrid [3]; and (ii) cloud deployment patterns -2-tier, content distribution network and hybrid data [6].

C. Taxonomy of Cloud Computing Patterns

Taxonomies and classifications facilitate systematic structuring of complex information. In software engineering, they are used for comparative studies involving tools and methods, for example, software evolution [16] and Global Software Engineering [17]. In this paper, we focus on using a taxonomy to structure cloud deployment patterns for cloud-hosted applications, in particular in the area of GSD tools.

Several attempts have been made by researchers to create classifications of cloud patterns to build, and deploy cloud-based applications. For example, Wilder [15] describes eleven patterns and then illustrates with the Page of Photos application and Windows Azure how each pattern can be used to build cloud-native applications. A collection of over 75 patterns (with known uses of their implementation) for building and managing a cloud-native application are provided by Fehling et al. [6]. Homer et al. [7] describes 24 patterns, for solving common problems in cloud application development. Moyer [18] also documents a collection of patterns and then uses a simple Weblog application to illustrate the use of these patterns. Other documentation of cloud deployment patterns can be found in [19][20][21][22].

Cloud patterns in existing classifications are applied to simple web-based applications (e.g., Weblog application [18]) without considering the different application processes they support. Moreover, these patterns have not been applied against a set of applications in software engineering domain, such as cloud-hosted GSD tools. GSD tools may have similar architectural structure but they (i) support different software development processes, and (ii) impose varying workload on the cloud infrastructure, which would influence the choice of a deployment pattern. ***Motivated by these shortcomings, we extend the current research by developing a taxonomy of deployment patterns for cloud-hosted applications and applying it to a set of GSD tools.***

III. Methodology

A. Development of the Taxonomy

We develop the taxonomy using a modified form of the approach used by Lilien [23] in his work for building a taxonomy of specialized ad hoc networks and systems for a given target application class. The approach summaries to the following steps:

Step 1: Select the target class of Software Tool- The target class is based on the ISO/IEC 12207 taxonomy for software life cycle process (see Definition 3 for details). The following class of tools are excluded: (i) tools not deployed in a cloud environment (even if they are deployed on a dedicated server to perform the same function); and (ii) general collaboration tools and development environments (e.g., MS Word, Eclipse).

Step 2: Determine the requirements for the Taxonomy- The first requirement is that the taxonomy should incorporate features that restricts it to GSD tools and Cloud Computing. In this case, we adopt the ISO/IEC 12207 framework [13] and NIST cloud computing definition [24]. Secondly, it should capture the components of an (architectural) deployment structure [4] - software elements (i.e., GSD tool to be deployed) and external environment (i.e., cloud environment). Therefore our proposed taxonomy is a combination of two taxonomies - Taxonomy A, which relates to the components of the cloud environment [24], and Taxonomy B which relates to the components of the cloud application architecture [6].

Step 3: Determine and prioritize the set of all acceptable categories and sub-categories of the Taxonomy- Determine and prioritize the set of all acceptable categories and sub-categories of the Taxonomy- We prioritized the categories of the taxonomy to reflect the structure of a cloud stack from physical infrastructure to the software process of the deployed GSD tool. The categories and sub-categories of the 2 taxonomies are described as follows:

(1) Application Process: the sub-categories (i.e., project processes, implementation processes and support processes) represent patterns for handling the workload imposed on the cloud infrastructure by the ISO/IEC 12207 software processes supported by GSD tools [13]; *(2) Core cloud properties:* the

 ISBN: 978-1-61208-388-9

sub-categories (i.e., rapid elasticity, resource pooling and measured service) contain patterns used to mitigate the core cloud computing properties of the GSD tools [6]; *(3) Service Model:* the sub-categories reflect cloud service models- SaaS, PaaS, IaaS [24]; *(4) Deployment Model:* the sub-categories reflect cloud deployment models- private, community, public and hybrid [24]; *(5) Application Architecture:* the sub-categories represent the architectural components that support a cloud-application such as application components (e.g., presentation, processing, and data access), multitenancy, and integration [6]; *(6) Cloud Offerings:* the sub-categories reflect the major infrastructure cloud offerings that can be accessed- cloud environment, processing, storage and communication offering [6]; *(7) Cloud Management:* contains patterns used to manage both the components and processes/runtime challenges) of GSD tools. The 2 sub-categories are - management components and processes [6]; *(8) Composite Cloud:* contains compound pattern (i.e., patterns that can be formed by combining other patterns or can be decomposed into separate components). The sub-categories are: decomposition style and hybrid cloud application [6].

Step 4: Determine the space of the Taxonomy- The selected categories and their associated sub-categories define the space of the taxonomy. Table 1 show the taxonomy captured in one piece. The upper-half represents Taxonomy A which is based on NIST Cloud Computing Definition, while the lower-half represents Taxonomy B which is based on the components of a typical cloud application architecture. Each Taxonomy, A and B, has four categories, each with a set of sub-categories. Entries in the "Related Pattern" column show examples of patterns drawn from well-known collections of cloud patterns such as [6][15][7]. The thick lines (Table I) show the space occupied by patterns used for hybrid-deployment scenarios.

