

Creating portable TOSCA archive for iKnow University Management System

Magdalena Kostoska, Ivan Chorbev and Marjan Gusev
University Ss. Cyril and Methodius, Faculty of Computer Sciences and Engineering,
1000 Skopje, Macedonia
{magdalena.kostoska, ivan.chorbev, marjan.gushev}@finki.ukim.mk

Abstract—Portability of developed solutions is a huge challenge for both the customers and cloud providers. A typical customer would like to realize a flexible software solution which can be easily deployed on a selected infrastructure or transferred on a cloud environment. Finally, the idea of a software environment independent of a specific cloud provider enables preferable conditions for customers.

These portability issues are analyzed for a case study of a custom University Management System - iKnow, used by the University Ss. Cyril and Methodius in Macedonia. This paper presents implementation of a Topology and Orchestration Specification for Cloud Applications (TOSCA) specification to enable a software solution to be deployed on a flexible and portable environment.

The motivation to consider the implementation of such a platform is initiated due to the rising number of students that use the iKnow system, reaching the peak during the enrollment periods. A possible solution is to consider the possibility of moving the iKnow system (or its part) to a cloud environment and balance the workload using flexible resources. We conducted research to describe the iKnow system using the TOSCA specification and to enable a rather easy deployment and maintenance.

This paper presents the challenges and issues regarding the TOSCA specification while implemented on the iKnow system.

I. INTRODUCTION

IKNOW is the University Management Information system that provides electronic services for both university management and students [1]. It is deployed in most of the universities in Macedonia, including the University Ss Cyril and Methodius, where this research is conducted.

Our analysis shows peak usage of the iKnow system during the enrollment period. More than 7 000 applicants use it to register, fill applications, overview ranked lists, submit appeals and perform similar activities in the applying process. From the other side, more than 400 administration users within committees of 27 different faculties work on data checking, evaluation, ranking and similar processes. At the same time more than 25 000 enrolled students use the iKnow system for course enrollment and exam application. Along with the students, this system is also frequently used by the administrative staff of the faculties.

Heavy workload only in a short time period is a problem that motivates us to find an alternative solution to increase the bandwidth and processing power in the enrollment period, without buying additional hardware. Although, the natural solution for our needs is the cloud, there is one more demand, the University Management does not prefer "lock in" to a

specific cloud provider and would like a flexible platform that can be easily ported from one cloud provider to another or return to the existing University server.

A solution to these portability demands initiated our research on how to create easily portable, reusable and maintainable system specification, that we can use on any cloud provider, flexible enough to choose the provider offering the desired highest performances. We propose the usage of Topology and Orchestration Specification for Cloud Applications (TOSCA), a new standard for description of applications and services [2][3].

The main goal of the realized research was to explore the possibility of using TOSCA to describe the iKnow system and to address all the challenges of moving this system to the cloud, or changing the cloud provider. As a result, the research identifies TOSCA features and drawbacks of such an implementation.

The rest of the paper is organized as follows: Related work about cloud portability and TOSCA analysis is presented in Section II. Background about the TOSCA specification and the iKnow system (architecture and current deployment) is given in Sections III and IV respectively. The TOSCA model of the iKnow system is presented in Section V. Section VI describes the challenges and issues we have met during this research. Finally, in Section VII we evaluate the significance of each of the found challenges and we conclude with future work.

II. RELATED WORK

The main problem in Cloud computing is the lack of unified standards [4]. Several standards are on the way of their development, approval of the community and wider adoption [5].

Latest research considering the portability of cloud services in each of the layers of the service stack, tackle the portability problem from three different perspectives. One perspective is the usage of standards to accomplish this task. In this direction they try to identify current challenges, efforts and options [6], [7] or to extend open standards for this purpose [8]. The second perspective searches for other proprietary solutions like the abstraction-driven approach where abstract languages are used to specify solutions [9] or to create a storage abstraction layer (CSAL)[10]. The third perspective consists of other approaches that focus on exploiting the semantic technology to create semantic interoperability framework [11],

to automatically analyze cloud vendor APIs [12] or to use mOSAIC for this intention [4], [13].

OASIS (Organization for the Advancement of Structured Information Standards) introduces a new cloud standard called TOSCA (Topology and Orchestration Specification for Cloud Applications). The TOSCA technical committee has published the first version of a standard for "interoperable description of application and infrastructure cloud services, the relationships between parts of the service, and the operational behavior of these services (e.g., *deploy, patch, shutdown*)" [2]. This initiated a spread of research activities in different directions, such as IBM efforts to implement this standard [14], [15], creation of environment for execution [16], [17], modeling tools [18], [19] and attempt to create this type of specification [20].

III. TOSCA

This section presents a brief overview of the language, the structure and the usage of TOSCA.

Topology and Orchestration Specification for Cloud Applications (TOSCA) is developed by OASIS, a non-profit, international consortium. The main goal of the OASIS TOSCA Technical Committee is to enhance the portability of cloud applications and services by enabling interoperable description of application and infrastructure cloud services, the relationships between parts of the service, and the operational behavior of these services [2].

