Clowiz: A Model-driven Development Platform for Cloud-based Information Systems

1st Jalal Kiswani
Department of Computer Science and Engineering
University of Nevada, Reno
Reno, NV, USA
jalal@nevada.unr.edu

2nd Sergiu M. Dascalu
Department of Computer Science and Engineering
University of Nevada, Reno
Reno, NV, USA
dascalus@cse.unr.edu

3rd Muhanna Muhanna
Department of Interactive Media
Luminus Technical University
Amman, Jordan
m.muhanna@luminuseducation.com

4th Frederick C Harris, Jr.
Department of Computer Science and Engineering
University of Nevada, Reno
Reno, NV, USA
fred.harris@cse.unr.edu

***Abstract*—Cloud-based solution of software systems development is currently the preferred approach over traditional on-premise one. In fact, utilizing cloud computing can reduce cost, time to market and allows for potential growth and scalability. However, this approach requires special quality attributes (i.e. non-functional requirements) to be taken into consideration. These attributes include traceability, recoverability, portability, and hot-deployability, along with the more common attributes of usability, reliability, efficiency, availability and security. Consequently, this increases the complexity of design and implementation of such applications. Therefore, such systems have a higher-cost and longer development time than traditional applications. In this paper, Clowiz is proposed. In particular, it is a cloud-based platform enabling rapid application development for building high-quality information systems applications. It is based on a model-driven development approach. In this approach, software developers shall use a web-based modeling toolkit to develop Internet based information systems, and modify them on the fly. The development process includes building application's views map (i.e. workflow), design systems entities (e.g. tables), and develop user interface forms. The platform enables application preview and deployment. In addition, it enables monitoring applications behavior at run time. The work presented in this paper contains a full waterfall software engineering process with full standard deliverables based on the UML modeling language, starting from requirements, and continuing to design and architecture, development, testing, and evolution.**

***Index Terms*—Cloud computing, software as a service, information systems , model-driven development, cloud applications.**

I. Introduction

Information Systems (IS) is an important category of software applications [1]. It is widely used to automate many parts of any business in any scale. Enterprise Resource Planning (ERP) systems are an examples of IS [2]. Traditional IS were developed using traditional development processes such as waterfall, that includes requirements, design, development and evolution [3]. Furthermore, traditional software development tools were used which included manual programming for forms and data persistence. In fact, relational database has been the standard data persistence mechanism for most IS [4].

However, developing IS in the traditional approach introduces many risks and higher percentage of failures. In particular, the gap in requirements, code duplications, and unavailability of technical expertise, are some of the risks [5]. Moreover, unawareness of best software architecture styles, patterns, and best practices, make it harder in achieving desired quality attributes of the system [6].

Even through IS development follows a general software development process most of the time, there are some special activities and tasks that are specific to the development of IS. They include database design and implementation, applications workflow, and entities operations. Database design and implementations include designing a logical view such as Entity Relationship Diagram (ERD) [7], and physical-view such as schema, tables and columns [4]. Applications workflow includes the automated navigation between pages or processes in the applications. The workflow is performed either grammatically or using a workflow engine such as jBPM [8]. On the other-hand, operations on entities (e.g. database tables), include operations such as Create, Read, Update and Delete (CRUD). In most IS, CRUD operations are programmed manually for every entity.

If those specific activities required for developing information systems are performed in a traditional way (e.g. manual programming), CRUD operations code will be duplicated, forms development, validation rules, fields masking, will be hard-coded inside applications, forms consistency will be hard to achieve which will consequently cause a lack of usability, and maintainability will become a serious risk [9].

As a solution for the high percentage of duplications in user-interface, functionality, and code, a meta-data driven approach can reduce development time, increase quality and achieve better usability [10]. Moreover, implementing and using the appropriate software infrastructure, such as version-control,

unit-testing, continuous-integration, and continuous-delivery, shall increase the software's overall quality, and reduce delivery time [11].