TABLE I. TAXONOMY OF DEPLOYMENT PATTERNS FOR CLOUD-HOSTED APPLICATIONS

Deployment Components	**Categories of Deployment Patterns**		**Related Patterns**
	Main Categories	*Sub-Categories*	
Cloud-hosted Environment (Taxonomy A)	Application Process	Project processes	Static workload
		Implementation processes	Continuously changing workload
		Support processes	Continuously changing workload
	Core Cloud Properties	Rapid Elasticity	Elastic platform,Autoscaling[15]
		Resource Pooling	Shared component, Private cloud
		Measured Service	Elastic Platform, Throttling[7]
	Cloud Service Model	Software resources	SaaS
		Platform resources	PaaS
		Infrastructure resources	IaaS
	Cloud Deployment Model	Private clouds	Private cloud
		Community clouds	Community cloud
		Public clouds	Public cloud
		Hybrid clouds	Hybrid cloud
	Composite Cloud Application	Hybrid cloud applications	Hybrid Processing, Hybrid Data,Multisite Deployment [15]
Cloud-hosted Application (Taxonomy B)		Decomposition style	2-tier/3-tier application, Content Delivery Network [15]
	Cloud Management	Management Processes	Update Transition Process, Scheduler Agent [7]
		Management Components	Elastic Manager, Provider Adapter, External Configuration Store [7]
	Cloud Offerings	Communication Offering	Virtual Networking, Message-Oriented Middleware
		Storage Offering	Block Storage, Database Sharding [15], Valet Key [7]
		Processing Offerings	Hypervisor, Map Reduce [15]
		Cloud Environment Offerings	Elastic Infrastructure, Elastic Platform, Runtime Reconfiguration [7]
	Cloud Application Architecture	Integration	Integration Provider, Restricted Data Access Component
		Multi-tenancy	Shared Component, Tenant-Isolated Component
		Application components	Stateless Component, User Interface Component

TABLE II. PARTICIPATING COMPANIES, SOFTWARE PROJECTS, SOFTWARE-SPECIFIC PROCESS AND GSD TOOLS USED

Companies	Projects	Software process	GSD tool
Company A, Bangalore	Web Mail Web Calendar	Issue tracking Code integration	JIRA Hudson
Company B, Bangalore	Web Mail Web Calendar	Issue tracking Version control	JIRA Subversion
Company H, Delhi	Customer service Airline	Agile tailoring Issue tracking	VersionOne JIRA
Company D, Bangalore (Offshore Provider to Company E)	Marketing CRM	version control Error tracking	Subversion Bugzilla
Company E, London	Banking Marketing CRM	Issue tracking Agile tailoring Code Building	JIRA VersionOne Hudson

B. GSD Tool Selection

We carried out an empirical study to find out: (1) the type of GSD tools used in large-scale distributed enterprise software development projects; and (2) what tasks they utilize the GSD tools for.

1) Research Site: The study involved 8 international companies and interviews were conducted with 46 practitioners. The study was conducted between January, 2010 and May, 2012; and then updated between December, 2013 and April, 2014. The companies were selected from a population of large enterprises involved in both on-shore and off-shore software development projects. The companies had head offices in countries spread across three continents: Europe (UK), Asia (India), and North America (USA). Data collection involved document examination/reviews, site visits, and interviews. Further details of the data collection and data analysis procedure used in the empirical study can be seen in Bass [10].

2) Derived Dataset of GSD Tools: The selected set of GSD tools are: JIRA [25], VersionOne [26], Hudson [27], Subversion [28] and Bugzilla [29]. We selected these tools for two main reasons: (i) Practitioners confirmed the use of these tools in large scale geographically distributed enterprise software development projects [10]; (ii) The tools represents a mixture of open-source and commercial tools that support different software development processes; and are associated with stable developers community (e.g., Mozilla Foundation) and publicly available records (e.g., developer's websites, whitepapers, manuals). Table 2 (another view of the one in [10]) shows the participating companies, projects and the GSD tools they used.

C. Applying the Taxonomy

We demonstrate the practicality of the taxonomy by applying it to position a selected set of GSD tools. We used the collection of patterns from [6] as our reference point, and then complimented the process with patterns from [15][7].

The structure of the positioned deployment pattern, in its textual form, is specified as a string consisting of three sections-(i) Applicable deployment patterns; (ii) Technologies required to support such implementation; and (iii) Known uses of how the GSD tool (or one of its products) implements or supports the implementation of the pattern. In a more general sense, the string can be represented as: [PATTERN; TECHNOLOGY; KNOWN USE]. When more than one pattern or technology is applicable, we separate them with commas. Each sub-category of the taxonomy represents a unique class of reoccurring cloud deployment problem, while the applicable deployment pattern represents the solution.

 ISBN: 978-1-61208-388-9

D. Validation of the Taxonomy

We validate the taxonomy in theory by adopting the approach used by Smite et al. [17] to validate his proposed taxonomy for terminologies in global software engineering. A taxonomy can be validated with respect to completeness by benchmarking against existing classifications and demonstrating its utility to classify existing knowledge [17].

We have benchmarked Taxonomy A to existing classifications: the ISO/IEC 12207 taxonomy of software life cycle processes and the components of a cloud model based on NIST cloud computing definition, NIST SP 800-145. Taxonomy B is benchmarked to components of a cloud application architecture such as cloud offering and cloud management, as proposed by Fehling et al. [6]. The collection of patterns in [6] captures all the major components/processes required to support a typical cloud-based application, such as cloud management and integration.

We demonstrate the utility of our taxonomy by positioning the 5 selected GSD tools within the taxonomy to evaluate applicable deployment patterns together with the supporting technologies for deploying GSD tools to the cloud. Table 3 and 4 show that several deployment patterns (selected from 4 studies) can be placed in the sub-categories of our taxonomy.

