Arquitetura de Software de Referência para Sistemas de Informação Governamentais

Alternative title: Reference Software Architecture for Government Information Systems

Maurício Serrano
Universidade de Brasília – FGA
Área Especial de Indústria Proj. A
72.444-240 Gama/DF – Leste
+55 (61) 3107-8901
serrano@unb.br

Milene Serrano
Universidade de Brasília – FGA
Área Especial de Indústria Proj. A
72.444-240 Gama/DF – Leste
+55 (61) 3107-8901
mileneserrano@unb.br

André Cruz Cavalcante
Universidade de Brasília – FGA
Área Especial de Indústria Proj. A
72.444-240 Gama/DF – Leste
+55 (61) 3107-8901
andrelink14@gmail.com

RESUMO

Nesse artigo, é descrita uma Arquitetura de Software de Referência para um Órgão Público Brasileiro. Essa arquitetura está baseada no Paradigma Orientado à Convenção sobre Configuração, o qual tenta diminuir a quantidade de decisões a serem tomadas pelos desenvolvedores, ganhando simplicidade, sem perder flexibilidade e produtividade. A arquitetura lida ainda com auditoria, rastreabilidade, autenticação, monitoramento e outros tópicos relevantes para sistemas de informação de Órgãos Públicos Brasileiros. Por fim, a arquitetura tem sido aplicada no desenvolvimento de dois sistemas de informação associados a um Órgão Público Brasileiro. Um desses é, atualmente, um dos mais importantes sistemas de informação do Órgão em questão.

Palavras-Chave

Arquitetura de Software de Referência, Reutilização, Governo, Padrão, Auditoria, Rastreabilidade, Autenticação, Monitoramento.

ABSTRACT

In this paper, we describe a Reference Software Architecture for a Brazilian Public Entity based on a software design paradigm, called Convention over Configuration, which tries to decrease the decisions made by developers, gaining simplicity, but not necessarily losing flexibility or productivity. Moreover, the proposed architecture deals with *auditing*, *tracing*, *logging*, *monitoring* and other relevant issues for information systems in the Brazilian government scope. Our Reference Software Architecture has been applying into two information systems' development at a Brazilian Public Entity. One of them is the most important information system at this Entity, nowadays.

Categories and Subject Descriptors

D.2 [**SOFTWARE ENGINEERING]:** D.2.1 [Requirements/Specifications] – *Elicitation methods*; *Methodologies*; *Tools*; D.2.11 [Software Architectures] – *Data abstraction*; *Domain-specific architectures*; *Information hiding*; *Patterns*; D.2.13 [Reusable Software] – *Reusable libraries*.

General Terms

Design, Security, Standardization, Languages.

Keywords

Reference Software Architecture, Reuse, Government, Pattern, Auditing, Tracing, Logging, Monitoring.

1. INTRODUÇÃO

As autarquias governamentais são organizações complexas, cujos impactos de suas atividades possuem alcance amplo. Nesse contexto, esses Órgãos registram, consultam e manipulam informações de alta relevância, tanto para conferir ao cidadão a prestação de serviço público quanto para apoiar o Estado na tomada de decisões estratégicas. Diante de problemas decorrentes da falta de disponibilidade, integridade, confidencialidade, autenticidade e manutenibilidade nos sistemas de informação associados aos Órgãos governamentais, ressalta-se a importância de padronização na Arquitetura de Software. No entanto, têm-se poucos estudos nesse contexto. Tornam-se necessários, ainda, esforços na área da Engenharia de Software e afins acordando soluções específicas para essas organizações. Por exemplo, a eficácia do serviço público demanda intensa cooperação entre organizações, visando, dentre outras necessidades, a troca de informações. Adicionalmente, normativas e padrões específicos, estabelecidos no setor público brasileiro (foco desse trabalho) devem ser rigorosamente respeitados. Apenas para ilustrar: Órgãos e entidades integrantes do Sistema de Administração dos Recursos de Tecnologia da Informação (SISP) devem observar o ePING [1] - *Padrões de Interoperabilidade de Governo Eletrônico* - no planejamento da contratação, aquisição e atualização de sistemas de informação. O ePING define um conjunto mínimo de premissas, políticas e especificações técnicas, que regulamentam a utilização das tecnologias de informação no âmbito do Governo Federal Brasileiro e estabelecem condições de interação entre –sistemas de informação – com as demais esferas governamentais.