Even though implementing a meta-data driven approach, and utilization of the right software infrastructure in IS development have many benefits, some challenges still exist. In particular, the development requires software developers with specific expertise. In addition, any change to the meta-data or code requires compilations and new deployments [10].

Moreover, and with the current trend of cloud computing, the need of faster time to market, a reduced development cycle and cost, and higher quality software, a new approach is needed. In fact, the percentage of organizations from all levels that are adopting cloud computing increases year after year [12].

To overcome the risks and issues produced by IS's traditional software development, to build-on and enhance the metadata driven approach, and to enable building a cloud-based IS more effectively, Clowiz is presented. In particular, Clowiz is a platform for building high-quality cloud-based information systems rapidly and more effectively. In this platform, rapid application development is achieved by utilizing a Model-Driven Development (MDD) approach [13]. Furthermore, developers are able to build meta-data models for their intended applications using a web-based visual designer [14]. These models include entities and their relationships, forms, view workflows, validation rules, user-interface fields masking, and other. All these meta-data are utilized to build all application's assets dynamically and on the fly without the need for application recompilation or redeployment. These assets include user interface, data-access layer and services in between. Clowiz enables developers to build these models in an effective way using the Clowiz Implementation Toolkit (CIT). In particular, CIT is a web-based application and design tool considered as an online and realtime integrated development environment for software developers and domain users, where modifications can be reflected directly on applications, without the need of application redeployment in which is known as hot-deployment [15].

In Clowiz design, usability in the model's development was given a high-priority, to enable rapid application development and efficiency for developers.

To the best of our knowledge, the best available tools that can achieve some of the goals of Clowiz are Zoho creator [16] and Sales-force platform [17].

Even though the available options are solutions that have been developed and provided by companies that have long-term investment and expertise in cloud software systems. We think that their current functionalities are some what limited and does not achieve some of the objectives proposed in our solution. In particular, the novelty in the work presented in this paper is the cloud-based model-driven modeling technique and implementation toolkit that were designed to be modern and usable. In addition, the hot and dynamic generation of all application artifacts at runtime id done in an efficient and reliable way.

This paper is organized into 4 sections. This section covers the introduction and background. Then, it is followed by Section 2, which includes the implementation details of Clowiz. Section 3 discusses the results of the work presented in this paper. Finally, Section 4 concludes the paper and identifies several directions of future work.

II. Implementation

Clowiz can be considered both Business-to-Business (B2B) and Business-to-Consumer (B2C) business models [18]. Clowiz developers shall be able to develop IS applications either for themselves, their organizations, or their clients. In fact, those developers will not require to have technical or coding skills, so domain users without any software development background can be able to develop a cloud-based software applications with same quality or maybe even better than professional software developers. Therefore, the intended users of the platform are people who aim to build a cloud-based information system, and have the required domain expertise.

