Using Constraint Reasoning on Feature Models to Populate Ecosystem-driven Cloud Services e-Marketplace

Azubuike Ezenwoke, Olawande Daramola, Matthew Adigun

[azu.ezenwoke; olawande.daramola] @covenantuniversity.edu.ng
madigun@pan.uzulu.ac.za

Abstract

Service providers leverage cloud ecosystems and cloud e-marketplaces to increase the business value of their services and reach a wider range of service users. A cloud ecosystem enable participating services to combine with other services, along their QoS properties; while the e-marketplace provides an environment where atomic services interconnect in unprecedented ways to be traded on the marketplace platform. Noting the unprofitability, impracticality and error-prone nature of performing ad hoc service combination of atomic services, the concern addressed in this technical report is how to guide the combination of atomic services participating in an ecosystem in a seamless manner. In this technical report, we proposed the use of feature models to model the inter-relationships and constraints among the atomic services, which is transformed into a constraint satisfaction problem and off-the-shelve constraint solvers are used to determining valid combinations. The collection of valid combinations become the blueprint that guides service composition and populates the e-marketplace service directory; users can then make service selection decisions based on the list. The applicability of the approach proposed in this report is demonstrated via an example of Customer relationship management as a service ecosystem**.**

Keywords: Cloud Computing; ecosystem; e-marketplace; feature model; constraint satisfaction problem

I. INTRODUCTION

Cloud computing is a model of internet-based service provisioning where dynamically scalable and virtualized resources (such as infrastructure, platform and software) are delivered and accessed as services over the internet [1; 2]. Cloud computing is technologically enabling new business models, which may not have existed before [2]. Gartner predicts that from 2013 through 2016, $677 billion will be spent on cloud services worldwide, and $310 billion of which will be spent on cloud advertising business by 2014[1], making cloud computing a growing phenomenon in the IT landscape. Cloud computing has been referred to as the fifth utility along with electricity, gas, water and telecommunication services [3]. Although basic cloud computing service models are Infrastructure as a service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [2]; however, more complex models evolves into the concept of Anything- or Everything-as-a-service (XaaS). Because everything and anything can be offered as a service, the maturity of cloud computing is fast tracked by commoditizing services in an e-marketplace facilitated by cloud ecosystem [4; 5].

[1] http://www.gartner.com/resId=2332215

A service ecosystem consists of a platform, a set of internal and external providers and a community of service brokers providing value-added service offerings to a community of service users, who consume relevant services [6; 5]. In spite of the current success with the advancement of cloud computing, some challenges with the current monolithic model require an extension to the current stack. The monolithic model still imposes vendor lock-in such that services cannot be dynamically combined with other services from external third party sources to offer more value-adding functionalities to the users [7]. Papazoglou et al., [7] proposed the blueprinting of the cloud, a model that allows the syndication, configuration, and deployment of virtual service-based application in the cloud; thus generating interest in cloud ecosystem. However, the current state of cloud ecosystem does not support the ultimate vision of offering XaaS via an e-marketplace platform [8]. Some examples of existing cloud service e-marketplace are Saasmax[2], oracle e-marketplace[3], Google play store[4], AppExchange[5] etc. Although, these possess some features of an e-marketplace like product (service) directory etc., more sophisticated features that maximize the dynamism of service composition are still lacking.

Akolkar, et al [9] identified six enablers for the realization of the vision of a true electronic emporium of cloud-based services; one of which includes the possibility of formal or incidental service composition to derive more complex business solutions. Formal composition refers to the combination of one or more services into composite services, which are also offered to users on the marketplace; incidental composition is one time composition based on specific user request. Apart from the functional capabilities they provide, cloud services possess non-functional or quality of service (QoS) attributes (e.g. reliability, response time, cost, security, availability etc.), which are aggregated to determine the overall QoS attribute of the composite service, hence pay major role in service composition [10; 11; 12].

To this end, we envisage that the e-marketplace of the future provides an environment where atomic services interconnect in unprecedented ways to form composite services that fulfills complex business processes; so that cloud service users can then discover these services from the service directory consume and pay for these services [9; 11]. We hypothesize that an ecosystem-driven model can be applied to coordinate formal composition of atomic services to populate the e-marketplace cloud service directory. However, noting the unprofitability, impracticality and error-prone nature of performing service combination ad hoc, the concern addressed in this technical report is how to guide the combination of atomic services participating in an ecosystem in a seamless manner, considering the QoS of individual services and the constraints guiding the composition.

Meanwhile, some techniques in the domain of product configuration and product line engineering have emerged to effectively structure a hierarchical inter-relationship among components and guides the composition or configuration of these components based on specific

[2] www.saasmax.com
[3] www.oraclemarketpalce.com
[4] www.googleplay.com
[5] www.appezchange.com

constraints. Since some of these techniques have been applied to configure cloud services (e.g. [13] and [14]), this report explores how these techniques can be adapted to structure and guide the derivation of valid combinations of services in a cloud ecosystem. More specifically we proposed the use of feature models to model the inter-relationships and constraints among the atomic services, which is transformed into a constraint satisfaction problem and off-the-shelve constraint solvers are used to determining valid combinations. The collection of valid combinations become the blueprint that guides service composition and populates the e-marketplace service directory, from which users make service selection decisions. The applicability of the approach proposed in this report is demonstrated via an example of Customer relationship management as a service ecosystem.

The rest of this report is structured as follows: section 2 defines the concept of cloud ecosystems and cloud service e-marketplace contains. In section 3, the main techniques employed in the design of the approach (i.e. feature modeling and constraint satisfaction) were discussed; while the proposed approach was presented in section 4. The illustrative example to demonstrate the applicability is contained in Section 5. Section 6 reviewed related works and this report concludes in section 7 with future works.

II. ECOSYSTEM-DRIVEN CLOUD SERVICES E-MARKETPLACE

An electronic e-marketplace is a platform where the demand and supply for certain products or services are fulfilled using information and communication technologies [5;15; 16]. The cloud e-marketplace extends the concept of an electronic e-marketplace, and is an online platform that manages the distribution and trading of cloud-based services. On this platform, service providers enlist services with the purpose of integration with other services to form composite services; while users can discover, consume and pay for service offerings [7; 5; 17; 8; 9; 18; 19].