This specification utilizes open standards to describe a service and explain how to manage it independently from the supplier and the cloud. The current release of this specification is Version 1.0 and defines services by using a Service Template document. The TOSCA language introduces a grammar for describing service templates by means of Topology Templates and plans. TOSCA utilizes XML Schema 1.0 and WSDL 1.1. It also contains non-normative references to BPEL 2.0, BPMN 2.0, OVF and XPATH 1.0.

The specification defines a meta model for defining the structure of an IT service and its management. The core of TOSCA is the Service Template, depicted in Fig. 1. It consists of two logical parts: service topology description and management plans. The Topology Template defines the structure of a service. Plans define the process models that are used to create, terminate and manage a service.

The structure of the services is defined by the Topology Template. This template consists of Node Templates and Relationship Templates. The Node Template describes the required components and their properties, operations and interfaces, while the Relationship Template describes connectivity between components (Node Templates) by stating the direction of the relationship (source and target) and can have additional parameters. The Node Type and Relationship Type templates formally describe nodes and relationships (in terms of properties, interfaces etc.), while the templates are created and set concrete values based on these types definitions. The purpose of Plans is to interpret the templates and execute appropriate actions. They are defined as process models and

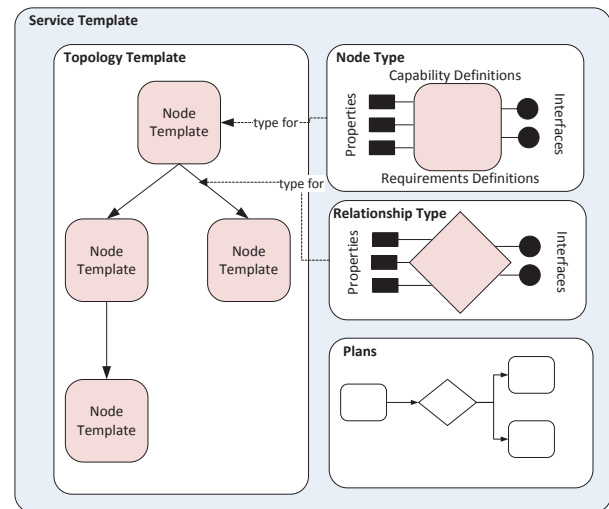


Fig. 1: TOSCA Service Template (adapted from [2])

they rely on BPMN (primarily) and BPEL. In addition, this specification allows nesting (ex. One Service Template to be part of another).

In order to allow automatic deployment or build of services TOSCA specification also defines Artifacts' installable or executional types and objects like scripts, libraries, installation binaries, installation images and other objects.

IV. IKNOW UNIVERSITY MANAGEMENT SYSTEM

The architecture and current deployment of the iKnow University Management System is presented in this section.

The main goal of the iKnow system is to provide electronic services and to store and exchange electronic information among students, professors, administration, university management and the Ministry of Education.

A. Architecture

The system consists of two main components, each consisting of modules: the new students Enrollment Student Services (ESS) and the Core Student Services (CSS) [1].

The Enrollment component consists of the faculty enrollment functionalities like enrollment wizard, manual entry of candidate's data and candidate's data processing, ranking and results publishing.

The Core student services component includes modules for administration, management of study programs and schedules, student activities, personal identification and access control, electronic payment and migration of legacy data.

The main architecture used to construct the system is NTier, depicted in Fig. 2. The two main components (Enrollment and Core) are almost independent of one another, except for several business processes (like transferring enrolled students into the core module). Although they use the same database server, physically they use two databases (enrollment and core