IV. FINDINGS

In this section, we present the findings obtained by applying the taxonomy against a selected set of GSD tools: JIRA, VersionOne, Hudson, Subversion and Bugzilla. Refer to section III- B for details of the processes supported by these tools.

A. Comparing the two Taxonomies

The cloud deployment patterns featured in Taxonomy A (i.e., upper part of Table 1) relates to the cloud environment hosting the application, while the cloud deployment patterns in Taxonomy B (i.e., lower part of Table 1) relates to the cloud-hosted application itself. For example, the PaaS pattern is used to provide an execution environment to customers on the provider-supplied cloud environment. Elastic platform pattern can be used in the form of a middleware integrated into a cloud-hosted application to provide an execution environment.

B. Hybrid-related deployment Patterns

Both taxonomies contain patterns for addressing hybrid deployment scenarios (i.e., the space demarcated with thick lines). For example, a hybrid cloud (Taxonomy A) integrates different clouds and static data centers to form a homogeneous hosting environment, while hybrid data (Taxonomy B) can be used in a scenario where data of varying sizes generated from a GSD tool resides in an elastic cloud and the remainder of the application resides in a static environment.

C. Patterns for Implementing Elasticity

We have observed patterns that can be used by GSD tools to address rapid elasticity at all levels of the cloud stack. For example, Elastic manager can be used at the application level. Elastic platform and Elastic infrastructure can be used at the platform, and Infrastructure resources level, respectively.

D. Positioning of GSD tools on the Taxonomy

Tables 2 and 3 show the findings obtained by positioning the cloud-hosted GSD tools on each sub-category of the taxonomy. In the following, we present a shortlist of these findings to show that we can identify applicable deployment patterns to address a wide variety of deployment problems.

(i) All the GSD tools considered in this study are based on web-based architecture. For example, Bugzilla and JIRA are designed as a web-based application, which allows for separation of the user interface, and processing layers from the database that stores details of bugs/issues being tracked.

(ii) All the GSD tools have support for API/Plugin architecture. For example, JIRA supports several API's that allows it to be integrated with other GSD tools. Bugzilla:Web services, a standard API for external programs to interact with Bugzilla implies support for stateless pattern. These APIs represent known uses how these deployment patterns are implemented.

(iii) Virtualization is a key supporting technology used in combination with other patterns to achieve elasticity at all levels of the cloud stack, particularly in ensuring fast provisioning and de-provisioning of infrastructure resources.

(iv) The GSD tools use Web services (through a REST API in patterns such as integration provider [6]) to hold external state information, while messaging technology (through message queues in patterns such as Queue-centric workflow [15] and Queue-based load leveling [7]) is used to exchange information asynchronously between GSD tools/components.

(vi) Newer commercial GSD tools (JIRA and VersionOne) are directly offered as SaaS on the public cloud. On the other hand, older open-source GSD tools (Hudson, Subversion and Bugzilla) are the preferred for private cloud deployment. They are also available on the public cloud, but by third party cloud providers.

We summarize our findings as follows: Although there are a few patterns that are mutually exclusive (e.g., stateless and statefull components [6]), most patterns still have to be combined with others (e.g, combining PaaS with Elastic platform). These deployment patterns may also use similar technologies such as REST, messaging and virtualization to facilitate their implementation.

V. DISCUSSION

The findings clearly suggest that by positioning a set of GSD tools on our proposed taxonomy, the purpose of the study has been achieved. The overarching result of the study is that most deployment patterns have to be combined with others during implementation. The findings presented here support previous research suggesting that most patterns are related and so two or more patterns can be used together [4][14].

A. Combining Related Deployment Patterns

Many deployment patterns are related and cannot be fully implemented without being combined with other ones, especially to address hybrid deployment scenarios. This scenario is very common in collaborative GSD projects, where a GSD

 ISBN: 978-1-61208-388-9

TABLE III. POSITIONING GSD TOOLS ON THE PROPOSED TAXONOMY (TAXONOMY A)