The success of cloud e-marketplace will depend on how effectively it will be able to instantiate and dynamically maintain computing platforms that meet arbitrarily varying service requirements of cloud service users, leveraging on integrators enabled by SOA and web services [20]. This is further enhanced by the e-marketplace providing a unified view of all available services and becomes a single point of access to composite services that results from participation in the ecosystem, and hides the complexity of the underlying interconnections among atomic services in the ecosystem [8]. Merriam-Webster defines an ecosystem as the complex of a community of organisms and its environment functioning as an ecological unit. In the context of cloud computing, cloud ecosystem describes the complex system of interdependent components that work together to enable cloud services. An ecosystem consists of interwoven mixture of infrastructure, platform and application contributing to increase their value as a collective than the value of the individual elements on its own. Building upon the traditional cloud computing stack (i.e. IaaS, PaaS and SaaS), anything/everything can be provisioned and consumed as a service. The cloud is growing collection of services, and has grown beyond just a platform to execute workloads, but limitless opportunities for possibilities that can service any aspect of business, such as compute, content delivery, storage, database management systems, specific SaaS.

The future of cloud computing would be fast-tracked by successful partnerships and collaborations among multiple service providers to tie services together, and enabling an environment where anything/everything as services are delivered to meet business needs via a marketplace environment [21]. In such case, the e-marketplace also provides mechanisms that support dynamic formal or incidental composition of services, enabling the reuse of services, pricing models and service level agreements [8; 7].

III. CONSTRAINT-BASED REASONING ON FEATURE MODEL

The multi-provider and multi-service dimension of cloud e-marketplace necessitates a means to capture the inter-relationships and constraints among the heterogeneous services in a single model. This model becomes the basis for a structured combinatorial guide that informs what is and what is not possible in the ecosystem, since all participating services are combined based on the constraints that underlay the hierarchical inter-relationships of services in delivering CRM solutions. The set of valid combinations share certain similarities with each other; while possessing specific differences, as a result of some functional aspects together with varying QoS values derived from aggregated QoS values of participating services.

In this technical report, we define a cloud service e-marketplace provider is the one who manages the ecosystem, and decides on the strategies to enhancing the value chain of the ecosystem. Enhancing the value inherent in the ecosystem entails identifying how and ensuring that services collaborate to deliver maximum value. However, to determine valid combinations of service in an ad hoc manner, would undermine the net value characteristic of ecosystems; more so, such ad hoc processing is error-prone and time consuming [22]. To adequate estimate the value of the ecosystem, first, there is need for a logical hierarchical arrangement of all the participating services into a knowledge model based on a specific combinatorial blueprint and, secondly, a means to automatically derive useful information by analysis of the logical hierarchy of these services. Furthermore, automating the analysis of the ecosystem knowledge model reveals a number of useful information about the ecosystem to address some the concerns of the e-marketplace provider. For example, the e-marketplace provider is interested in knowing how many valid combinations are available in the ecosystem. This information translates to the number of potential service offerings that can be provisioned via the e-marketplace. Potentially, this number can be very high depending on the number of collaborating services and the constraints on their inter-relationship. Knowing the number of potential solutions provides an informed basis for the e-marketplace provider to decide the range of products the e-marketplace would offer. Another concern is that, are there services that will not fully benefit from the value chain in the ecosystem (partly or fully due to their presence in a few or none of the likely combinations). Moreover, a structured model and automated analysis also offers some strategic benefit to service providers; as it were, service providers can analyze this information to estimate the profitability of their offerings, and position their offerings for better competiveness in the ecosystem.

Since the structure of cloud ecosystem is analogous to the fundamental principles of software product line engineering (SPLE) and product configuration (PC) domain [23; 24], variability modeling techniques, like feature modeling, used in the SPLE and PC is proposed and adopted in this report to effectively structure the hierarchical inter-relationships among ecosystem services. The PC domain is concerned with the ability to mass-customize products targeted at specific requests and/or user segments, which is a crucial determinant of reducing lead time, and increase business process efficiency in mass-manufacturing [25]. Mass-customization techniques have been applied to concrete products (e.g., bicycles[6], and baby strollers[7]) as well as insubstantial products like software and services (e.g. insurance, tourism etc.). Configuration software is employed to adapt products or services to suit specific requirements by combining components characterized by specified attributes, based on the constraints that underlay valid combinations [26]. The features in this report refer to the participating services being combined in accordance to a prescribed blueprint. The cornerstone of performing product configuration and deriving software instance from a SPL are the knowledge representation of product/software features based on variabilities and commonalities and computer-aided reasoning techniques employed to support both PC and SPL process. Such representation can be achieved using feature models.

A. Feature Modeling

Feature model is a graphical representation of common aspects and differences in a collection of products in a product-line and is used to structure and constrain the product options. A feature is defined as the end-users' understanding of the capabilities of systems in the domain [24]. A feature model is a hierarchically arranged collection of features and consists of the inter-relationships between a parent feature and its child features, and a set of cross–tree constraints that define the criteria for feature inclusion or exclusion. A feature model represents in a single model, all possible alternatives that the scope of the feature model covers. Each solution is a valid instantiation of the feature model. In this report, we define and abstract each participating ecosystem service as a feature in the feature model, and the range of possible solutions that is obtainable from the ecosystem is defined by the feature model. Cross-tree constraints provide a legal basis of how services and their QoS attributes can be legally combined. Benavides, et al., [22], identified three main types of feature-based models: basic, cardinality-based and extended feature models. Basic feature model (also known as the FODA feature model) was introduced by Kang, et al., [27] and it describes three feature types (Mandatory, Optional, and Alternative) and two cross-tree constraints (Requires and Excludes). A mandatory feature is a feature that must be included in a product, while an optional feature is a feature that may or may not be included in a feature. Given a set of features from which only one feature is selected to be included in a product is called an alternative feature. Required and Excludes cross–tree constraints in basic feature model are defined as follows: given features X and Y; X requires Y is defined as if X is included in a product, then Y should also be included, but not vice-versa. X excludes Y means that if X is included then Y should not be included, and vice-versa. The inadequacy of alterative relationship

[6] http://www.bikeconfig.com/
[7] https://www.bugaboo.com

to model situations with multiple child features motivated cardinality-based feature model, in which numbers are introduced to denote the multiplicities of the set of features of basic feature model. A cardinality-based feature model is a hierarchal collection of features, with each feature having cardinality [28]; that is to say; the number of times copies of a feature is included in a product is determined by its cardinality.