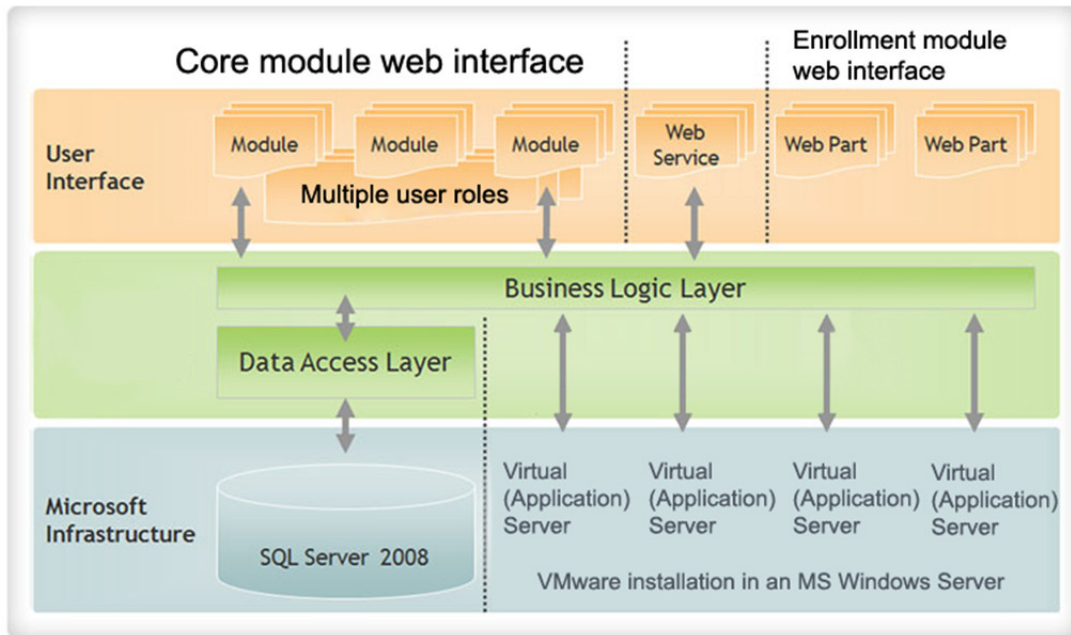


Fig. 2: iKnow architecture [1]

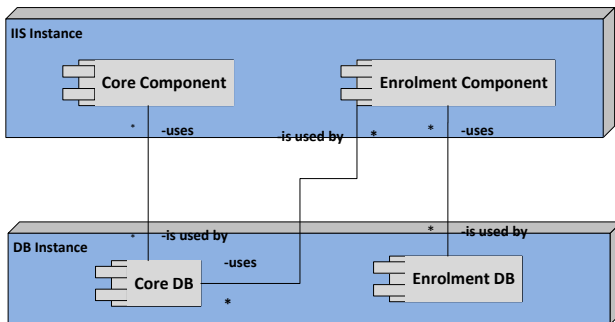


Fig. 3: Modules and database interactions

database). Fig. 3 shows the interactions among modules and databases.

The current solution consists of several project and both modules use some of them, like UniSS.DataModel, a Class library project containing the Entity framework model of the data as well as the audit log component. The UniSS.Repositories project is a Class library project storing the classes for implementation of the Repository template. The UniSS.Logic project contains the implementation of the MVP template, while UniSS.Forms is a Web Forms application project in which all forms are separated according to their functionalities.

The Model-View-Presenter (MVP) is used for all the modules [1].

B. Current Deployment

The current deployment includes two physical servers with implemented virtualization approach and an independent storage system. The first server is used as SQL server with Microsoft SQL Server installed and the second is used as an application server. The application server is upgraded by installing 4 virtual Windows servers using VMware. Additional Load Balancer balances the workload of each instance of the application. Both servers have an Intel Xeon e5630 2.56 GHz CPU and 16GB of RAM, 1 TB storage. The system is supported by additional backup internet link in case of failures of the primary link. In addition, a third fully synchronized server is installed on remote location and can take over in case of failure of the primary servers. All communications are encrypted using SSL protocol and a digital certificate.

V. IKNOW TOSCA SPECIFICATION AND ARTIFACTS

The TOSCA specification for the iKnow University Management System will be presented by specification of corresponding nodes, relationships, artifacts and plans.

A. Nodes

The first step is to identify the logical set of component services that the application is both composed of, as well as those services that the application relies upon for deployment, installation, execution and other lifecycle operations [21]. Each of these basic components, their interfaces and properties are described as TOSCA *Node Types*. TOSCA recommends creation of three node types: base, specific and custom. Fig. 4

