A New Architecture Description Language for Social Machines

Leandro M. Nascimento[1,2], Vanilson A. A. Burégio[1], Vinicius C. Garcia[1], Silvio R. L. Meira[1]

[1] Informatics Center - Federal University of Pernambuco (UFPE)
[2] Department of Informatics - Federal Rural University of Pernambuco (UFRPE)
Recife – Pernambuco – Brazil

{lmn2, vaab, vcg, srlm}@cin.ufpe.br

ABSTRACT

The term 'Social Machine' (SM) has been commonly used as a synonym for what is known as the programmable web or web 3.0. Some definitions of a Social Machine have already been provided and they basically support the notion of *relationships* between distributed entities. The type of relationship molds which services would be provided or required by each machine, and under certain complex constraints. In order to deal with the complexity of this emerging web, we present a language that can describe networks of Social Machines, named SMADL – the Social Machine Architecture Description Language. In few words, SMADL is as a relationship-driven language which can be used to describe the interactions between any number of machines in a multitude of ways, as a means to represent real machines interacting in the real web, such as, Twitter running on top of Amazon AWS or mash-ups built upon Google Maps, and obviously, as a means to represent interactions with other social machines too.

Categories and Subject Descriptors

D.2.11 [**Software Architectures**]: Languages, Domain-specific architectures

General Terms

Languages, Design, Algorithms.

Keywords

Architecture description language, programmable web, web 3.0, social machines.

1. INTRODUCTION

We are entering the web 3.0 phase. Also known as the programmable web, it is the networked space-time where innovation lies on the power of developing software for the web, through the web, and in the web, using the web as both programming and deployment platform, and execution environment, replacing the usual "computer + operating system + development environment" platform. A good picture of this whole scenario can be found on *programmableweb.com* site, where more than 10,000 web APIs are catalogued and more than 7,000 mash-ups are being built upon those.

This multifaceted scenario brings up several different technologies, each with its own peculiarities, such as SOA [2], REST [3], Cloud Computing and Everything-as-a-Service (XaaS) [4]. Such diverse possibilities in web-based software development represent serious difficulties in understanding software basic elements and how they can be efficiently combined to develop real, practical systems in either personal, social or enterprise contexts. In order to overcome those difficulties, we try to understand and explain this emerging web in terms of a concept called *Social Machines* and envisage a language that can describe networks of such. The main goal of this work is to present an architecture description language that abstracts many complex details while developing real-world social machines.

2. SOCIAL MACHINES: UNDERSTANDING THE PROGRAMMABLE WEB

We firstly investigated the concept of a Social Machine in [5] and evolved it in a following publication, shown in [1]. Based on this last work, we believe a Social Machine (SM) can be defined as:

"*A connectable and programmable building block that wraps **(WI)** an information processing system **(IPS)** and defines a set of required **(RS)** and provided services **(PS)**, dynamically available under constraints **(C)** which are determined by, among other things, its relationships **(Rel)** with others*."

According to [1], an Information Processing System (**IPS**) abstracts any computational unit whose behavior is defined by the functional relationship between inputs and outputs. The Relationship (**Rel**) is the centerpiece of the SM model. A relationship "*defines the kinds of interactions between a computing process and its environment*". The Wrapper Interface (**WI**) abstracts any communication layer through which a SM externalizes its services to allow interactions with other SMs. The Provided Services (**PS**) represent the SM's business logic that is exposed as a dynamic set of services, which are directly affected by the type of relationship established with other SMs. The Required Services (**RS**) are an optional element defined by the model. They represent the set of services that a Social Machine needs to invoke in order to work properly. In addition, the Constraints (**C**) element specifies the rules or limitations that take place during the establishment of relationships between SMs.

In order to deal with every aspect of a SM, as formerly described, we created a high-level architecture description language. Other initiatives, such as service composition/orchestration, do not take into consideration the fundamentally simple SM concept.

3. SMADL: A NEW LANGUAGE FOR DESCRIBING SOCIAL MACHINES

We define **SMADL** – the **S**ocial **M**achines **A**rchitecture **D**escription **L**anguage – as an attempt to be a different way to program the Web, mixing concepts from Architecture Description Languages (ADLs) and Domain-Specific Languages (DSLs).

SMADL presents a simple textual syntax in favor of expressiveness. It was developed in Xtext language workbench, making it fully integrated to the Java Virtual Machine and Eclipse

IDE. The language can be used with or without low-level constructions (conditionals and loops). The concepts of the SM model are directly mapped into the language, facilitating new comers to use it. In SMADL, a relationship is represented by a single keyword, so composition possibilities for several SMs can be infinite, making it possible to create a network of SMs. Each SM establishes a relationship with others, just like an import mechanism in Java. To exemplify the language syntax, Figure 1 shows a code snippet of SMADL. The code sections are numerated to facilitate the following explanations.

```
/*
 * The 'relates to' section is optional. When present, this
 * section lists the other social machines used in
 * MyNewSocialMachine scope. If any relationships are listed
 * here, it is mandatory to setup each appropriate
 * interaction view according to its particular constraint
 */
SocialMachine MyNewSocialMachine relates to facebook, dpbox {
                                              2
  general constraint UNLIMITED

  Relationships {
 //SM 'dpbox' must be listed in the 'relates to' section
     dropBoxFiles with dpbox [
       uri = " https://www.dropbox.com/oauth2/"
       api-key = "745132132131"
       secret =  "ysdhgfs8485gas098hoashq98eyo3qwe"                  3
       user-token = "745132132131|HYlks234BeNj67V9kj323e4"
     ] type: FULL_ACCESS //every single operation of dpbox
                         5
 //SM 'facebook' must be listed in the 'relates to' section
     facebookPosts with facebook [
       uri = "https://graph.facebook.com/oauth/token"
       api-key = "543216431893328"
       secret =  "55dey851g0ff43b4df8e0n3dad1a32a0"                  4
       user-token = "5432164318933286BTeH0BSpUF6Cbj1EM3MI"
     ] type: LIST_OF_OPS (wallPost, listOfFriends)
  }                                                  6

  constructor(String baseUrl, Integer initialPort) {
    //Constructor body (dynamically typed expression)        1
    var destination = baseUrl + initialPort
  }

  op listFilesInDropboxFolder returns
    java.util.List<java.io.File> (String folder)
      constraint PRE_AUTH_SM                     7

  op createFacebookPost(String text)
}
```