Table 1. CONCEPTS of Extended Feature Model

EFM Concept	Description
Feature	A functional characteristic of a service or an increment in product functionality. E.g. an SMS notification cloud service, or an email cloud service
Attribute	Any measurable characteristic of a feature that can be measured. For Example, reliability is a cloud service QoS attribute.
Attribute domain	The attribute domain specifies the range of values that an attribute can assume; i.e. qualitative or quantitative (discrete and continuous) values corresponding to the QoS attributes.
Attribute value	Attribute values define the actual value that belongs to a particular domain. The attributes values are usually an aggregation of all the values of corresponding features of the final product. For example, the cost of a service aggregates all the cost of the features included in a product.

Although basic feature model can be used to provide a basis for automated configuration of actual products, there is need to sometimes include in the feature model quality information about features (such as non-functional attributes). In extended feature models, feature model are annotated with quality information, analysis could use these qualities as basis in specifying valid combination. In classic SPL domain, the concepts that describe EFM are presented in Table 1. Extended feature models are desirable for modeling cloud ecosystem, so as to capture cloud services, their QoS attributes, inter-relationship and constraints, which is vital to the generation of valid combinations to populate the e-marketplace service directory.

B. Automated Analysis of feature model using Constraint Satisfaction

Deriving useful information from the ecosystem model requires automated mechanism that is able to reason on and analyze the model upon which the service inter-relationship is built [22; 29; 30]. Some studies on automated analysis on feature models have been conducted (e.g. [22]) and a number of analysis operations have been proposed. Automated analysis of feature models uses computer-aided tools to extract important information from feature models [31; 22]. The automated approach entails transforming the feature models into a specific formal logic-based representation, which becomes inputs to solvers, and analysis operations are performed to obtain useful information. A solver is a software package that accepts formal representations as inputs and determines some satisfiability criteria [22], e.g. Choco [32]. Approaches that can transform the feature model into formal representations have been classified into: Description Logic, Propositional Logic, and Constraint Programming (cf. [22]). In this repoet, we have employed Constraint programming, as a method that uses constraints as a programming method to encode and solve constraint satisfaction problems. The mapping from a feature model to a particular CSP solver is less straightforward than with Propositional Logic because the encoding structure is solver dependent. Formally, CSP is fined as:

Definition1 (CSP): A constraint satisfaction problem (CSP) is defined as a finite set of variables, each of which is associated with a finite domain, and a set of constraints that restricts the values the variables can simultaneously take.

Feature models are encoded as CSP model and CSP solvers use constraint programming to find assignment for each variable that satisfies the constraints [22]. However, the following steps apply in encoding feature models as CSP [22]:

Step 1: Each feature of the feature model maps to a variable of the CSP with a domain of $[0..1]$ (i.e. true or false), depending on the kind of variable supported by the solver.

Step 2: Each relationship of the Model is mapped into a constraint depending on the type of relationship.

Step 3: The resulting CSP is the one defined by the variables of step 1 and the corresponding domains and constraints that is the conjunction of all precedent constraints plus additional constraint assigning true to the variable that represent the root, depending on the variable's domain.

The rules encoding feature model as CSP are presented in Table 2.

1) *Automated Analysis Operations on Feature Model*

After encoding model into a formal logic-based representation, mathematical operations based on the semantic of the underlying logic-representation can be performed to derive useful information about the model. A number of analysis operations exist, but the following analysis operations are relevant to the objectives set out in this report: *Determine the Satisfiability of a model; solutions count; generate all the valid solutions.*

a) *Determine the Satisfiability of a model*

This operation examines the feature model and determines returns a verdict that determines the satisfiability of the feature model, by telling if the feature model is void or not. A feature model is said to be satisfiable, when at least one valid combination, can be derived from it. In other words, a feature model is not void if it represents at least one solution.

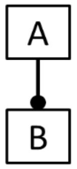
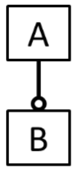
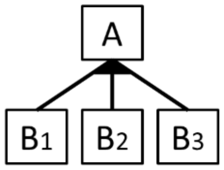
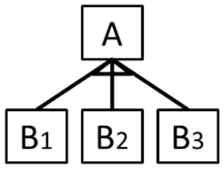
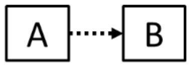
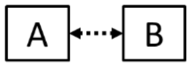
b) *Products count*

This operation returns the number of valid combinations that can be derived from the feature model. This also relates to the satisfiability operation, such that if the count is zero, then the feature model is void. The e-marketplace provider can estimate at every point the number of services that is offer-able on the e-marketplace.

c) *Generate all the valid products*

This operation generates all valid combinations in the feature model that satisfies all the constraints in their inter-relationship. In the context of this research, the set of valid combinations forms the set of alternatives indexed in the service directory from which the user selects a composite service that approximates that user's requirements.

Table 2. Feature Model to Constraint Programming Mapping

Relationships in CEFM	CSP Mapping
 Mandatory	$A = B$
 Optional	$if\,(A = 0)$ $B = 0$
 OR	$if\,(A > 0)$ $Sum\ (B_1, B_2 \ldots B_n) in\ (1 \ldots n)$ $else$ $B1 = 0, B2 = 0 \ldots B_n = 0$
 Alternative	$if\,(A > 0)$ $Sum\ (B_1, B_2 \ldots B_n) in\ (1 \ldots 1)$ $else$ $B1 = 0, B2 = 0 \ldots B_n = 0$
 Requires	$if\,(A > 0)$ $B > 0$
 Excludes	$if\,(A > 0)$ $B = 0$

2) ***QoS Aggregation Functions***

At least more than one service are composed in a valid combination as contained in the service directory, therefore the QoS properties of the constituent services are aggregated to determine the overall QoS values for the valid combination. Usually, the overall QoS properties of such composite services, conceptualized into a business process, are determined by the QoS attributes of constituent services and their composition relationships. There are four basic composition patterns that inform the arrangement of constituent services in a business process [33; 34; 35; 36]. They include: 1) **Sequential**: A sequence pattern an activity (or services) in the business process executes after another activities has concluded execution. In order words, the services are executed one after the other. 2) **Parallel**- In a parallel pattern, all the branches are executed at the same time. 3) **Conditional (or branch):** Only one branch, with a set of activities is selected for execution in the branch pattern. 4) **Loop:** In a loop pattern, an activity in the business process is executed for $(n > 0)$ times.