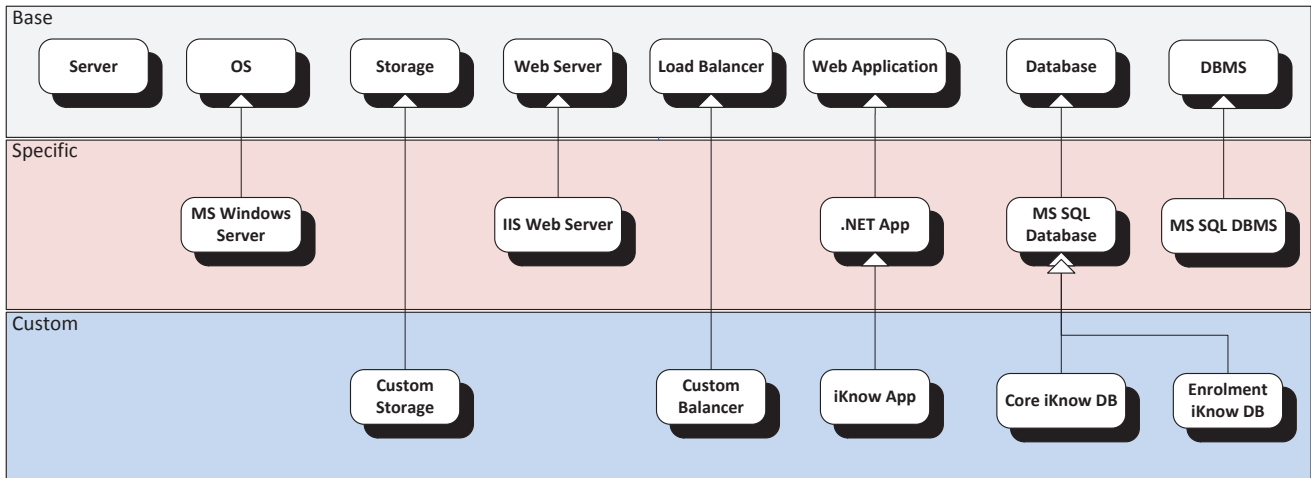


Fig. 4: Node Type Modeling and Node Type Inheritance Hierarchy for iKnow

illustrates all node types and inheritance hierarchy for the iKnow system.

The *Base Node Types* describe the basic set of components services needed for the proper functioning of the application. All Base Node Types are directly derived from a generic TOSCA root node type. Listing 1 shows part of the definition, due to the large specification. For the same reason the name spaces are removed.

```
<NodeType name="Server">
  <DerivedFrom typeRef="RootNodeType"/>
  <RequirementDefinitions>
    <RequirementDefinition lowerBound="0"
      name="container"
      requirementType="ServerContainerRequirement"
      upperBound="1"/>
  </RequirementDefinitions>
  ...
</NodeType>
<NodeType name="WebServer">
  <DerivedFrom typeRef="RootNodeType"/>
  <Operation name="install"/>
  <Operation name="configure"/>
  <Operation name="start"/>
  <Operation name="stop"/>
  <Operation name="uninstall"/>
  ...
</NodeType>
<NodeType name="LoadBalancer">
  <DerivedFrom typeRef="RootNodeType"/>
  <Operation name="install"/>
  <Operation name="configure"/>
  <Operation name="start"/>
  <Operation name="stop"/>
  <Operation name="uninstall"/>
  ...
</NodeType>
<NodeType name="WebApplication">
  <DerivedFrom typeRef="RootNodeType"/>
```

```
<Interfaces>
  <Interface name="lifecycle">
    <Operation name="install"/>
    <Operation name="configure"/>
    <Operation name="uninstall"/>
  </Interface>
</Interfaces>
...
</NodeType>
<NodeType name="Database">
  <DerivedFrom typeRef="RootNodeType"/>
  <Interfaces>
    <Interface name="lifecycle">
      <Operation name="install"/>
      <Operation name="uninstall"/>
      <Operation name="start"/>
      <Operation name="stop"/>
    </Interface>
  </Interfaces>
  ...
</NodeType>
<NodeType name="DBMS">
  <DerivedFrom typeRef="RootNodeType"/>
  <Interfaces>
    <Interface name="lifecycle">
      <Operation name="install"/>
      <Operation name="configure"/>
      <Operation name="start"/>
      <Operation name="stop"/>
      <Operation name="uninstall"/>
    </Interface>
  </Interfaces>
  ...
</NodeType>
```

Listing 1: Definitions of Basic Node Types

The next step is to extend the base types to represent specific instances (to define *Specific Node Types*). They are derived directly from the base types. In our use case, we define five

specific node types: MS Windows Server, IIS Web Server, .NET Application, MS SQL Database and MS SQL Database Management Service. Listing 2 shows the definition for IIS Web Server.

```
<NodeType name="IISWebServer">
  <DerivedFrom typeRef="WebServer"/>
  <Interfaces>
    <Interface name="lifecycle">
      <Operation name="install"/>
      <InputParameters>
        <InputParameter name="httpport"
          type="integer"/>
        <InputParameter name="username"
          type="string"/>
        <InputParameter name="password"
          type="string"/>
        ...
      </InputParameters>
    </Interface>
  </Interfaces>
</NodeType>
```

Listing 2: Definition of IIS Web Server Specific Node Type

Furthermore we can extend the specific node types to *Custom Node Types*, which are the custom build elements of the solution. In the iKnow system use case we define five custom node types for the storage, the balancer, the iKnow application as well as the two databases (Core and Enrolment). Listing 3 shows the definition for the Core database.

```
<NodeType name="CoreiKnowDB">
  <DerivedFrom typeRef="MSSQLDB"/>
  <Interfaces>
    <Interface name="lifecycle">
      <Operation name="install"/>
      <Operation name="start"/>
      <Operation name="stop"/>
      <Operation name="uninstall"/>
      <Operation name="backup"/>
    </Interface>
  </Interfaces>
</NodeType>
```

Listing 3: Definitions of Core DB custom node types

The final step in the node definition is to create *Node Templates*, based on the defined node types, which can be instantiated with specific properties. Node types only describe properties, requirements and capabilities, while Node Templates provide specific property settings to be applied when the specification is deployed on the cloud provider. Listing 4 shows the definition for the IIS Web Server Node Template.