Category	Sub-Category	*JIRA*	*VersionOne*	*Hudson*	*Subversion*	*Bugzilla*
Application Process	Project processes	**Static workload, Continuously changing workload**; *SaaS*; JIRA used by small no. of users, issues tracked reduces over time[25]	**Static workload**; *SaaS*; VersionOne is installed for a small number of users[26]	Process not supported	Process not supported	Process not supported
	Implementation processes	Process not supported	Process not supported	**Continuously changing workload**; *PaaS*; Hudson builds reduces gradually as project stabilizes)[27]	Process not supported	Process not supported
	Support processes	Process not supported	Process not supported	Process not supported	**Static workload, Continuously changing workload**;*PaaS, Hypervisor*; rate of code files checked into Subversion repository is nearly constant or reduces over time[28]	**Continuously changing workload**; *PaaS,Hypervisor*; Errors tracked using Bugzilla reduces over time[29]
Core Cloud Properties	Rapid Elasticity	**Stateless pattern, Elastic platform**; *REST API*; JIRA is installed in cloud as SaaS[25]	**Stateless pattern, Elastic platform**; *REST API*; VersionOne is installed in cloud as SaaS[26]	**Elastic infrastructure, shared component**; *hypervisor*; Hudson server is supported by hypervisor in a private cloud[27]	**Elastic infrastructure, tenant-isolation component**; *hypervisor*; Subversion repository is supported by Elastic infrastructure[28]	**Stateless pattern**; *REST API*; Bugzilla is installed in cloud as SaaS in private cloud[29]
	Resource Pooling	**Hypervisor, Public Cloud,** ; *Virtualization*; JIRA deployed on the public cloud as SaaS[25]	**Hypervisor, Public cloud**; *Virtualization*; VersionOne deployed on public cloud as SaaS[26]	**Hypervisor, Tenant-isolated component**; *Virtualization*; Hudson is deployed on a hypervisor[27]	**Hypervisor, Tenant-isolated component**; *Virtualization*; Subversion is deployed on a hypervisor[25]	**Hypervisor, Public cloud**; *Virtualization*; Bugzilla deployed on the public cloud[29]
	Measured Service	**Static workload, Elastic Infrastructure,Throttling[7]**; *Virtualization*; Small number JIRA users generates a nearly constant workload[25]	**Static workload, Elastic Infrastructure,Throttling[7]**; *Virtualization*; Small number of VersionOne users generates small workload[26]	**Static workload, Elastic Infrastructure,Throttling[7]**; *Virtualization*; Hudson can be supported on public cloud by elastic infrastructure[27]	**Static workload, Elastic Infrastructure,Throttling[7]**; *Virtualization*; Subversion can be supported on public cloud by elastic infrastructure[28]	**Static workload, Elastic Infrastructure,Throttling[7]**; *Virtualization*; Bugzilla can be supported on third party public cloud by elastic infrastructure[29]
Cloud Service Model	Software resources	**SaaS**; *Web Services, REST*; JIRA OnDemand[25]	**SaaS**; *Web Services, REST*; VersionOne OnDemand[26]	**SaaS**; *Web Services, REST*; Hudson is offered by 3^{rd} party cloud providers like CollabNet[30]	**SaaS**; *Web Services, REST*; Subversion is offered by 3^{rd} party cloud providers like CollabNet[30]	**SaaS**; *Web Services, REST*; Bugzilla is offered by 3^{rd} party cloud providers like CollabNet[30]
	Platform resources	**PaaS**; *Elastic platform, Message Queuing*; JIRA Elastic Bamboo[25]	**PaaS**; *Elastic platform, Message Queuing*; No known use	**PaaS**; *Elastic platform, Message Queuing*; Build Doctor and Amazon EC2 for Hudson	**PaaS**; *Elastic platform, Message Queuing*; Flow Engine powered by Jelastic for Subversion	**PaaS**; *Elastic platform, Message Queuing*; No known use
	Infrastructure resources	Not applicable	Not applicable	**IaaS**; *Hypervisor*; Hudson is a distributed execution system comprising master/slave servers[27]	**IaaS**; *Hypervisor*; Subversion can be deployed on a hypervisor	Not applicable
Cloud Deployment Model	Private usage	**Private cloud**; *Hypervisor*; JIRA can be deployed on private cloud using private cloud software like OpenStack	**Private cloud**; *Hypervisor*; VersionOne On-premises[26]	**Private cloud**; *Hypervisor*; Hudson can be deployed on private cloud using private cloud software	**Private cloud**; *Hypervisor*; Subversion can be deployed on private cloud using private cloud software	**Private cloud**; *Hypervisor*; Bugzilla can be deployed on private cloud using private cloud software
	Community usage	**Community cloud**; *SaaS*; Bugzilla can be deployed on private cloud	**Community cloud**; *SaaS*; Bugzilla can be deployed on community cloud	**Community cloud**; *SaaS,Paas, IaaS*; Bugzilla can be deployed on community cloud	**Community cloud**; *SaaS,IaaS*; Bugzilla can be deployed on community cloud	**Community cloud**; *SaaS, PaaS*; Bugzilla can be deployed on community cloud
	Public usage	**Public cloud**; *SaaS*; JIRA OnDemand is hosted on public cloud[25]	**Public cloud**; *SaaS*; VersionOne is hosted on public cloud[26]	**Public cloud**; *SaaS,PaaS,IaaS*; Hudson is hosted on public cloud(via 3^{rd} party providers)[30]	**Public cloud**; *SaaS, IaaS*; Subversion is hosted on public cloud(via 3^{rd} party providers)[30]	**Public cloud**; *SaaS, PaaS*; Bugzilla is hosted on public cloud(via 3^{rd} party providers)[30]
	Hybrid usage	**Hybrid cloud**; *SaaS*; JIRA used to track issues on multiple clouds	**Hybrid cloud**; *SaaS*; Agile projects are stored in different clouds[28]	**Hybrid cloud**; *SaaS,PaaS, IaaS*; Hudson builds done in separate cloud	**Hybrid cloud**; *SaaS, IaaS*; Subversion repository resides in multiple clouds	**Hybrid cloud**; *SaaS, PaaS*;Bugzilla DB can be stored in different clouds

tool either requires multiple cloud deployment environments or components, each with its own set of requirements. Our taxonomy, unlike others [15][7], clearly shows where to look for hybrid-related deployment patterns (i.e., the space demarcated by thick lines in Table I) to address this challenge. For example, when using Hudson there is usually a need to periodically extract the data its generates to store in an external storage during continuous integration of files. This implies the implementation of a hybrid data pattern. Hudson can be used in combination with other GSD tools such as Subversion (for version control) and Bugzilla (for error tracking) within a particular software development project, each of which may also have their own deployment requirements.