Diante do exposto, o grupo dos autores tem desempenhado esforços na especificação de uma Arquitetura de Software de Referência (ASR) para novas contratações de desenvolvimento de Sistemas de Informação Governamentais por terceirizadas. Esse grupo atua como uma frente de Arquitetura de Software em um projeto de pesquisa e desenvolvimento, uma parceria entre a Universidade de Brasília – FGA e um Órgão Público Brasileiro. Dentre as contribuições relevantes do grupo, tem-se, principalmente: a análise de portfólio de sistemas de informação do Órgão; a investigação constante em busca de soluções de projeto, implementação, teste e manutenção específicas às necessidades desse tipo de Órgão governamental; a versão 1.0 da especificação da ASR; a capacitação de terceirizadas para o desenvolvimento de sistemas de informação segundo essa arquitetura; e o apoio no desenvolvimento, por parte da terceirizada, de *releases* de dois novos sistemas. Um desses sistemas é de uso interno do Ministério e o outro é de grande porte, âmbito nacional, e considerado um dos mais importantes sistemas de informação associados ao Ministério em questão.

Esse artigo, organizado em seções, procura apresentar em linhas gerais: o processo de análise institucional junto ao Órgão (Seção 2), no intuito de estudar o portfólio de sistemas de informações e as arquiteturas associadas a esses sistemas; a apresentação da ASR para Sistemas de Informação Governamentais (Seção 3); o desenvolvimento de sistemas de informação governamentais orientados pela arquitetura bem como outras contribuições associadas (Seção 4); e, finalmente, as considerações finais e possibilidades de trabalhos futuros (Seção 5).

2. ANÁLISE DE PORTFÓLIO

Ao longo do processo de análise do portfólio de sistemas de informação do Ministério, foram identificados mais de cinquenta sistemas em produção. Desses sistemas em produção, 58% foram desenvolvidos usando como base uma ASR em Java JSF e Struts [2]; 28% com base em uma ASR em PHP e 14% com base em uma ASR em ASP. Portanto, predominantemente, os sistemas de informação em produção nessa Autarquia são em Java.

Adicionalmente, ainda procurando acordar conformidades e não conformidades com base no portfólio de sistemas de informação, foram realizadas várias reuniões e entrevistas bem como aplicadas técnicas de introspecção e observação junto aos membros do Órgão Público Brasileiro. Dentre os membros participantes nesse processo, têm-se: membros da TI do Ministério; e membros das terceirizadas (i.e. desenvolvimento, qualidade e infraestrutura).

Tomando como base o rastro dos relatos acordados nas reuniões e entrevistas, foi possível a identificação de várias não conformidades, tipicamente encontradas em Órgãos governamentais, tais como: falta de qualidade dos sistemas de informação desenvolvidos; baixa produtividade da Fábrica de Software; insatisfação constante dos clientes finais dos sistemas de informação; falta de controle em termos de auditabilidade, *tracing* e *logging*; não possibilidade de monitoramento nos sistemas de informação em vigência; não possibilidade de extensibilidade dos sistemas de informação desenvolvidos, não sendo possível, por exemplo, acessá-los usando dispositivos móveis; ausência de flexibilidade desses sistemas visando à integração bem como a interconexão entre sistemas; excesso de documentação (maioria delas incompleta e não útil sob o ponto de vista do cliente e/ou de manutenção); não possibilidade de avaliação plena do sistema no ato da entrega do mesmo por parte da Fábrica de Software, e dificuldade de implantação de metodologias ágeis no desenvolvimento dos sistemas.

Apesar das não conformidades mencionadas, os membros da TI do Ministério solicitaram o não rompimento com as tecnologias já estabelecidas no Órgão. A preocupação sempre foi evitar mudanças bruscas em termos dos editais vigentes com as terceirizadas e/ou processos longos de capacitação da equipe de infraestrutura e da Fábrica de Software.