B. Relationships

TOSCA uses *Relationship Types* to describe how these TOSCA node relate to each other in the cloud deployment. The same as node, the relationship type can be base, specific and custom. Some of the relationship types are shown in Listing 5.

Similar to Node Templates, TOSCA *Relationship Templates* are describing the logical relationships and other dependencies between the application's node templates and are needed for

deployment. It describes the source and target Node Templates. Listing 6 describes the relationship between iKnow Core Database and MS SQL by using the Requirement and Capability properties defined in this node templates.

```
<NodeTemplate id="IISWebServer"
  name="IIS Web Server"
  type="IISWebServer">
  <Properties>
    <httpport>80</httpport>
    <username>sa</username>
    <password>sa</password>
  </Properties>
</NodeTemplate>
```

Listing 4: Definitions of IIS Web Server Node Template

```
<RelationshipType name="DeployedOn">
  <DerivedFrom
    typeRef="RootRelationshipType"/>
  <ValidSource typeRef="WebApplication"/>
  <ValidTarget typeRef="WebServer"/>
</RelationshipType>

<RelationshipType name="ConnectsTo">
  <DerivedFrom
    typeRef="RootRelationshipType"/>
  <ValidSource typeRef="WebApplication"/>
  <ValidTarget typeRef="Database"/>
</RelationshipType>

<RelationshipType name="HostedOn">
  <DerivedFrom
    typeRef="RootRelationshipType"/>
  <ValidSource typeRef="Database"/>
  <ValidTarget typeRef="DBMS"/>
</RelationshipType>

<RelationshipType
  name="CoreDBHostedOnMSSQLDBMS">
  <DerivedFrom typeRef="HostedOn"/>
  <SourceInterfaces>
    <Interface name="HostedOn">
      <Operation name="hostOn"/>
    </Interface>
  </SourceInterfaces>
  <ValidSource typeRef="CoreiKnowDB"/>
  <ValidTarget typeRef="MSSQLDBMS"/>
</RelationshipType>

<RelationshipType
  name="iKnowAppDeployedOnIISWebServer">
  <DerivedFrom typeRef="DeployedOn"/>
  <SourceInterfaces>
    <Interface name="DeployedOn">
      <Operation name="deployOn"/>
    </Interface>
  </SourceInterfaces>
  <ValidSource typeRef="iKnowApp"/>
  <ValidTarget typeRef="IISWebServer"/>
</RelationshipType>
```

Listing 5: Sample of iKnow Relationship Types

```

<RelationshipTemplate
  id="CoreDB_HostedOn_MSSQLDBMS"
  name="hosted on"
  type="CoreDBHostedOnMSSQLDBMS">
  <SourceElement ref="CoreDB_container"/>
  <TargetElement ref="MSSQLDBMS_databases"/>
</RelationshipTemplate>

<NodeTemplate id="CoreDatabase"
  name="iKnowCoreDB"
  type="CoreDB">
  <Properties>... </Properties>
  <Requirements>
  <Requirement id="CoreDB_container"
    name="container"
    type="MSSQLDBContainerReq"/>
  </Requirements>
  <Capabilities>...</Capabilities>
</NodeTemplate>

<NodeTemplate id="MSSql" name="MS_SQL"
  type="MySQL">
  <Properties>... </Properties>
  <Requirements>... </Requirements>
  <Capabilities>
  <Capability id="MSSql_databases"
    name="databases"
    type="MSSQLDBContainerCap"/>
  </Capabilities>
</NodeTemplate>

```

Listing 6: Sample Relationship Template for Node Templates

C. Artifacts

Artifacts are needed in order to deploy and install the cloud application. They may vary from scripts, files to packages and virtual images.

```

<ArtifactType name="FileArtifact">
  <DerivedFrom typeRef="RootArtifactType"/>
</ArtifactType>

<ArtifactType name="ScriptArtifact">
  <DerivedFrom typeRef="RootArtifactType"/>
  <PropertiesDefinition
    element="ScriptArtifactProperties"/>
</ArtifactType>

<ArtifactType name="ArchiveArtifact">
  <DerivedFrom typeRef="RootArtifactType"/>
  <PropertiesDefinition
    element="ArchiveArtifactProperties"/>
</ArtifactType>

<ArtifactType name="PackageArtifact">
  <DerivedFrom typeRef="RootArtifactType"/>
  <PropertiesDefinition
    element="PackageArtifactProperties"/>
</ArtifactType>

```

Listing 7: TOSCA Primer Artifacts Types used for iKnow[21]

The installation of the iKnow system on the cloud environment requires careful setting of the environment, artifact to set

the service components (like the required web application and database servers) and packages and file to deploy and host the custom solutions. All of the elements must be orchestrated and set in a predefined order.

The basic artifact types for the iKnow use case are: file, script, package and archive artifacts. Since the TOSCA primer already defines these types, we use the same. Listing 7 shows the basic artifacts defined by TOSCA primer [21].