However, the sequential composition pattern is assumed in this report. Sequential pattern is the fundamental pattern, as the other patterns (i.e. parallel, conditional and loop), can be may be reduced or converted to the sequential pattern [36; 37]. Based on the nature of QoS attribute, different aggregation functions can be applied [36]. However, in this report considers two types of aggregation functions; summation and multiplication (cf. Table 3): 1) **Summation**: In summation aggregation function, the values of a QoS attributes are summed up (e.g. cost and response time). For cost, the overall *cost* for a valid combination service should be a summative total of the cost of all constituent services. 2) **Multiplication**: Multiplication function implies that the aggregate is a product of all the values of a QoS attribute of all the constituent services (e.g. availability).

Definition (QoS Aggregation): Let a service $s \in S$ be a valid combination composed of t number of distinct services $Z_{(1\ to\ t)}$ with n QoS attributes and acts sequentially. Let $q_i(Z_k)$ be the value of the i^{th} QoS attribute for the k^{th} distinct service. Such that the aggregated value i^{th} QoS attributes for all distinct services composed in s is given as:

$$q_i(s) = (q_i(Z_1) \bowtie q_i(Z_2) \bowtie \cdots \bowtie q_i(Z_t)) \qquad (1)$$

Where $\bowtie$ represents the aggregation operator based on the aggregation function employed with respect to the QoS type, and $t > 1$. Meanwhile, the vector Q of QoS values for a valid combination s is given as:

$$Q(s) = (q_1(s), q_2(s) \ldots q_n(s)) \qquad (2)$$

The QoS aggregation rules for the four QoS properties considered in this case study (i.e. cost, response time, availability and security) are given in Table 3. The availability and reliability aggregate functions are non-linear functions. In order to make all aggregate functions to be linear ones, we transform them by using the logarithmic function (see Eqn. 3) [38].

$$\log\ (q_i(s)) = \log\left(\prod_{j=1}^{t} q_i(Z_j)\right) = \sum_{j=1}^{t} \log(q_i(s)) \qquad (3)$$

IV. THE PROPOSED SOLUTION APPROACH

The proposed approach to populating the ecosystem-driven service directory of cloud service e-marketplace consist of modeling the ecosystem of atomic cloud services using extended feature models, transforming the model into a constraint programming model and reasoning on the logic-based model using constraint solver (see Figure 1). Details are presented in subsequently.

Table 3. Summary of Aggregation functions

Aggregation Type	QoS Attribute	Aggregation Function
Summation	Cost	$q_i(s) = \sum_{j=1}^{t} q_i(Z_j)$
	Response Time	
Multiplication	Availability	$q_i(s) = \prod_{j=1}^{t} q_i(Z_j)$
	Reliability	

A. Modeling the Cloud service Ecosystem

We modeled the cloud service ecosystem by adopting feature models, which we called Cloud Ecosystem Feature Model (CEFM). More specifically, an extended feature model was employed due to its modeling flexibility that captures the QoS attributes, and the constraints that exist among them.

B. Reasoning Engine

The CEFM was transformed into a formal representation based on CSP, and we employed he general purpose constraint solver, Choco, to perform automated analysis of the model to derive useful information that is beneficial to both e-marketplace provider and service providers. The solver determines the satisfiability of the CSP, and if a CSP is satisfiable, then solutions can be obtained. The solver searches for a solution in a CSP, using search algorithms to generate all the possible combinations of values for each variable in the CSP and certifies that they correspond to a solution of the CSP. Table 2 shows the rule for mapping constructs in the CEFM into CSP. The corresponding CSP representation of the CSEM is read by the reasoning engine, and performs automated analysis on the CSP representation to generate all valid service combinations. Furthermore, we determined the overall quality performance of valid combinations, by considering the QoS factors of constituent services and their impact of the overall QoS of the valid combinations. This was performed by aggregated the QoS of each atomic service using QoS aggregation functions in Table 3.

C. Service Directory

The service directory indexes all the QoS information about the collection of valid combination services generated by the all products operations on the CEFM. The service directory is modeled as case base of valid combinations and their QoS information.

Figure 2, depicts the process of organizing ecosystem information into a model for obtaining useful information pertinent to operationalizing the cloud service e-marketplace.

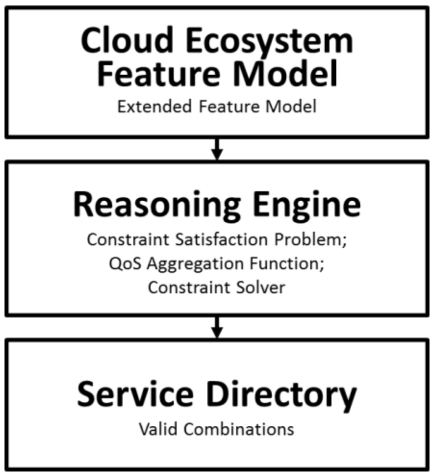


Fig. 1. Process Architecture of the Proposed Approach

V. Illustrative Example

The approach proposed in this report is validated using an hypothetical Customer Relationship Management as a Service (CRMaaS). CRMaaS is enabled by a cloud ecosystem of CRM components services for Small and Medium Enterprises (SME) delivered through the cloud e-marketplace. A SME that requires a complete cloud-based CRM solution for managing its user relationship processes in a bit to improve business relationship and increase the bottom-line can find the most appropriate solution via the e-marketplace. An instance of the CRMaaS offering is a combination of any/all of these services to create a complete CRM solution. On the e-marketplace, multiple variants of CRMaaS solutions exist and are differentiated by QoS factors. An SME can then search for and consume CRM solution that aligns with their specific aspiration and preferences. Therefore, the e-marketplace service directory contains a set of *m* CRM solutions that can be evaluated along *n* decision criteria with respect to an SME's preferences. Having expressed requirements, which is converted to a search query, the e-marketplace facilities generates search results in form ranking of complete CRM solutions that approximates the requirements expressed.