B. GSD Tool Comparison

The taxonomy gives us a better understanding of various GSD tools and their cloud specific features. While other taxonomies and classifications use simple web applications [15] to exemplify their patterns, we use a mixture of commercial and open-source GSD tools. For example, commercial GSD tools (i.e., JIRA and VersionOne) are offered as a SaaS on the public cloud and also have a better chance of reflecting the essential cloud characteristic. Their development almost coincides with the emergence of cloud computing, allowing new features to be introduced into revised versions. The downside is that they offer less flexibility in terms of customization [31]. On the other hand, open-source GSD tools (i.e., Hudson, Subversion) are provided on the public cloud by third party providers and they rely on API/plugins to incorporate support for most cloud features. The downside is that many of the plugins available for integration are not maintained by the developer's community and so consumers use them at their own risk. The taxonomy also revealed that open-source GSD tools (e.g., Hudson, Subversion) are used at a later stage of a software life-cycle process in contrast to commercial tools which are used at the early stages.

C. Supporting Technologies and API/Plugin Architecture

Another interesting feature of our taxonomy is that by positioning the selected GSD tools on it, we discovered that the

 ISBN: 978-1-61208-388-9

TABLE IV. POSITIONING GSD TOOLS ON THE PROPOSED TAXONOMY (TAXONOMY B).

Category	Sub-Category	*JIRA*	*VersionOne*	*Hudson*	*Subversion*	*Bugzilla*
Application Architecture	Application Components	**User interface component,Stateless**; *REST API, AJAX*; State information in JIRA thru REST API[25]	**User-interface component,Stateless**; *jQuery AJAX, REST/Web Service*; VersionOne REST API[26]	**User-interface component,Stateless**; *REST API, AJAX*; Hudson Dashboard pages via REST[27]	**User-interface component,Stateless**;*REST API, AJAX*; ReSTful Web Services used to interact with Subversion Repositories [28]	**Stateless**; *Bugzilla:WebService API*; Bugzilla::WebService API[29]
	Multitenancy	**Shared component**; *Elastic Platform, Hypervisor*; JIRA login system[25]	**Shared component**; *Hypervisor*; VersionOne supports re-useable configuration schemes[26]	**Shared component**; *Hypervisor*; Hudson 3.2.0 supports multi-tenancy with Job Group View and Slave isolation[27]	**Tenant Isolated component**; *Hypervisor*; Global search/replace operations are shielded from corrupting subversion repository.[28]	**Shared component**; *Hypervisor*; Different users are virtually isolated within Bugzilla DB[29]
	Cloud Integration	**Restricted Data Access component, Integration provider**; *REST API*; JIRA REST API is used to integrate JIRA with other applications[25]	**Integration provider**; *REST, Web Services*; VersionOne OpenAgile Integrations platform, REST Data API for user stories[26]	**Integration provider**; *REST, Web Services*; Stapler component of Hudson's architecture uses REST[27]	**Integration provider**; *REST, Web Services*; Subversion API[28]	**Integration provider**; *REST, Web Services*; Bugzilla::WebService API[29]
Cloud Offering	Cloud environment Offering	**Elastic platform**; *PaaS*; JIRA Elastic Bamboo runs builds to create instances of remote agents in the Amazon EC2[25]	**Integration provider**; *REST, Web Services*; Versionone's Project Management tools are used with TestComplete for automated testing environment [26]	**Elastic Infrastructure/Platform, Node-based Availability**; *PaaS, IaaS*; Hudson is a distributed build platform with "master/slave" configuration [27]	**Elastic platform**; *PaaS*; Subversion repository can be accessed by a self-service interface hosted on a shared middleware	**Elastic Platform**; *PaaS*; Bugzilla s hosted on a middleware offered by providers[29]
	Processing Offering	**Hypervisor**; *Virtualization*; JIRA is deployed on virtualized hardware	**Hypervisor**; *Virtualization*; VersionOne can be deployed on virtualized hardware	**Hypervisor**; *Virtualization*; Hudson is deployed on virtualized hardware	**Hypervisor**; *Virtualization*; Subversion is deployed on virtualized hardware	**Hypervisor**; *Virtualization*; Bugzilla is deployed on virtualized hardware
	Storage Offering	**Block**; *Virtualization*; Elastic Bamboo can access centralized block storage thru an API integrated into an operating system running on virtual server[25]	**Block storage**; *Virtualization*; VersionOne can access centralized block storage thru an API integrated into an operating system running on virtual server[26]	**Block, Blob storage**; *Virtualization*; Azure Blob service used as a repository of build artifacts created by a Hudson	**Hypervisor**; *Virtualization*; Subversion can access centralized block storage thru an API integrated into an operating system running on virtual server	**Hypervisor**; *Virtualization*; Bugzilla can access centralized block storage thru an API integrated into an operating system running on virtual server
	Communication Offering	**Message-Oriented Middleware**; *Message Queuing*; JIRA Mail Queue[25]	**Message-Oriented Middleware**; *Message Queuing*; VersionOne's Defect Work Queues[26]	**Message-Oriented Middleware, Virtual networking**;*Message Queuing,Hypervisor*;Hudson's Execution System Queuing component	**Message-Oriented Middleware**;*Message Queuing*;Subversion's Repository layer[28]	**Message-Oriented Middleware**;*Message Queuing*; Bugzilla's Mail Transfer Agent[29]
Cloud Management	Management Components	**Provider Adapter, Managed Configuration, Elastic manager**;*RPC, API*; JIRA Connect Framework[25], JIRA Advanced configuration	**Managed Configuration**;*RPC, API*; VersionOne segregation and appl. configuration	**Elastic load balancer, watchdog**;*Elastic platform*; Hudson execution system's Load Balancer component)	**Managed Configuration**;*RPC, API*; configuration file is used to configure how/when builds are done	**Managed Configuration**;*RPC, API*; Bugzilla can use configuration file for tracking and correcting errors
	Management Processes	**Elastic management process**;*Elasticity Manager*; JIRA Elastic Bamboo, and Time Tracking feature[25]	**Elastic management process**;*Elasticity Manager*; VersionOne's OnDemand security platform[26]	**Update Transition process**;*Message Queuing*; continuous integration of codes by Hudson's CI server[27]	**Update Transition process**;*Message Queuing*; continuous updates of production versions of the appl. by Subversion[28]	**Resiliency management process**;*Elasticity platform*; Bugzilla Bug monitoring/reporting feature[29]
Composite Application	Decomposition Style	**3-tier**;*stateless, processing and data access components*; JIRA is web-based application[25]	**3-tier**;*stateless, processing and data access components*; VersionOne is a web application[26]	**3-tier, Content Dist. Network**;*user interface, processing, data access components, replica distr.*; Hudson is an extensible web application, code file replicated on multiple clouds[27]	**3-tier**;*stateless, processing and data access components*; Subversion is a web-based application [28]	**3-tier**;*stateless, processing and data access components*; Bugzilla is a web application[29]
	Hybrid Cloud Application	**Hybrid processing**; *processing component*; JIRA Agile used to track daily progress work[25]	**Hybrid Development Environment**;*processing component*; VersionOne's OpenAgile Integration[26]	**Hybrid Data, Hybrid Development Environment**; *data access component*;Separate environment for code verification and testing	**Hybrid Data, Hybrid Backup**; *data access component,stateless*;Code files extracted for external storage	**Hybrid Processing**; *processing component*; DB resides in data center, processing done in elastic cloud