The archive artifacts are used for packaging a collection of files and can contain additional metadata. In the iKnow use case these types are used for packaging the storage files. The package artifacts contain collections of software application or service files and in our case are used for the application deployment code. Listing 8 shows the created archive artifact for this intention. The script packages are used for deployment, configuration and similar activities.

```

<ArtifactTemplate id="iKnowApp"
  name="iKnowApplication-archive"
  type="ArchiveArtifact">
  <Properties>
  <ArchiveArtifactProperties>
  <ArchiveInformation
    archiveReference="files/iKnowApp.zip"
    archiveType="zip"/>
  </ArchiveArtifactProperties>
  </Properties>
  <ArtifactReferences>
  <ArtifactReference
    reference="files/iKnowApp.zip"/>
  </ArtifactReferences>
</ArtifactTemplate>

```

Listing 8: Archive Artifact used for iKnow application

D. Plans

Plans are defined as process models. TOSCA relies on BPMN or BPEL languages to define a plan. There is no official support for creation of management plans. For that reason we have created diagrams that can be transformed into one of the offered languages. At this moment the plans for iKnow system include deployment and server instantiation management in case of increased workload of the system. Fig. 5 depicts the deployment topology plan this application.

VI. DISCUSSION

TOSCA is a recent proposal for standard and it is still in its early stage of adoption, presented by the current version 1.0. There are no commercial environments that enable usage of this specification, besides the several efforts made in this area for open-source solutions [17], [16]. Although it is promising, there is no guarantee of a wider acceptance by major cloud providers and software developers; and further standard commercialization.

Another possible approach is the usage of BPMN to define plans that will manage the life cycle of the application, especially for more complex management. BPMN is a general-purpose language and does not offer support for creation of

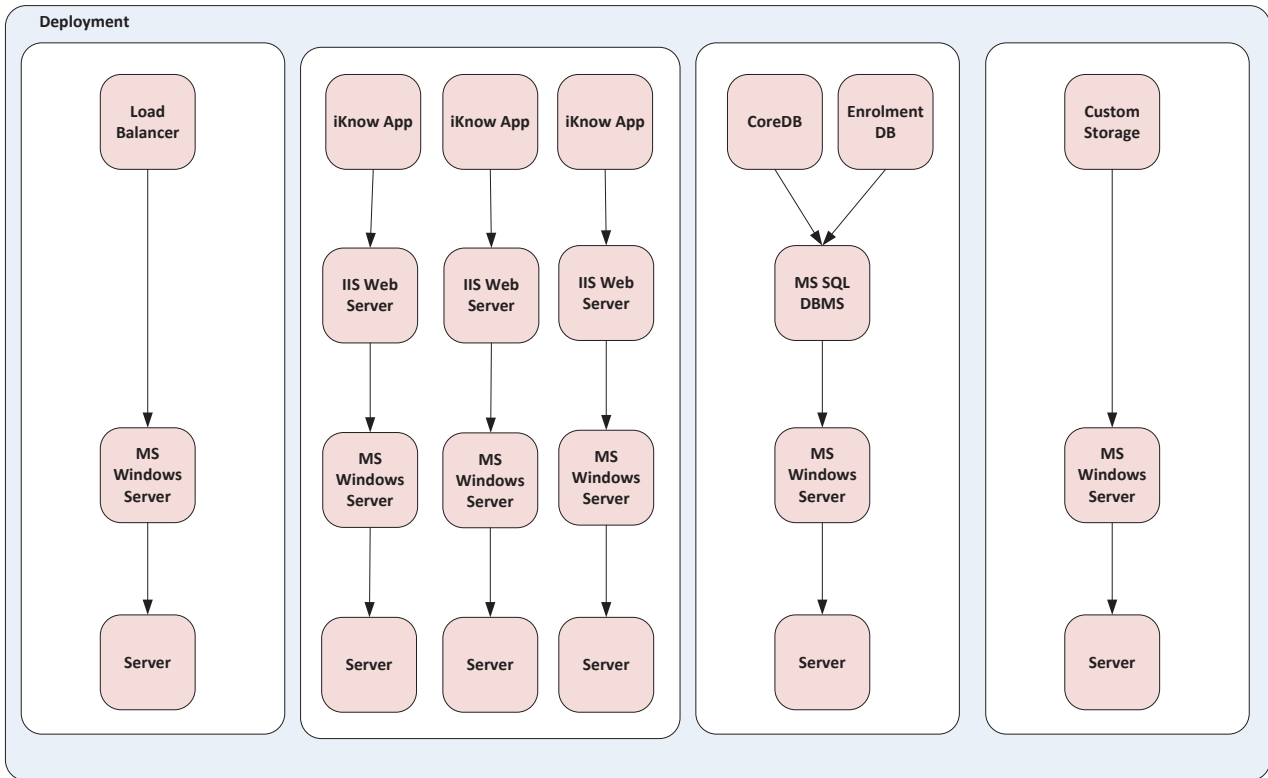


Fig. 5: Deployment Topology for iKnow

management plans. An effort in this direction is in process [22], but without commercial environment to support this effort, the results are not visible yet. This is the reason why the plans for the iKnow use case are still work in progress.