The components that make up the CRMaaS ecosystem includes: Contact management, Database, Marketing and Social-media analysis (See Figure 2). The CRMaaS solution is realized by the participation of various service providers in the ecosystem. One or many providers can contribute one or more of the following range of services to the ecosystem with differentiated QoS factors. The description of each module is as follows:

Contact Management Service**:** tool to manage user contacts and communication; including appointment management, task management and scheduling, communication (SMS, email),

Cloud Database**:** cloud-based RDBMS to store user information including user personal data, purchase history, preferences etc.

Marketing Service: Tools for communicating with users; including email marketing, text message marketing, social media marketing etc.

Social Media Analytics: Tool that monitors conversations on social media and analyze feedbacks, capturing user sentiments.

Cloud Platform: The derived valid combinations would require a cloud platform on which to run.

First we identified all the constituent services that can fulfill each component, together with the values of the QoS attributes. The QoS attributes considered in this example includes: availability and reliability, measured in percentages (%); response time measured in milliseconds (ms), while cost is measured in Dollars/month ($/Month). The number of candidate services for each CRMaaS component is given as follows (see Table 4): Contact management (CM_1, CM_2, CM_3,CM_4); Cloud Database (CD_1, CD_2,CD_3); Marketing (M_1, M_2); Social Media Analysis (SMA_1, SMA_2, SMA_3); Platform (P_1, P_2).

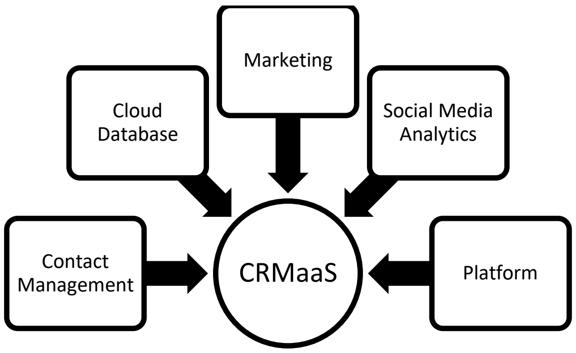


Fig. 2. High-level Structure of the components of a CRMaaS

Table 4. Candidate cloud services to realize CRMaaS Components

CRMaaS Components	**Candidate Services**	**QoS Values**			
		Avail.	***Resp. Time***	***Reliability***	***Cost***
Contact Management	CM1	90	--	90	30.50
	CM2	95	--	67	29.99
	CM3	70	--	40	25.50
	CM4	99	--	79	34.99
Cloud Database	CD1	89	100.22	60	13.50
	CD2	79	50.54	75	20.50
	CD3	97	120.34	80	50.00
Marketing	M1	99	--		55.50
	M2	91	--		59.99
Social Media Analysis	SMA1	90	200.45	88	49.99
	SMA2	95	138.56	90	50.00
	SMA3	85	125.45	79	45.67
Platform	P1	99	300.45	70	199.99
	P2	99	423.10	75	149.99

The rules guiding the combination of these candidate services are contained in Table 5, while the CEFM that models the relationships and constraints is presented in Figure 3. All CRMaaS components are mandatory; however, each candidate service is an alternative to other candidate services within the same component group.

Table 5. Require and Exclude Constraints on Candidate service combination

Service1	**Constraint**	**Service2**
CM1	Requires	P1
CM1	Requires	CD1
CM2	Excludes	M1
SMA1	Requires	CD2
CD2	Excludes	P2
SMA2	Requires	M1
SMA3	Excludes	CD2

The encoding of the CEFM as CSP, together with the aggregation functions were implemented using Java in NetBeans 8.1 based on the constraints provided in the Choco library; Choco solver was used as the constraint solver to derive valid combinations from the CEFM. The analysis operation performed to generate all products from the CEFM yielded a total 38 valid combinations (See Table 6), including the constituent atomic services, and the aggregated values for each QoS attributes. The generated composite services are then indexed as the services contained in the cloud service e-marketplace service directory. The indexed list becomes the catalogue from which users are served recommendations with respect to their QoS requirements.