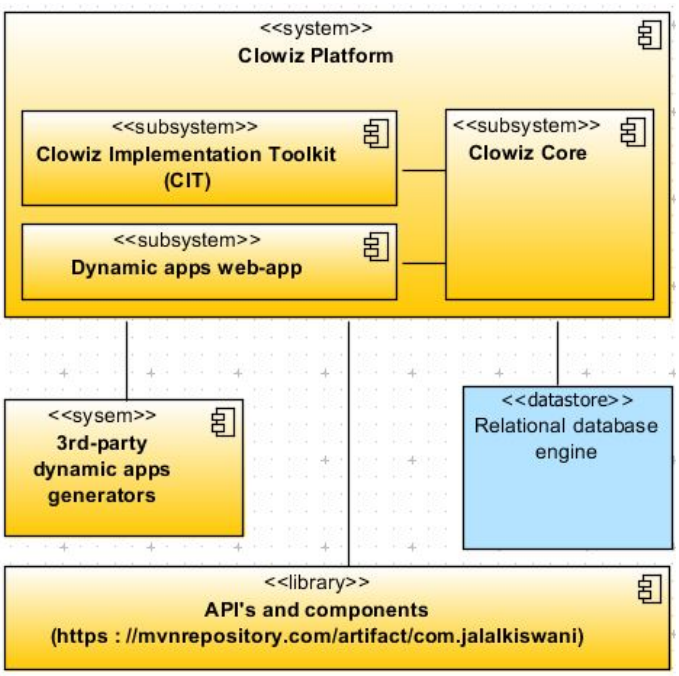

Fig. 1. Clowiz UML context diagram

Implementation of Clowiz followed the standard waterfall process model [3]. In particular, the process started by requirements definition phase, where interviews with stakeholders was conducted, followed by documenting these requirements in a Software Requirement Specification (SRS). The following phase was system and software design; in this phase, high-level, medium level, and low level design was created and documented. Later, software was developed in the implementation and unit testing phase. After that, the phase of integration and system testing was significant to insure the required software was built based on the specifications and was built right.

Finally, Clowiz was deployed in the maintenance and operation phase. Figure 1 shows the context diagram for it.

The following subsections will include stakeholder interviews, functional requirements, non-functional requirements, and use case models. The source code of Clowiz is available as open-source on Github, and can be found on the main author's GitHub repository [19]. The prototype of the final version can be accessed at the website [20].

A. Technology and Tools

During the different project life-cycle phases, Unified Modeling Language (UML) was extensively used [21]. On the other-hand, and from the technology perspective, portability is required to enable smooth and transparent migration to different cloud-providers, thus a platform independent technology is required. In addition, a mature and standard technology is needed to ensure long-term stability and support for the application. Furthermore, having a native-support for cloud-based features will guarantee an improved compatibility and performance. Therefore, Java technology [22] and Spring framework [23] was chosen to implement the back-end of Clowiz. On the other-hand, and to give the users more interaction and better user experience, Google Angular has been chosen as the front-end technology [24].

B. Stakeholders Interview

Requirements of Clowiz were based on interviews with information technology stakeholders in the financial and governmental sectors. In fact, those stakeholders are domain experts in these fields with long years (5-25) of experience. The following are the list of questions that were asked:

- **Q1**. Did your organization have developed any cloud-based applications so far? If yes, what are these systems? If not, what are the reason for not doing that?
- **Q2**. What do you think are the advantages of building cloud-based software applications?
- **Q3**. What do you think are the disadvantages of building cloud-based software applications?
- **Q4**. Does your organization have any concerns migrating legacy traditional systems or building new ones as cloud-based? Please explain.
- **Q5**. Do you think developing cloud-based software systems is different from developing traditional on-premise systems? Please explain.
- **Q6**. Do you think the current expertise and human resources available in the market are sufficient to migrate traditional systems or build new cloud-based solutions? Please explain.
- **Q7**. Does your organization have any plans to build cloud-based software systems in the next 5 years? Please explain
- **Q8**. Did your organization utilized any visual modeling tools for developing software systems? If yes, discuss this experience. If not, why?
- **Q9**. In the perfect world, what do you think would be the most efficient way to build cloud-based software applications? Please explain.
- **Q10**. If you find a solution that can enable building high-quality cloud-based software systems on much lower cost than traditional software development, would you consider it for your future projects?
- **Q11**. What do you prefer, enabling domain users to build software systems directly using model-driven approach, or to keep software development tasks conducted by developers? Please explain.

Based on the responses received in the interviews, all the interviewees think that cloud computing is the future of software industry, especially for B2C, small-medium businesses. On the other hand, they think that large and enterprise organizations such as banking and governments will keep their core software systems internal for many reasons related to security and political issues. However, they can definitely go for the cloud for non-core systems such as customers relations management (CRM) and recruitment systems. Regarding development costs, they think the main challenges of migrating and start building cloud-based software solutions, are immaturity of this field, lack of human resources that can build high quality software systems, and the unavailability of reliable and efficient tools. However, they all agreed that having a platform to build high-quality cloud-based solutions, with minimal efforts and cost, will certainly determine them move to the cloud faster.