On the other hand TOSCA offers flexible definition of a software application. For smaller applications it offers straightforward way of definition, but for more complex applications, the specification may become very large and complex. In this paper we have analyzed the iKnow case study, as a mid-size project, where the specification grows fast. Usage of modeling tools like Winery [18] can ease this process. The greatest benefit is the promise of re-using the same specification by different cloud providers and environments. Another benefit, in this moment, is the formalized specification, which can be transformed in future and used for other purposes.

To create a TOSCA specification for a given application is a huge challenge at this time, since there is only one published primer [21] and it does not offer a lot of details. Creation of some parts of the iKnow TOSCA specification was based on assumptions considering the absence of concrete environment to test, like proper configuring of the environment with scripts, ability to convert the plans for installation and instantiation management into BMPL, etc.

An important concern is the usage of commercial solutions (in the case of iKnow it is based on Microsoft solutions, such

as MS Server, MS SQL Database etc.). Due to the closed environment it is not always possible to properly configure all the necessary data with scripts or APIs. Other main concern is the security definition and settings. This is still a crucial work-in-progress area in the cloud.

Initial steps towards specifying an automated general procedure to extend the TOSCA specification and enable portability of SaaS applications has been reported by the authors in [23]. Automated portability was analysed in [24]. We have also demonstrated how a general procedure can be applied on a simple SOA transactional-based service example [25].

VII. CONCLUSION

In this paper we have presented a TOSCA specification of the iKnow University Management System, which is so far a unique example and case study to apply this new standard for university management software solutions. The proposed specification and challenges we have faced are not specific only for the given software solution, but can be rather used as a case study to ease the process of TOSCA specification of similar solutions.

Besides the benefits that TOSCA offers, we have pointed current drawbacks of this specification. This work contributes to support the future progress of TOSCA as a developing

standard with bright future in the area of cloud provision of interoperable services.

As future work we plan to continue working and expanding this specification with plans and to help creating an environment where this specification can be tested. We also plan to redefine and rework the iKnow system to create two independent modules (Core and Enrollment) so they can be independently transferred to separate platforms.

ACKNOWLEDGMENT

This work was partially financed by the Faculty of Computer Science and Engineering at the "Ss. Cyril and Methodius" University, Skopje, Macedonia through the project XaaS-in-Cloud (Everything-as-a-Service in Cloud). All testing was performed on facilities at the Laboratory of Innovative Internet Technologies.

The iKnow project has been developed as a part of the Tempus JPGR 511342 Project.