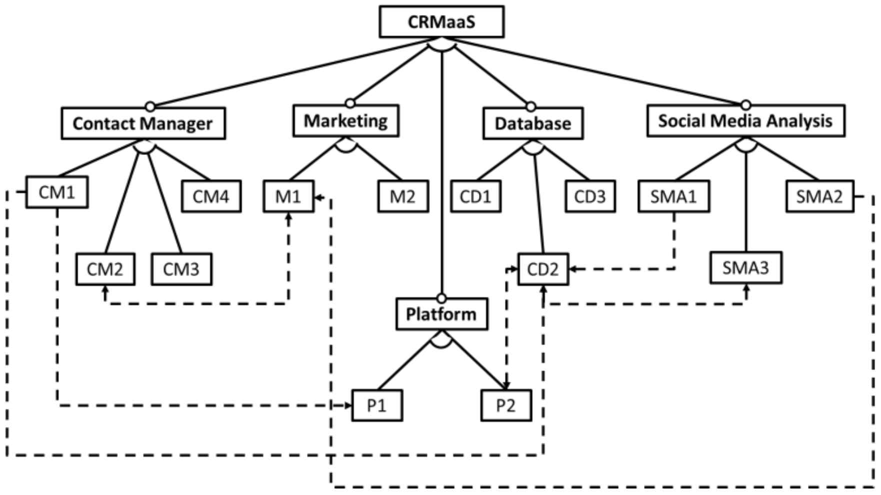


Fig. 3. CRMaaS Cloud Ecosystem Feature Model

VI. RELATED WORKS

Previous works have proposed the use of feature models to capture the variabilities of cloud services and applied automated means generate valid cloud service offerings. A SPL-based approach for cloud service selection that employs feature models, extended with cardinalities and attributes, to describe the variability in cloud environments has been proposed in [39]. The approach utilizes a domain model to support the consistent configuration of complete stack of cloud services that complies with user's functional and quality requirements and automates the deployment of such configurations by generating executable deployment scripts. Feature models provide the template for how artifacts are to be combined to yield a complete software product that satisfies a set of defined constraints. A tool support was developed based on Constraint Satisfaction, as part of an earlier SALOON framework [13] to demonstrate plausibility of this approach. Meanwhile, the limitation imposed by using a given cloud service and the benefit inherent in using several cloud platforms to deploy multi-cloud applications necessitate approaches that can handle the intrinsic variabilities among heterogeneous cloud service providers. SALOON [13] is a model-driven Ontology-based approach founded on feature model, to handle the variability in cloud services while managing the derivation of specific cloud configurations. Ontology was employed to model the semantics underlying the description of a variety of cloud systems. SALOON is proposed as a solution that can assist in deploying multi-cloud application, particularly when one provider is incapable meeting all application requirement rather than doing so in an ad hoc manner. The SALOON framework is extensible by adding new feature models that conforms with the originating SALOON-based feature model meta-model. Cloud services are modeled as features, and selected features are transformed into propositional logic and constraints, and SAT solvers (e.g. Sat4j) are used to confirm the validity of the configuration.

Table 6. List Valid combinations based on CRMaaS Cloud Ecosystem Model

CRM_ID	**Constituents Services**	**Aggregated QoS Values**			
		Availability	***Response Time***	***Reliability***	***Cost***
CRM_1	CM4 ; CD3 ; SMA3 ; M2 ; P2	98.68	668.89	75.73	340.64
CRM_2	CM3 ; CD3 ; SMA3 ; M2 ; P2	97.16	668.89	72.78	331.15
CRM_3	CM4 ; CD3 ; SMA3 ; M2 ; P1	98.67	546.24	75.43	390.64
CRM_4	CM3 ; CD3 ; SMA3 ; M2 ; P1	97.16	546.24	72.48	381.15
CRM_5	CM4 ; CD1 ; SMA3 ; M2 ; P2	98.29	648.77	74.48	304.14
CRM_6	CM3 ; CD1 ; SMA3 ; M2 ; P2	96.79	648.77	71.53	294.65
CRM_7	CM4 ; CD1 ; SMA3 ; M2 ; P1	98.29	526.12	74.19	354.14
CRM_8	CM3 ; CD1 ; SMA3 ; M2 ; P1	96.79	526.12	71.23	344.65
CRM_9	CM2 ; CD3 ; SMA3 ; M2 ; P2	98.49	668.89	75.02	335.64
CRM_10	CM2 ; CD3 ; SMA3 ; M2 ; P1	98.49	546.24	74.72	385.64
CRM_11	CM2 ; CD1 ; SMA3 ; M2 ; P2	98.11	648.77	73.77	299.14
CRM_12	CM2 ; CD1 ; SMA3 ; M2 ; P1	98.11	526.12	73.47	349.14
CRM_13	CM4 ; CD3 ; SMA3 ; M1 ; P2	99.03	668.89	75.73	336.15
CRM_14	CM3 ; CD3 ; SMA3 ; M1 ; P2	97.53	668.89	72.78	326.66
CRM_15	CM4 ; CD3 ; SMA2 ; M1 ; P2	99.51	682	76.3	340.48
CRM_16	CM3 ; CD3 ; SMA2 ; M1 ; P2	98.01	682	73.34	330.99
CRM_17	CM4 ; CD3 ; SMA3 ; M1 ; P1	99.03	546.24	75.43	386.15
CRM_18	CM3 ; CD3 ; SMA3 ; M1 ; P1	97.53	546.24	72.48	376.66
CRM_19	CM4 ; CD3 ; SMA2 ; M1 ; P1	99.51	559.35	76	390.48
CRM_20	CM3 ; CD3 ; SMA2 ; M1 ; P1	98.01	559.35	73.04	380.99
CRM_21	CM4 ; CD1 ; SMA3 ; M1 ; P2	98.66	648.77	74.48	299.65
CRM_22	CM3 ; CD1 ; SMA3 ; M1 ; P2	97.15	648.77	71.53	290.16
CRM_23	CM4 ; CD1 ; SMA2 ; M1 ; P2	99.14	661.88	75.05	303.98
CRM_24	CM3 ; CD1 ; SMA2 ; M1 ; P2	97.63	661.88	72.1	294.49
CRM_25	CM4 ; CD1 ; SMA3 ; M1 ; P1	98.66	526.12	74.19	349.65
CRM_26	CM3 ; CD1 ; SMA3 ; M1 ; P1	97.15	526.12	71.23	340.16
CRM_27	CM4 ; CD1 ; SMA2 ; M1 ; P1	99.14	539.23	74.75	353.98
CRM_28	CM3 ; CD1 ; SMA2 ; M1 ; P1	97.63	539.23	71.8	344.49
CRM_29	CM1 ; CD1 ; SMA3 ; M2 ; P1	97.88	526.12	74.75	349.65
CRM_30	CM1 ; CD1 ; SMA3 ; M1 ; P1	98.24	526.12	74.75	345.16
CRM_31	CM1 ; CD1 ; SMA2 ; M1 ; P1	98.73	539.23	75.32	349.49
CRM_32	CM4 ; CD2 ; SMA1 ; M2 ; P1	98.02	551.35	75.62	360.46
CRM_33	CM3 ; CD2 ; SMA1 ; M2 ; P1	96.52	551.35	72.67	350.97
CRM_34	CM2 ; CD2 ; SMA1 ; M2 ; P1	97.84	551.35	74.91	355.46
CRM_35	CM4 ; CD2 ; SMA2 ; M1 ; P1	98.62	489.46	75.72	360.98
CRM_36	CM3 ; CD2 ; SMA2 ; M1 ; P1	97.12	489.46	72.76	351.49
CRM_37	CM4 ; CD2 ; SMA1 ; M1 ; P1	98.39	551.35	75.62	355.97
CRM_38	CM3 ; CD2 ; SMA1 ; M1 ; P1	96.88	551.35	72.67	346.48

In the same line, Wittern, et al., [14] argues that the increase in cloud services provide the need for a means to capture the variety of capabilities, and asserts that many cloud service section approaches assume the underlying representation of the cloud service capabilities which should serve as input to the selection process. Therefore, authors [14] presented an approach to harness cloud service capabilities using variability model. The variability models serve as representation mechanisms and are called *Cloud Feature Models* (CFMs). CFMs are used to elicit requirements and to perform filtering operation within a process the authors referred to as cloud service selection process (CSSP). The CSSP prunes the list of likely candidates based on user's requirements and these candidates (called *Alternative models*) are deployable valid cloud configurations.