C. Functional Requirements

Functional requirements of Clowiz are summarized in Table I. The numbering format is FR.X.Y.; X represents the release of Clowiz, future releases may include new features and requirements. Y represent the order of requirements based on its priority.

TABLE I
FUNCTIONAL REQUIREMENTS OF CLOWIZ

Requirement ID	Description
FR.1.1	Platform's users shall be able to model application's simple entities using visual representations
FR.1.2	Platform's users shall be able to model application's forms using visual representations
FR.1.3	Platform's users shall be able to model application workflow using visual representations
FR.1.4	Platform users shall be able to deploy application
FR.1.5	Application users shall be able to access dynamically generated forms and views

D. Non Functional Requirements

The non-functional requirements of Clowiz are shown in Table II. As discussed in the previous section, the numbering format is NFR.X.Y; X represents the release of the Clowiz, Y represents the order of the functional requirement based on the priority. Usability is given a higher priority because the authors think it is the most important factor for user traction in cloud-based applications.

TABLE II
NON-FUNCTIONAL REQUIREMENTS OF CLOWIZ

Requirement ID	Description
NFR.1.1	The applications shall be usable
NFR.1.2	The applications shall be hot-deployable
NFR.1.3	The applications shall be secured
NFR.1.4	The applications shall be scalable

E. Use Case models

Clowiz mainly have three user's roles: (i) platform users, (ii) application administrators, (iii) and application users. As shown in Figure 2, a platform user(i.e. application developer) can develop a dynamically generated application. In particular, he/she will be able to develop, preview, and manage applications.

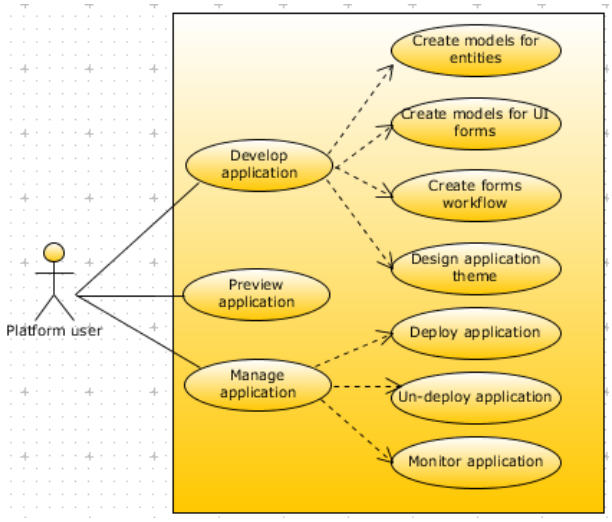

Fig. 2. Use case diagram for application development on the Clowiz platform

Figure 3 shows the use cases that can be executed on a dynamically generated applications by application administrators or application users. Application administrators will be able to manage application's security and configurations. On the other hand, the application users will be able to use the dynamically generated application.