support for the implementation of most deployment patterns is practically achieved through API integration [32]. For example, JIRA's Elastic Bamboo support for Blob storage on Windows Azure is through an API [25]. JIRA has a plugin for integrating with Hudson, Subversion and Bugzilla [25] and vice versa. The technologies used to support software processes of GSD tools are highlighted, unlike others which focus mostly on the design of cloud-native applications [6]. Web services (via REST) and messaging (via message queues) are the preferred technologies used by cloud deployment patterns (e.g., stateless pattern) to interconnect GSD tools and other components. REST style is favoured by public cloud platforms. For example, JIRA's support for SOAP and XML-RPC is depreciated in favour of REST [25]. This trend is also reported in [15][32].

D. Patterns for Cloud-application Versus Cloud-environment

Our taxonomy, can be used to guide an architect in focusing on a particular architectural deployment component of interest - that is, either cloud-hosted application or cloud-hosted environment. Other taxonomies [7][15] are concerned with the design of cloud-native applications. Assuming an architect is either interested in providing the right cloud resources, or mapping the business requirement to cloud properties that cannot be changed (e.g., location and ownership of the cloud infrastructure), then Taxonomy A would be more relevant. However, if interest is in mitigating certain cloud properties that can be compensated at an application level (e.g., improving the availability of the cloud-hosted GSD tool), then Taxonomy B should be considered. Fehling et al. describes other cloud properties that are either unchangeable or compensatable [6].

VI. RECOMMENDATIONS

Based on the experience gathered from positioning the GSD tools on the taxonomy, we present a set of recommendations in the form of selection criteria in Table 5 to guide an architect in choosing applicable deployment patterns for deploying any GSD tool. To further assist the architect in making a good choice, we also recommend that the architect should obtain information concerning- (i) the business requirements of the organization; and (ii) the architectural structure, installation

 ISBN: 978-1-61208-388-9

TABLE V. RECOMMENDATIONS FOR SELECTING APPLICABLE DEPLOYMENT PATTERNS FOR CLOUD DEPLOYMENT OF GSD TOOLS.

Category	**Sub-Category**	***Selection Criteria***	***Applicable Patterns***
Application Process	Project Processes	Elasticity of the cloud environment is not required	Static workload
	Implementation Processes	Expects continuous growth or decline in workload over time	Continuously changing workload
	Support Processes	Resources required is nearly constant;continuous decline in workload	Static workload, Continuously changing workload
Core Cloud Properties	Rapid Elasticity	Explicit requirement for adding or removing cloud resources	Elastic platform, Elastic Infrastructure
	Resource Pooling	Sharing of resources on specific cloud stack level-IaaS, PaaS, SaaS	Hypervisor, Standby Pooling Process
	Measured Service	Prevent monopolization of resources	Elastic Infrastructure, Platform, Throttling/Service Metering[7]
Cloud Service Model	Software Resources	No requirement to deploy and configure GSD tool	Software as a Service
	Platform Resources	Requirement to develop and deploy GSD tool and/or components	Platform as a Service
	Infrastructure as a Service	Requires control of infrastructure resources (e.g., storage, memory) to accommodate configuration requirements of the GSD tool	Infrastructure as a Service
Cloud Deployment Model	Private Usage	Combined assurance of privacy, security and trust	Private cloud
	Community Usage	Exclusive access by a community of trusted collaborative users	Community cloud
	Public Usage	Accessible to a large group of users/developers	Public cloud
	Hybrid Usage	Integration of different clouds and static data centres to form a homogenous deployment environment	Hybrid cloud
Application Architecture	Application Components	Maintains no internal state information	User Interface component, Stateless pattern
	Multitenancy	Many different users access and share the same resources	Shared component
	Integration	Integrate GSD tool with different components residing in multiple clouds	Integration provider, Restricted Data Access component
Cloud Offering	Cloud environment	Requires a cloud environment configured to suit PaaS or IaaS offering	Elastic platform, elastic infrastructure
	Processing Offering	Requires functionality to execute workload on the cloud	Hypervisor
	Storage Offering	Requires storage of data in cloud	Block storage, relational database
	Communication Offering	(1) Require exchange of messages internally between appl. components; (2) Require communication with external components	(1) Message-oriented middleware; (2) Virtual Networking
Cloud Management	Management Components	(1) Pattern supports Asynchronous access; (2) State information is kept externally in a central storage	(1) Provider Adapter; Elastic manager; Managed Configuration
	Management Processes	(1)Application component requires continuous update; (2) Automatic detection and correction of errors	(1) Update Transition process; (2) Resiliency management process
Composite Application	Decomposition Style	Replication or decomposition of application functionality/components	(1) 3-tier; (2) Content Distribution Network
	Hybrid Cloud Application	Require the distribution of functionality and/or components of the GSD tool among different clouds	(1) Hybrid processing; (2) Hybrid Data; (3) Hybrid Backup; (4) Hybrid Development Environment