Also to manage the variability among cloud-based applications with support for multiple stakeholders, authors in [40] applied extended feature modeling to configure cloud-based multi-tenant aware applications, by using the model to express the variability in functionality and QoS attributes. The proposed approach manages dynamic configuration that involves an adaptive staged configuration process capable of adding and removing providers or users dynamically from the cloud-platform and that allows for reconfiguration of variant services as user's provider's requirements changes.

In these approaches, users are expected to painstakingly configure cloud services, with the assumption that all users are full domain experts. However, a cloud service e-marketplace should among others, provide a real online shopping experience similar to exiting ecommerce platforms [9; 5], where available service offerings indexed in the e-marketplace service directory, more like a catalogue, and seamlessly updated in a manner completely transparent to the users. The user is shielded from the underlying complexity of performing service configuration, and since all possible alternatives is pre-determined (formal service composition [9]), the users are able to explore other alternatives with respect to their requirements. Furthermore, the ecosystem model should be scalable to accommodate new services, and that the decision making process is able to use this service information representation in a manner that is seamless and natural to an ecommerce platform, with little or no disruption to marketplace operations. The approached proposed in this report automatically includes scenarios of new entrants and exists of services into and from the ecosystem. With each case of entrants or exists based on the stated entrance and exit policies of the e-marketplace, the feature model is altered; and a seamless automated update of the e-marketplace service directory can be still achieved. This presupposes that service registration and disengagement from the ecosystem is performed offline, not at request time, giving this approach the scalability advantages in the event of multiple concurrent users of the e-marketplace.

VII. CONCLUSION

A cloud marketplace is an ecosystem of heterogeneous services from multiple providers. The different ways in which these services are aggregated creates a plethora of potential offerings with varied QoS factors that can meet diverse business needs of users. In this technical report, we proposed a constraint-based reasoning on extended feature models to address the need to explicitly capture the cloud services, their QoS attributes, and the cross-service relationships and constraints in a logical and structural manner as part of an ecosystem. We used this model to determine blueprints to consistently generate valid compositions. With the aid of an example, we demonstrated how the service directory is constantly updated with composite services from the ecosystem, and those services can then be offered to users via the e-marketplace platform. Since CSP solvers have the ability to analyze numeric or text-like attributes, the proposed approach will be improved to cater for qualitative QoS attributes like security, user-friendliness and eco-friendliness whose values are qualifier tags. Our goal is to improve the user experience of the cloud service e-marketplace environment in the near future.

REFERENCES

[1] Rimal, B. P., Jukan, A., Katsaros, D., & Goeleven, Y. (2011). Architectural Requirements for Cloud Computing systems: An Enterprise Cloud Approach. Journal of Grid Computing , 9 (1), 3-26. DOI: 10.1007/s10723-010-9171-y

[2] Qaisar, E. J. (2012). Introduction to Cloud Computing for Developers-Key concepts, the players and their offerings. 2012 IEEE TCF Information Technology Professional Conference. IEEE. DOI: 10.1109/TCFProIT.2012.6221131

[3] Al-Shammari, S., & Al-Yasiri, A. (2014). Defining a Metric for Measuring QoE of SaaS Cloud Computing., (pp. 251-256).

[4] Buyya, R., Yeo, C. S., & Venugopal, S. (2008). Market-oriented cloud computing. Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC'08) (pp. 5-13). IEEE. DOI: 10.1109/HPCC.2008.172

[5] Menychtas, A., Vogel, J., Giessmann, A., Gatzioura, A., Garcia Gomez, S., Moulos, V., et al. (2014). 4CaaSt marketplace: An advanced business environment for trading cloud services. Future Generation Computer Systems , 104–120. DOI: 10.1016/j.future.2014.02.020

[6] Bosch, J., & Bosch-Sijtsema, P. (2010). From integration to composition: On the impact of software product lines, global development and ecosystems. Journal of Systems and Software , 67-76. DOI: 10.1016/j.jss.2009.06.051

[7] Papazoglou, M., & van den Heuvel, W.-J. (2011). Blueprinting the cloud. IEEE Internet Computing , 74-79. DOI: 10.1109/MIC.2011.147

[8] Gatzioura, A., Menychtas, A., Moulos, V., & Varvarigou, T. (2012). Incorporating Business Intelligence in Cloud Marketplaces. IEEE 10th International Symposium on Parallel and Distributed Processing with Applications (ISPA) (pp. 466-472). IEEE.

[9] Akolkar, R., Chefalas, T., Laredo, J., Peng, C.-S., Sailer, A., Schaffa, F., et al. (2012). The Future of Service Marketplaces in the Cloud. IEEE Eighth World Congress on Services (SERVICES) (pp. 262-269). IEEE.

[10] Chen, X., Zheng, Z., Liu, X., Huang, Z., & Sun, H. (2013). Personalized QoS-Aware Web Service Recommendation and Visualization. IEEE Transactions on Services Computing , 35-47.

[11] Barros, A. P., & Dumas, M. (2006). The rise of Web service ecosystem. IT Professional , 8 (5), 31-37.

[12] Garg, S. K., Versteeg, S., & Buyya, R. (2011). SMICloud: A Framework for Comparing and Ranking Cloud Services. 2011 Fourth IEEE International Conference on Utility and Cloud Computing (UCC) (pp. 210-218). IEEE.

[13] Quinton, C., Haderer, N., Rouvoy, R., & Duchien, L. (2013). Towards Multi-Cloud Configurations Using Feature Models and Ontologies. Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds (pp. 21-26). ACM.