Figure 1 – SMADL code snippet

A SM entity is defined using scopes between curly braces, following a Java-like syntax. In Figure 1, a SM entity is defined and called `MyNewSocialMachine`. Following, the SM model elements are mapped on SMADL structures: **(IPS)** ➔ **Part 1** allows the definition of an optional constructor, which may contain code in a JVM-based dynamically typed language called Xbase, provided as part of Xtext framework. **(Rel)** ➔ **Parts 2, 3 and 4** show how relationships are handled in SMADL. Notice that, in the piece of code, `facebook` and `dpbox` must have been previously defined as Social Machine entities just like `'MyNewSocialMachine'`. Parts 3 and 4 in code snippet show the actual configuration of each relationship in the '**`Relationships`**', hereby called interaction view. The current version of SMADL allows the creation of two types of relationships: OAuth-based and generic. The former abstracts all the details involved with authorization protocol OAuth, commonly used in nowadays web apps. The later allows the establishment of generic relationships, with any given number of parameters. **(WI)** ➔ **Part 7** illustrates the set of provided services that, in conjunction with their respective access constraints, composes the wrapper interface. **(PS)** ➔ **Part 7** shows an example of a provided service abstractly defined in terms of their signature. Notice that when defining PS, there is no need to establish relationships. The actual declaration of the relationship is under responsibility of the service requester. To define open common services in SMADL, the user only needs to write an operation without constraints as shown in the `createFacebookPost` operation, for instance. Relationship-driven services are supposed to be defined under certain constraints in the provider and declared in the **`LIST_OF_OPS`** section (part 6) for each interaction view in the requester code. **(RS)** ➔ **Parts 2 and 6**, these sections illustrate dependencies of the services provided by other SMs listed in the '**`relates to`**' section. Notice that the interaction view named '`facebookPosts`' can only access the following operations '`(wallPost, listOfFriends)`' from '`facebook`' social machine. This implicitly defines the set of required services in SMADL. And **(C)** ➔ **Parts 5, 6 and 7**, represented by the keyword '**`constraint`**' and '**`type`**'. In SMADL, constraints can be of three types: (1) general constraint, applicable to all provided services and written right after SM definition; (2) operation constraint, applicable to one operation (provided service) at once, it has a higher priority than a general constraint, that means, when an operation declares a constraint, the general constraint is not considered anymore; and (3) relationship constraint, which restricts which operations of the provider SM can be accessed in an interaction view.

SMADL was intentionally conceived in a level of abstraction to enable the generation of code for different contexts, augmenting its future use. Such contexts are covered by what we hereby name as a **generation profile**. Initially two profiles are defined: **1) Web Apps**, where the main goal is to generate code targeting well-known web architectures, such as Model-View-Controller (MVC) pattern using RESTful services and enabling OAuth 2.0 compliant authentication; and **2) Datacenter as a Service** (DaaS), which describes the internal elements of a data center and their relationships, including routers, data bases, load balancers, virtual machines, and so on, generating scripts for automatic instantiation of virtual machines according to the configuration the user gives. For this profile, a visual representation of the language was created using Eclipse Graphical Modeling Framework (GMF).

Creating two generation profiles helps on establishing SMADL as a practical solution and a relevant contribution out of this work. For future developments, we intend to provide more practical case studies of both visual and textual representations of SMADL, using it to specify web-enabled systems as crowd sourced platforms, combining already existent popular APIs, such as Facebook and Twitter, to acquire and process information, so creating practical social systems. This work was partially supported by the Brazilian National Institute of Science and Technology for Software Engineering (INES, www.ines.gov.br).

REFERENCES

[1] Buregio, V.A.A., Meira, S.L., Rosa, N.S. and Garcia, V.C. 2013. Moving Towards "Relationship-aware" Applications and Services: A Social Machine-oriented Approach. *17th IEEE International EDOC Conference (EDOCW 2013)* (Vancouver, Canada, 2013).

[2] Erl, T. 2005. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR.

[3] Fielding, R. and Taylor, R. 2000. Principled design of the modern Web architecture. *Proceedings of the 2000 International Conference on Software Engineering. ICSE 2000 the New Millennium* (2000), 407–416.

[4] Hazra, K. 2009. Cloud computing-the next chasm. *2009 International Conference on Computers and Devices for Communication* (2009).

[5] Meira, S.R.L., Buregio, V.A.A., Nascimento, L.M., Figueiredo, E., Neto, M., Encarnacao, B. and Garcia, V.C. 2011. The Emerging Web of Social Machines. *2011 IEEE 35th Annual Computer Software and Applications Conference* (Jul. 2011), 26–27.