O Coordenador Geral da TI do Ministério (CGTI), também participante no processo de análise, enfatizou três critérios de qualidade em particular, os quais deveriam ser contemplados na especificação da nova ASR: produtividade, flexibilidade e extensibilidade. As preocupações dos demais membros da TI do Órgão concentravam-se nas questões de auditabilidade, *tracing*, *logging* e monitoramento. Os membros de infraestrutura do Órgão sensibilizaram-se com a manutenibilidade evolutiva dos sistemas de informação bem como com questões de integração entre sistemas. Essa integração necessariamente deveria se orientar pelo ePING - *Padrões de Interoperabilidade de Governo Eletrônico.*

Conforme detalhado na próxima seção, a frente de Arquitetura de Software do projeto dedicou-se à especificação de uma nova ASR capaz de atender às expectativas dos interessados, considerando os pontos de vista analisados e um suporte tecnológico emergente.

3. ARQUITETURA DE SOFTWARE

3.1 Convenção sobre Configuração

O Paradigma Orientado à Convenção sobre Configuração [3] é um modelo de desenvolvimento emergente, que procura minimizar o número de decisões tomadas pelo time de desenvolvimento, simplificando o processo. Essa simplificação é provida devido ao uso de padrões e configurações. Essencialmente, estar orientado à convenção sobre configuração significa que o time de desenvolvimento precisa definir apenas aspectos não convencionais. Por exemplo, quais convenções adotar visando padronizar o código em termos de anotações.

Dessa forma, o desenvolvimento orientado à convenção sobre configuração, quando apoiado por uma ferramenta e/ou plataforma que colabora na definição do comportamento desenhado pelo time de desenvolvimento, tende a diminuir o esforço na redação de arquivos de configuração; facilitar aspectos de padronização; melhorar a produtividade do time, e proporcionar uma manutenibilidade facilitada.

3.2 Suporte Tecnológico

Atualmente, existem várias propostas de plataformas e *frameworks* que se orientam pelo Paradigma Orientado à Convenção sobre Configuração, provendo suporte à redação de arquivos de configuração. Nesse cenário, o grupo dos autores pesquisou vários suportes tecnológicos, com destaque para: Grails [4], RubyOnRails [5] e Play [6]. Todas são plataformas que conferem ao time de desenvolvimento um ambiente completo, orientado à convenção sobre configuração, com *plugins*, arquivos de configuração, geração automática de código (em alguns casos) e outros facilitadores que permitem aumentar a produtividade do time de desenvolvimento, mantendo a qualidade do software.

Os estudos, conduzidos com experimentações, provas de conceito, pesquisa-ação, levantamento bibliográfico, e outros suportes à tomada de decisão, permitiram à equipe optar pela plataforma de desenvolvimento Grails. Dentre as razões que justificaram essa escolha, têm-se:

(i) o não rompimento com as tecnologias já utilizadas nas terceirizadas, prestadoras de serviços de desenvolvimento, qualidade e infraestrutura do Órgão Público em estudo. Esse tópico era uma grande preocupação dos membros da TI do Órgão, pois demandaria mudanças acentuadas na forma de trabalhar da Fábrica de Software; na geração dos relatórios de qualidade aferidos pela contratada de qualidade; nos ambientes de desenvolvimento, homologação, teste e produção mantidos pela equipe de infraestrutura associada ao Órgão, e nos editais de contratação vigentes na Autarquia. Portanto, optar por RubyOnRails seria adotar uma nova linguagem de programação, o Ruby. Essa linguagem não se assemelha à linguagem Java, predominantemente utilizada nos sistemas do Órgão. A adoção dessa plataforma demandaria maior esforço na capacitação das equipes, além de vários ajustes nos editais, infraestrutura e outros setores críticos da Autarquia. O Play faz uso da linguagem de programação Scala. Trata-se de uma linguagem objeto-funcional próxima à linguagem Java, inclusive compilada para Java bytecode e executável em uma

Máquina Virtual Java. O Grails usa a linguagem de programação Groovy, também compilada para Java bytecode e executável em uma Máquina Virtual Java. O Grails tem vantagens em relação ao Play por contar, principalmente, com toda a comunidade Java, a qual apoiou o desenvolvimento de *plugins* e outros suportes associados ao Grails. Adicionalmente, nossas pesquisas de tendência de mercado e na literatura [7] indicaram maior adesão da comunidade (academia e empresas) à plataforma Grails em detrimento do Play.