[14] Wittern, E., Kuhlenkamp, J., & Menzel, M. (2012). Cloud Service Selection Based on Variability Modeling. Service-Oriented Computing , 127-141.

[15] Bakos, Y. (1998). The emerging role of electronic marketplaces on the Internet. Communications of the ACM , 41 (8), 35-42.

[16] Akingbesote, A., Adigun, M., Jembere, E., Othman, M., & Ajayi, I. (2014). Determination of optimal service level in cloud e-marketplaces based on service offering delay. International Conference on

Computer, Communications, and Control Technology (I4CT) (pp. 283-288). Langkawi, Kedah, Malaysia : IEEE.

[17] Javed, B., Bloodsworth, P., Rasool, R. U., Munir, K., & Rana, O. (2016). Cloud Market Maker: An automated dynamic pricing marketplace for cloud users. Future Generation Computer Systems , 52-67.

[18] Vigne, R., Mach, W., & Schikuta, E. (2013). Towards a smart webservice marketplace. IEEE 15th Conference on Business Informatics (CBI) (pp. 208-215). IEEE.

[19] Schulz-Hofen, J. (2007). Web Service Middleware - An Infrastructure For Near Future Real Life Web Service Ecosystems. IEEE International Conference on Service-Oriented Computing and Applications. IEEE.

[20] Cavalcante, E., Batista, T., Lopes, F., Rodriguez, N., de Moura, A. L., Delicato, F. C., et al. (2012). Optimizing Services Selection in a Cloud Multiplatform Scenario. IEEE Latin America Conference on Cloud Computing and Communications (LATINCLOUD) (pp. 31-36). IEEE.

[21] Baek, S., Kim, K., & Altmann, J. (2014). Role of Platform Providers in Service Networks: The Case of Salesforce. com App Exchange. IEEE 16th Conference on Business Informatics (CBI) (pp. 39-45). IEEE.

[22] Benavides, D., Segura, S., & Ruiz-Cortes, A. (2010). Automated analysis of feature models 20 years later: A literature review. Information Systems , 615-636.

[23] Hubaux, A., Jannach, D., Drescher, C., Murta, L., Mannisto, T., Czarnecki, K., et al. (2012). Unifying software and product configuration: A research roadmap. Proceedings of the Workshop on Configuration (ConfWS'12), (pp. 31-35). Montpellier, France.

[24] Berger, T., Pfeiffer, R.-H., Tartler, R., Dienst, S., Czarnecki, K., Wasowski, A., et al. (2014). Variability Mechanisms in Software Ecosystems. Information and Software Technology , 56 (11), 1520-1535.

[25] Haug, A., Hvam, L., & Mortensen, N. H. (2011). The impact of product configurators on lead times in engineering-oriented companies. Artificial Intelligence for Engineering Design, Analysis and Manufacturing , 197-206.

[26] Hvam, L., Henrik Mortensen, N., & Riis, J. (2008). Product Customization. Springer Science & Business Media.

[27] Kang, K., Cohen, S., Hess, J., Novak, W., & Peterson, S. (1990, November). Feature–Oriented Domain Analysis (FODA) Feasibility. Technical Report CMU/SEI-90-TR-21 . Software Engineering Institute, Carnegie Mellon University.

[28] Czarnecki, K., Helsen, S., & Eisenecker, U. (2005). Formalizing cardinality-based feature models and their specialization. Software Process: Improvement and Practice , 10 (1), 7-29.

[29] Karataş, A. S., Oğuztüzün, H., & Doğru, A. (2012). From extended feature models to constraint logic programming. Science of Computer Programming , In-Press.

[30] Elfaki, A. O., Abouabdalla, O. A., Fong, S. L., Johar, M. G., Aik, K. L., & Bachok, R. (2012). Review and Future Directions Of The Automated Validation In Software Product Line Engineering. Journal of Theoretical and Applied Information Technology , 75-93.

[31] Batory, D., Benavides, D., & Ruiz-Cortes, A. (2006, December). Automated analysis of feature models: challenges ahead. Communications of the ACM , pp. 45-47.

[32] Jussien, N., Rochart, G., & Lorca, X. (2008). Choco: an open source java constraint programming library. CPAIOR'08 Workshop on Open-Source Software for Integer and Contraint Programming (OSSICP'08, (pp. 1-10).

[33] Mohabbati, B., Gašević, D., Hatala, M., Asadi, M., Bagheri, E., & Bošković, M. (2011). A Quality Aggregation Model for Service-Oriented Software Product Lines Based on Variability and Composition Patterns. Service-Oriented Computing , 436-451.

[34] Bouanaka, M. A., & Zarour, N. (2013). An approach for an optimized web service selection based on skyline. International Journal of Computer Science Issues , 10 (1), 412-418.

[35] He, Q., Han, J., Yang, Y., Grundy, J., & Jin, H. (2012). QoS-Driven Service Selection for Multi-Tenant SaaS. IEEE 5th international conference on Cloud computing (cloud) (pp. 566-573). IEEE.

[36] Yu, T., & Lin, K.-J. (2005). Service Selection Algorithms for Composing Complex Services with Multiple QoS Constraints. Proceedings of the International Conference on Service-Oriented Computing-ICSOC 2005, (pp. 130–143).

[37] Alrifai, M., Skoutas, D., & Risse, T. (2010). Selecting skyline services for QoS-based web service composition. Proceedings of the 19th international conference on World wide web (pp. 11-20). ACM.

[38] Li, J., Zheng, X.-L., Chen, S.-T., Song, W.-W., & Chen, D.-r. (2014). An efficient and reliable approach for quality-of-service-aware service composition. Information Sciences , 269, 238-254.

[39] Quinton, ,. C., Romero, D., & Duchien, L. (2014). Automated Selection and Configuration of Cloud Environments Using Software Product Lines Principles. IEEE 7th International Conference on Cloud Computing (CLOUD) (pp. 144-151). IEEE.

[40] Schroeter, J., Mucha, P., Muth, M., Jugel, K., & Lochau, M. (2012). Dynamic Configuration Management of Cloud-based Applications. Proceedings of the 16th International Software Product Line Conference-Volume 2, (pp. 171-178).