and configuration requirements of the GSD tool (using a process such as IDAPO [5]). Based on this information, a suitable level of cloud stack that will accommodate all the configuration requirements of the GSD tool can be selected. The architect has more flexibility to implement or support the implementation of a deployment pattern when there is greater control of the cloud stack. For example, to implement the hybrid data pattern [6] during cloud deployment of Hudson, the architect would require control of the infrastructure level of the cloud stack to allow for provisioning (and de-provisioning) of resources (e.g., storage, memory, CPU).

VII. LIMITATIONS OF THE STUDY

The GSD tools included in the dataset were stable and mature, and used by all the companies involved in the empirical study. This reduces external threats to the study that may come from the proliferation of GDS tools deployed in the cloud. The study should not be generalized to small and medium size projects. Large projects are usually executed with stable and reliable GSD tools. For small projects (with few developers and short duration), high performance and low cost may be the main consideration in tool selection. The small number of GSD tools in the selected dataset is appropriate because we are not carrying out a feature-analysis based study of GSD tools, but only using it to apply against our proposed taxonomy. Future research can be done to re-evaluate how new GSD tools can be positioned within the taxonomy.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we have created and used a taxonomy of deployment patterns for cloud-hosted applications to contribute to the literature on cloud deployment of Global Software Engineering tools.

Eight categories that form the taxonomy have been described: Application process, Cloud properties, Service model, Deployment model, Application architecture, Cloud offerings, Cloud management, and Composite applications. *Application process* contains patterns that handles the workload imposed on the cloud infrastructure by the ISO/IEC 12207 software processes. *Cloud properties* contains patterns for mitigating the core cloud computing properties of the tools. Patterns in *Service model* and *Deployment model* reflects the NIST cloud definition of service models and deployment models, respectively. *Application architectures* contains patterns that supports the architectural components of a cloud-application. Patterns in *Cloud offerings* reflects the main offerings that can be provided to users on the cloud infrastructure. Cloud management contains patterns used to manage both the components and processes of software tools. *Composite cloud* contains patterns that can be formed by combining other patterns or can be decomposed into separate components.

These categories were further partitioned into 24 subcategories, which were mapped to the components of an (architectural) deployment structure. This mapping reveals two components classes: cloud-hosted environment and cloud-hosted application. Cloud-hosted environment and cloud-hosted application classes captures patterns that can be used to address deployment challenges at the infrastructure level and application level, respectively.

By positioning a selected set of software tools, JIRA, VersionOne, Hudson, Subversion and Bugzilla, on the taxonomy, we were able to identify applicable deployment patterns together with the supporting technologies for deploying cloud-hosted GSD tools. We observed that most deployment patterns are related and can be implemented by combining with others ones, for example, in hybrid deployment scenarios to integrate data residing in multiple clouds. We have also provided recommendations in a tabular form which shows the selection criteria to guide an architect in choosing applicable deployment patterns. Examples of deployment patterns derived

 ISBN: 978-1-61208-388-9

from applying these selection criteria have been presented.

We plan to carry out several Case Studies involving the deployment of cloud-hosted GSD tools to compare how well different deployment patterns perform under different deployment conditions with respect to specific software development processes (e.g., continuous integration with Hudson). In the future, we will evaluate performance and reliability (through simulation) in multi-user collaborations involving cloud-hosted GSD tools for different deployment patterns.

ACKNOWLEDGMENT

This research was supported by the Tertiary Education Trust Fund (TETFUND), Nigeria and IDEAS Research Institute, Robert Gordon University, UK.

REFERENCES

[1] R. Buyya, J. Broberg, and A. Goscinski, Cloud Computing: Principles and Paradigms. John Wiley & Sons, Inc., 2011.

[2] M. A. Chauhan and M. A. Babar, "Cloud infrastructure for providing tools as a service: quality attributes and potential solutions," in Proceedings of the WICSA/ECSA 2012 Companion Volume. ACM, 2012, pp. 5–13.