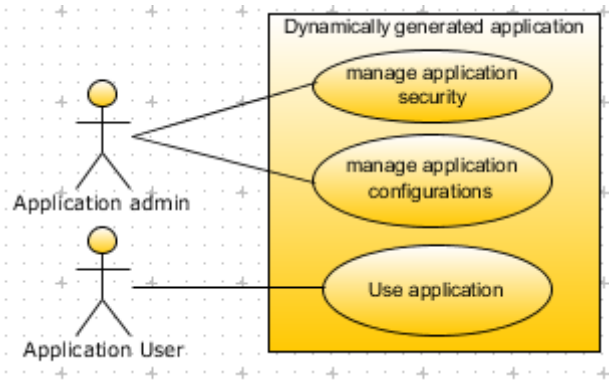

Fig. 3. Use case diagram for applications usage on the Clowiz platform

The core functionality of the platform is developing IS applications using MDD web-based tools. It enables users to develop all aspects of desired applications using visual models. This includes creating models for entities, their fields, and their relationships. In fact, these entities are considered the data-model of the target application. Furthermore, the entity models include all the main attributes of entity's fields. For example, they include the data-types, length, null-ability. In addition, they include some user-interface attributes, such as width of the form, masking, label, and translation for localization purposes. On the other hand, users shall be able to design models for user interface forms. These forms will be built from composites of entities and may have some calculated fields. Also, users shall be able to create the application's workflow among the modeled forms. Moreover, the user shall have the ability to create themes for their applications or pick current predefined themes.

Moreover, platform users shall be also able to preview applications directly from within the platform. This feature will enhance the users' experience and make them more productive, instead of consume their time switching between different views and browsers.

After finishing an application development and preview, and when the application is ready for use by end users, the platform user shall be able to manage the application, by being able to deploy or undeploy it. Furthermore, users can be able to monitor the application for load and security purposes.

After an application is deployed, it will be ready for use by application administrators and users. In particular, applications administrator can manage the security of application, and manage the general configurations. Managing the security includes roles, privileges, roles privileges, users, and user-roles. Furthermore, it includes management views to handle the application configurations such as mail server information. Consequently, the application will be ready for user to use from both technical and configuration perspectives.

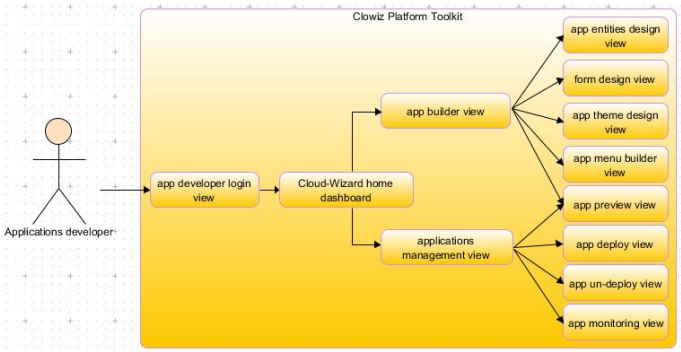

Fig. 4. Application-map for the Clowiz platform

Figure 4 shows the application-map for the Clowiz platform, and the names of views that can be accessed by platform users.

III. Results

The main results of the work presented in this paper is an open-source project available for public access at GitHub [19]. In addition, a functional prototype of Clowiz [20], has been built based on the waterfall process model. This section contains important prototype screenshots of the Clowiz implementation toolkit.

The entry point for the platform implementation toolkit is the Clowiz login page.

Fig. 5. Login page of the Clowiz platform

After the user logging in, he/she will be redirected automatically to the platform dashboard, as shown in Figure 6. In this view, application developers can create, view, deploy and undeploy applications. Moreover, current available applications will be shown with their status clear, with appropriate background color and available functionalities for each project. These statuses will be: (i) ready for deployment, (ii) not ready for deployment due to errors, (iii) deployed.

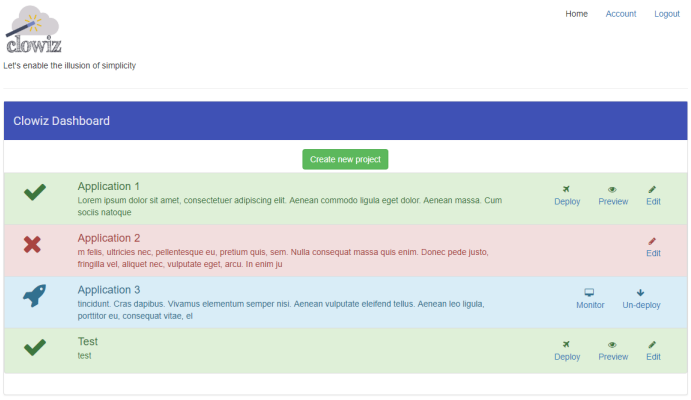

Fig. 6. Dashboard page of the Clowiz platform

Figure 7 shows the Clowiz application builder view. In this view, the user can input the general application's information, such as name and description. In addition, it includes a tabbed pane for the application's builder core components such as application views map, entity designs and theme configuration. In the views map, the developer can create new views and include them in the navigation. Furthermore, he/she can select one view and click on "open selected view in Form Designer" button to design the view.