(ii) a necessidade de manter aspectos de auditabilidade, criação de perfis, *logging*, *tracing* e monitoramento para maior controle por parte dos clientes do sistema e pelos membros da TI do Órgão. Plataformas de desenvolvimento orientadas à convenção sobre configuração fazem uso de suportes reutilizáveis e de fácil configuração visando contemplar as aplicações de funcionalidades tipicamente requisitadas em sistemas de grande porte. Dentre essas funcionalidades existem várias associadas às questões de segurança, pois essas são vistas como necessidades de primeira ordem nos sistemas supracitados. Nesse sentido, o Grails é uma plataforma que provê vários *plugins* bem reconhecidos na comunidade Java, tais como: SpringSecurity [8], Log4J [9], AuditLogging [10] e JavaMelody [11]. O SpringSecurity provê suporte, por exemplo, para autorização no nível de método e de objetos usando ACLs (*Access Control List*); interfaces básicas para visualizar as configurações de segurança; controle de acesso de múltiplos sistemas independentes (*single sign-on*); autenticação OpenID e LDAP (*Lightweight Directory Access*).

(iii) extensibilidade para viabilizar o acesso dos sistemas via dispositivos móveis, tanto via *browser*, quanto via aplicação nativa. Ambos os casos são suportados pela plataforma Grails. Uma aplicação desenvolvida em Grails, por *default*, pode ser visualizada via *browser* de dispositivos móveis. Testes foram feitos nas arquiteturas móveis Android e FirefoxOS, acessando as aplicações em diferentes *browsers* de internet usando *tablets* e celulares das marcas Samsung® e LG®. O acesso via aplicação nativa é facilitado pelos serviços RESTFul [12].

(iv) integração entre sistemas com base no ePING e outros padrões de interoperabilidade do setor público brasileiro. Nesse caso, a ASR especifica o uso de serviços RESTFul para integração interna entre sistemas do Órgão, e *WebServices* [12] para integração com sistemas terceiros, de outros Órgãos públicos. Serviço RESTFul é um estilo arquitetural que faz uso de vários padrões: HTTP, URI, JSON, XML e outros. O Grails tem suporte que facilita o uso desse tipo de serviço. A abordagem usada na plataforma sugere a construção de serviços RESTFul com base em solicitações HTTP, que são gerenciadas pelas controladoras.

3.3 Visão Lógica

A ASR está estruturada de acordo com o Padrão MVC. A arquitetura base é a da plataforma Grails, adaptada de acordo com as necessidades de desenvolvimento de sistemas de informação governamentais brasileiros, conforme apresentado na Figura 01.

Na camada de visão, a ASR comporta os componentes visuais do sistema de informação, ou seja, os componentes de interação direta com o usuário. Nessa camada, a ASR está apoiada no Bootstrap [13]. Trata-se de uma estrutura de *front-end* intuitiva e emergente voltada para o desenvolvimento web padronizado, combinando elementos para alternar o conteúdo visível.

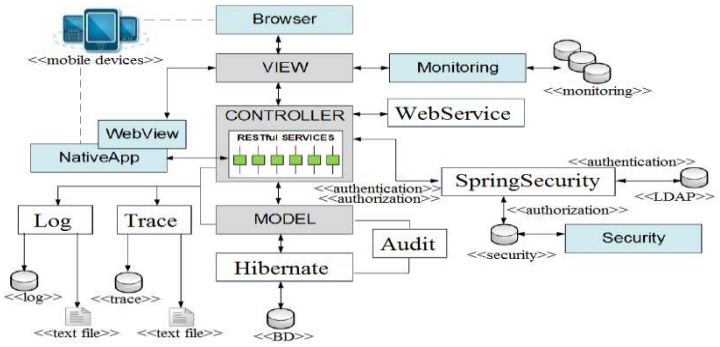

Figura 01: Arquitetura de Software de Referência

Na camada de visão, a ASR comporta os componentes visuais do sistema de informação, ou seja, os componentes de interação direta com o usuário. Nessa camada, a ASR está apoiada no Bootstrap [13]. Trata-se de uma estrutura de *front-end* intuitiva e emergente voltada para o desenvolvimento web padronizado, combinando elementos para alternar o conteúdo visível.