[3] S. Junuzovic and P. Dewan, "Response times in n-user replicated, centralized, and proximity-based hybrid collaboration architectures," in Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work. ACM, 2006, pp. 129–138.

[4] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice, 3/E. Pearson Education India, 2013.

[5] K.-J. Stol, P. Avgeriou, and M. A. Babar, "Design and evaluation of a process for identifying architecture patterns in open source software," in Software Architecture. Springer, 2011, pp. 147–163.

[6] C. Fehling, F. Leymann, R. Retter, W. Schupeck, and P. Arbitter, Cloud Computing Patterns. Springer, 2014.

[7] A. Homer, J. Sharp, L. Brader, M. Narumoto, and T. Swanson, Cloud Design Patterns, R. Corbisier, Ed. Microsoft, 2014.

[8] L. Badger, T. Grance, R. Patt-Corner, and J. Voas, "Cloud computing synopsis and recommendations," NIST special publication, vol. 800, 2012, p. 146.

[9] S. Hansma, "Go fast and don't break things: Ensuring quality in the cloud." in Workshop on High Performance Transaction Systems(HPTS 2011), Asilomar, CA, October 2011. Summarized in Conference Reports column of USENIX; login 37(1), February 2012., 2012.

[10] J. Bass, "How product owner teams scale agile methods to large distributed enterprises," Empirical Software Engineering, 2014, pp. 1–33.

[11] F. Lanubile, "Collaboration in distributed software development," in Software Engineering. Springer, 2009, pp. 174–193.

[12] J. D. Herbsleb and A. Mockus, "An empirical study of speed and communication in globally distributed software development," Software Engineering, IEEE Transactions on, vol. 29, no. 6, 2003, pp. 481–494.

[13] J. Portillo-Rodriguez, A. Vizcaino, C. Ebert, and M. Piattini, "Tools to support global software development processes: a survey," in Global Software Engineering (ICGSE), 2010 5th IEEE International Conference on. IEEE, 2010, pp. 13–22.

[14] J. Vlissides, R. Helm, R. Johnson, and E. Gamma, "Design patterns: Elements of reusable object-oriented software," Reading: Addison-Wesley, vol. 49, 1995.

[15] B. Wilder, Cloud Architecture Patterns, 1st ed., R. Roumeliotis, Ed. 1005 Gravenstein Highway North, Sebastopol, CA 95472.: O'Reilly Media, Inc., 2012.

[16] J. Buckley, T. Mens, M. Zenger, A. Rashid, and G. Kniesel, "Towards a taxonomy of software change," Journal of Software Maintenance and Evolution: Research and Practice, vol. 17, no. 5, 2005, pp. 309–332.

[17] D. Smite, C. Wohlin, Z. Galvina, and R. Prikladnicki, "An empirically based terminology and taxonomy for global software engineering," Empirical Software Engineering, 2012, pp. 1–49.

[18] C. Moyer, Building Applications for the Cloud: Concepts, Patterns and Projects. Pearson Education, Inc, Rights and Contracts Department, 501 Boylston Street, Suite 900, Boston, MA 02116, USA: Addison-Wesley Publishing Company, 2012.

[19] Z. Mahmood, Ed., Cloud Computing: Methods and Practical Approaches. Springer-Verlag London, 2013.

[20] N. Sawant and H. Shah, Big Data Application Architecture - A problem Solution Approach. Apress, 2013.

[21] S. Strauch, U. Breitenbuecher, O. Kopp, F. Leymann, and T. Unger, "Cloud data patterns for confidentiality," in Proceedings of the 2nd International Conference on Cloud Computing and Service Science, CLOSER 2012, 18-21 April 2012, Porto, Portugal. SciTePress, 2012, pp. 387–394.

[22] J. Varia, "Migrating your existing applications to the cloud a phase-driven approach to cloud migration." Amazon Web Services (AWS), [Online: retrieved in October, 2014 from http://aws.amazon.com/whitepapers/].

[23] L. Lilien, "A taxonomy of specialized ad hoc networks and systems for emergency applications," in Mobile and Ubiquitous Systems: Networking & Services, 2007. MobiQuitous 2007. Fourth Annual International Conference on. IEEE, 2007, pp. 1–8.

[24] P. Mell and T. Grance, "The nist definition of cloud computing," NIST special publication, vol. 800, no. 145, 2011, p. 7.

[25] Atlassian.com. Atlassian documentation for jira 6.1. Atlassian, Inc. [Online: retrieved in November, 2015 from https://www.atlassian.com/software/jira/].

[26] VersionOne, "Versionone-agile project management and scrum," [Online: retrieved in November, 2014 from www.versionone.com/].

[27] M. Moser and T. O'Brien, "The hudson book." Oracle, Inc., USA, [Online: retrieved in November, 2014 from http://www.eclipse.org/hudson/the-hudson-book/book-hudson.pdf].

[28] B. Collins-Sussman, B. Fitzpatrick, and M. Pilato, Version control with subversion. O'Reilly, 2004.

[29] Bugzilla.org. The bugzilla guide. [Online: retrieved in October, 2014 from http://www.bugzilla.org/docs/].

[30] CollabNet. Subversionedge for the enterprise. CollabNet, Inc. [Online: retrieved in October, 2014 from http://www.collab.net/products/subversion].

[31] I. Sommerville, Software Engineering. Pearson Education, Inc. and Addison-Wesley, 2011.

[32] J. Musser. Enterprise-class api patterns for cloud and mobile. CITO Research. (2012)

 ISBN: 978-1-61208-388-9