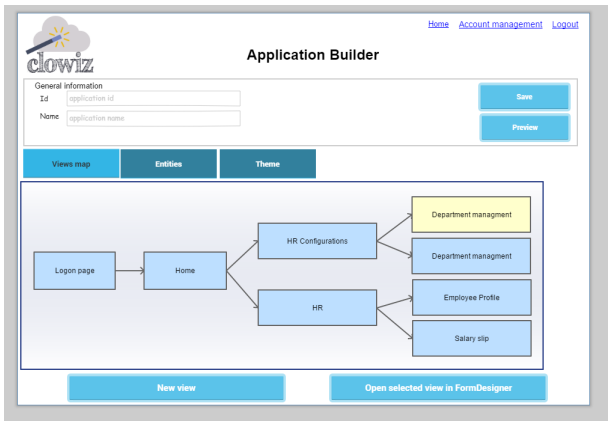

Fig. 7. Application Developer view of the Clowiz platform

In the Form Designer view, users can design the views using domain entities. A domain entity contains the meta-data for data that should be input, processed or outputted using the generated application instances. To allow users to create a domain entity's meta-data directly from this view without getting back to the application builder home page, New entity button is available to enable application developer to add entity meta-data directly. On the other-hand, the Entity View Designer enables application developers to design the domain entities meta-data using the drag and drop user interface tool. In particular, a developer can configure the entity attributes, their data types, and their relations with other entities.

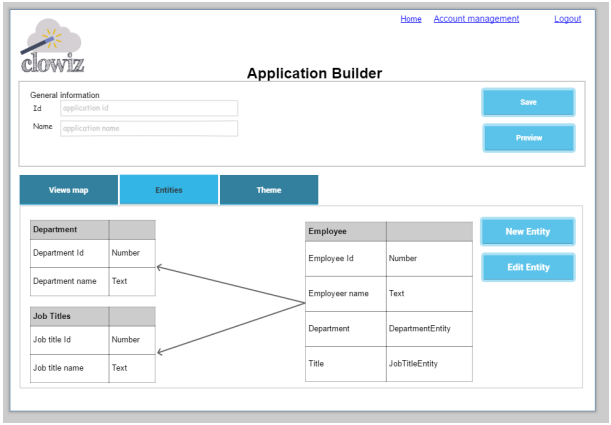

Fig. 8. Entity Developer view of the Clowiz platform

Using the application builder views, application developers can build an entire application's aspects and assets meta-data using a model driven approach with diagramming tools, without the need of having any coding skills.

IV. Conclusion and Future Work

In this paper, many aspects about information systems were discussed. In particular, their importance, characteristics, and the traditional way of developing them. Furthermore, the limitations and challenges of traditional development were presented, and how the meta-data development reduces the risk and development costs. However, since the meta-data approach requires developers, and doesn't fit well with the new trends of cloud computing, Clowiz was introduced. In particular, Clowiz includes a model-driven based platform, which is cloud-based, to enable rapid IS application development.

The waterfall approach was followed in the development of Clowiz and details about it were presented. This includes the scope of work which was defined by interviews with domain experts, followed by detailed design and architecture, development, testing and evolution. Furthermore, technology chosen, development process, deliverables, and final results were presented.

Clowiz source code is available at Github; In addition, a prototype is available publicly on the Internet.