REFERENCES

- [1] I. Chorbev, M. Gusev, D. Gjorgjevikj, and A. Madevska-Bogdanova, "Architecture of an electronic student services system and its implementation," in *ICT Innovations 2012, Web proceedings*, S. Markovski and M. Gusev, Eds., 2012. ISBN 1857-7288 pp. 381 – 400.
- [2] Organization for the Advancement of Structured Information Standards. (2013, Mar.) Topology and orchestration specification for cloud applications version 1.0. [Online]. Available: <http://docs.oasis-open.org/tosca/TOSCA/v1.0/cs01/TOSCA-v1.0-cs01.html>
- [3] T. Binz, G. Breiter, F. Leymann, and T. Spatzier, "Portable cloud services using toasca." *IEEE Internet Computing*, vol. 16, no. 3, pp. 80–85, 2012. doi: 10.1109/MIC.2012.43
- [4] F. Moscato, R. Aversa, B. Di Martino, T. Fortis, and V. Munteanu, "An analysis of mosaic ontology for cloud resources annotation," in *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on*. IEEE, 2011, pp. 973–980.
- [5] S. Labes, J. Repschlager, R. Zarnekow, A. Stanik, and O. Kao, "Standardization approaches within cloud computing: evaluation of infrastructure as a service architecture," in *Computer Science and Information Systems (FedCSIS), 2012 Federated Conference on*. IEEE, 2012, pp. 923–930.
- [6] G. A. Lewis, "Role of standards in cloud-computing interoperability," in *System Sciences (HICSS), 2013 46th Hawaii International Conference on*. IEEE, 2013. doi: 10.1109/HICSS.2013.470 pp. 1652–1661.
- [7] Z. Zhang, C. Wu, and D. W. Cheung, "A survey on cloud interoperability: taxonomies, standards, and practice," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 4, pp. 13–22, 2013.
- [8] F. Galán, A. Sampaio, L. Ródero-Merino, I. Loy, V. Gil, and L. M. Vaquero, "Service specification in cloud environments based on extensions to open standards," in *Proceedings of the Fourth International ICST Conference on COMMunication System softWARE and middleWARE*. ACM, 2009. doi: 10.1145/1621890.1621915 pp. 1–12.
- [9] A. Ranabahu, E. Maximilien, A. Sheth, and K. Thirunarayan, "Application portability in cloud computing: An abstraction driven perspective," *IEEE Transactions on Services Computing*, vol. PP, no. 99, p. 1, 2013. doi: 10.1109/TSC.2013.25
- [10] Z. Hill and M. Humphrey, "CSAL: A cloud storage abstraction layer to enable portable cloud applications," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*. IEEE, 2010. doi: 10.1109/CloudCom.2010.88 pp. 504–511.
- [11] N. Loutas, E. Kamateri, and K. Tarabanis, "A semantic interoperability framework for cloud platform as a service," in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*. IEEE, 2011, pp. 280–287.
- [12] G. Cretella and B. Di Martino, "Towards automatic analysis of cloud vendors apis for supporting cloud application portability," in *Complex, Intelligent and Software Intensive Systems (CISIS), 2012 Sixth International Conference on*. IEEE, 2012. doi: 10.1109/CISIS.2012.162 pp. 61–67.
- [13] D. Petcu, "How to build a reliable mOSAIC of multiple cloud services," in *Proceedings of the 1st European Workshop on Dependable Cloud Computing*, ser. EWDC '12. ACM, 2012. doi: 10.1145/2365316.2365320. ISBN 978-1-4503-1149-6 pp. 4:1–4:2.
- [14] G. Breiter, M. Behrendt, M. Gupta, S. Moser, R. Schulze, I. Sippli, and T. Spatzier, "Software defined environments based on TOSCA in IBM cloud implementations," *IBM Journal of Research and Development*, vol. 58, no. 2, pp. 1–10, 2014. doi: 10.1147/JRD.2014.2304772
- [15] S. Durairajan and P. Sundararajan, "Portable service management deployment over cloud platforms to support production workloads," in *Cloud Computing in Emerging Markets (CCEM), 2013 IEEE International Conference on*, Oct 2013. doi: 10.1109/CCEM.2013.6684438 pp. 1–7.
- [16] T. Binz, U. Breitenbücher, F. Haupt, O. Kopp, F. Leymann, A. Nowak, and S. Wagner, "OpenTOSCA – a runtime for TOSCA-based cloud applications," in *11th International Conference on Service-Oriented Computing*, ser. LNCS, vol. 8274. Springer, 2013. doi: 10.1007/978-3-642-45005-1_62 pp. 692–695.
- [17] M. Kostoska, M. Gusev, and S. Ristov, "A new cloud services portability platform," *Procedia Engineering*, vol. 69, no. 0, pp. 1268 – 1275, 2014. doi: 10.1016/j.proeng.2014.03.118
- [18] O. Kopp, T. Binz, U. Breitenbücher, and F. Leymann, "Winery – modeling tool for TOSCA-based cloud applications," in *11th International Conference on Service-Oriented Computing*, ser. LNCS, vol. 8274. Springer, 2013, pp. 700–704.
- [19] U. Breitenbücher, T. Binz, O. Kopp, F. Leymann, and D. Schumm, "Vino4TOSCA: A visual notation for application topologies based on TOSCA," in *OTM 2012, Part I*, ser. LNCS, vol. 7565. Springer-Verlag, 2012. doi: 10.1007/978-3-642-33606-5_25 pp. 416–424.
- [20] F. Li, M. Vogler, M. Claessens, and S. Dustdar, "Towards automated IoT application deployment by a cloud-based approach," in *Service-Oriented Computing and Applications (SOCA), 6th IEEE International Conference on*, 2013, pp. 61–68.
- [21] Organization for the Advancement of Structured Information Standards. (2013, Jan.) Topology and orchestration specification for cloud applications (TOSCA) primer version 1.0. [Online]. Available: <http://docs.oasis-open.org/tosca/tosca-primer/v1.0/tosca-primer-v1.0.html>
- [22] O. Kopp, T. Binz, U. Breitenbücher, and F. Leymann, "BPMN4TOSCA: A domain-specific language to model management plans for composite applications," in *Business Process Model and Notation*, ser. Lecture Notes in Business Information Processing, vol. 125. Springer, 2012. doi: 10.1007/978-3-642-33155-8_4. ISBN 978-3-642-33154-1 pp. 38 – 52.
- [23] M. Kostoska, M. Gusev, and S. Ristov, "P-TOSCA portability model for PaaS hosted applications," University Ss Cyril and Methodius, Faculty of Computer Sciences and Engineering, Skopje, Macedonia, Tech. Rep. LiIT 22, 2014.
- [24] M. Gusev, M. Kostoska, and S. Ristov, "P-TOSCA automated application portability," University Ss Cyril and Methodius, Faculty of Computer Sciences and Engineering, Skopje, Macedonia, Tech. Rep. LiIT 55, 2014.
- [25] S. Ristov, M. Kostoska, and M. Gusev, "P-TOSCA portability demo case," in *2014 IEEE 3rd Int. Conf. on Cloud Networking (IEEE CLOUDNET)*, 2014.