Clowiz can be the base for future work in cloud and Internet based model-driven development applications. Some future directions may include enabling model driven development approach for building Big Data management and Big Data analysis web-based projects. In addition, it will be useful to make the platform more generic to include non IS systems. Furthermore, enabling integration through standard interfaces can make Clowiz interoperable and integrable with other systems.

V. Acknowledgment

This material is based upon work supported by the National Science Foundation under grant number IIA-1301726. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] David Avison and Guy Fitzgerald. *Information systems development: methodologies, techniques and tools*. McGraw Hill, 2003.

[2] Daniel E O'Leary. *Enterprise resource planning systems: systems, life cycle, electronic commerce, and risk*. Cambridge university press, 2000.

[3] Sommerville. *Software Engineering (10th Edition)*. AddisonWesley, 2015. pp.45-53.

[4] Ramez Elmasri and Shamkant Navathe. *Fundamentals of database systems*. Addison-Wesley Publishing Company, 2010.

[5] Elisabeth J Umble, Ronald R Haft, and M Michael Umble. Enterprise resource planning: Implementation procedures and critical success factors. *European journal of operational research*, 146(2):241–257, 2003.

[6] Rick Kazman Len Bass, Paul Clements. *Software Architecture in Practice (3rd Edition)*. "Addison-Wesley Professional", 2012.

[7] Peter Pin-Shan Chen. The entity-relationship modeltoward a unified view of data. In *Readings in artificial intelligence and databases*, pages 98–111. Elsevier, 1988.

[8] jBPM. Retrieved Jan 2018, from https://www.jbpm.org.

[9] Kent Beck, Martin Fowler, and Grandma Beck. Bad smells in code. *Refactoring: Improving the design of existing code*, pages 75–88, 1999.

[10] Jalal Kiswani, Muhanna Muhanna, and Abdullah Qusef. Using metadata in optimizing the design and development of enterprise information systems. In *Information and Communication Systems (ICICS), 2017 8th International Conference on*, pages 188–193. IEEE, 2017.

[11] Jalal Kiswani, Muhanna Muhanna, Sergiu Dascalu, and Frederick Harris. Software infrastructure to reduce the cost and time of building enterprise software applications: Practices and case studies. In *Proceedings of ISCA 26th International Conference on Software Engineering and Data Engineering (SEDE 2017)*. ISCA, 2017.

[12] RightScale. State of the cloud report. Technical report, RightScale, 2017.

[13] Oscar Pastor, Sergio España, José Ignacio Panach, and Nathalie Aquino. Model-driven development. *Informatik-Spektrum*, 31(5):394–407, 2008.

[14] Akon Dey, Gajanan Chinchwadkar, Alan Fekete, and Krishna Ramachandran. Metadata-as-a-service. In *Data Engineering Workshops (ICDEW), 2015 31st IEEE International Conference on*, pages 6–9. IEEE, 2015.

[15] Redhat. Hot vs cold deployment. Retrieved Jan 2018, from https://developer.jboss.org/wiki/HotVsColdDeployment.

[16] Zoho. Retrieved Jan 2018, from http://zoho.com.

[17] Salesforce. Retrieved Jan 2018, from http://salesforce.com.

[18] Allan Afuah and Christopher L Tucci. *Internet business models and strategies*. McGraw-Hill New York, 2001.

[19] Clowiz-source. Retrieved Jan 2018, from https://github.com/kiswanij/clowiz.

[20] Clowiz-website. Retrieved Jan 2018, from http://clowiz.com.

[21] Ivar Jacobson Grady Booch Rumbaugh, James. *Unified modeling language reference manual*. Pearson Higher Education, 2004.

[22] Java. Retrieved Jan 2018, from https://go.java/index.html?intcmp=gojava-banner-java-com.

[23] Spring cloud native applications. Retrieved Jan 2018, from https://pivotal.io/spring-app-framework.

[24] Angular. Retrieved Jan 2018, from http://angular.io.