Na camada de controle, os componentes são capazes de receber requisições vindas da visão. Ao enviar uma requisição, a camada de controle trata essa solicitação conforme necessário. Essa camada contém ainda os serviços RESTFul. Nesses últimos, as aplicações nativas de dispositivos móveis podem interagir com a aplicação desenvolvida de acordo com as especificações da ASR. Por fim, deve-se considerar que a camada de controle é a responsável por colaborar com a camada de modelo.

Na camada de modelo, têm-se as classes de domínio do sistema de informação, as quais possuem os dados que deverão ser persistidos no banco de dados. Essa camada também é chamada de camada lógica, contendo as regras de negócio da aplicação.

Considerando módulos mais específicos da ASR, destacam-se os módulos responsáveis por: *logging*, *tracing*, monitoramento, auditabilidade, segurança e dispositivos móveis. No Módulo *Log*, têm-se as funcionalidades para controle do uso de um dado sistema de informação, permitindo o armazenamento de mensagens sobre as requisições que um dado usuário realizou. As mensagens podem ser armazenadas em arquivos de textos, ou mesmo em um banco de dados separado do banco do sistema de informação. Para padronização desse módulo, o mesmo faz uso de um *plugin* específico, chamado Log4J, reconhecido na comunidade de desenvolvedores Java e Grails.

No módulo *Trace*, esse também é responsável por controlar o uso do sistema de informação, entretanto, focando no controle de erros, falhas e/ou exceções que ocorrerão ao longo da execução desse sistema. Os erros, falhas e exceções podem ser armazenados em arquivos de texto ou outro banco de dados em separado. Esses eventos podem ser lidos normalmente em um arquivo de *logging*, sendo utilizados tipicamente para realização de *debugs*. A própria integração com Log4J já estabelece níveis de *tracing*.

O módulo *Monitoring* representa um componente separado dos componentes do sistema de informação. Sendo assim, o monitoramento é realizado em outro sistema. Nesse caso, basicamente, o sistema de monitoramento envia uma requisição HTTP ao sistema de informação, verificando o código de retorno dessa requisição. A partir desse código é possível saber se o sistema de informação está funcionando corretamente ou não. Caso não tenha um código de retorno, um erro é apresentado ao usuário. O sistema de monitoramento apresenta os códigos de

retorno em cores, permitindo ao usuário visualizar facilmente o status do sistema de informação monitorado. Os resultados quanto ao monitoramento podem ser salvos em bancos de dados - mais de um. Um *plugin* específico, chamado *JavaMelody*, permite monitorar os sistemas de informação através de gráficos.

No módulo *Audit*, inserções, alterações e remoções são gravadas (i.e. quem fez e quando foi realizado o evento). Tal prática torna o sistema de informação mais seguro e robusto, viabilizando identificar se alguém manipulou o sistema de forma indevida. O *plugin* do Audit Logging, incorporado à arquitetura, oferece suporte completo para auditoria de sistemas.

Existe um componente específico para tratar de segurança. Nesse componente, o *plugin* do Spring Security foi usado para fazer a autenticação e a autorização dos usuários nos sistemas. Trata-se de um *plugin* de código aberto, consolidado na comunidade Java. O Spring Security permite acessar uma base de dados com LDAP, um conjunto de protocolos multiplataforma desenhados para acessar informações distribuídas em uma rede. Adicionalmente, permite realizar a autenticação dos usuários no sistema. Nesse acesso, os dados e papéis de cada usuário são salvos em um banco de dados específico, dedicado às questões de segurança. Têm-se ainda os cadastros das URLS dos sistemas associados, possibilitando definir papéis de acesso para cada URL cadastrada.

A ASR provê também a extensibilidade dos sistemas de informação com dispositivos móveis. Essa extensibilidade pode ser de duas formas: via o próprio *browser* do dispositivo, acessando a URL padrão do sistema; ou por meio de aplicações nativas, instaladas nos próprios dispositivos. Nesse último caso, usam-se serviços RESTFul, presentes na *controller*. Ressalta-se que, adicionalmente ao estudo acordado nesse artigo, foram investigadas várias arquiteturas de referência (ex. [14] [15] [16]). Entretanto, tal detalhamento não cabe no presente artigo.

4. PRIMEIRAS IMPRESSÕES

Até o momento, além de várias provas de conceito, conduzidas usando pesquisa-ação e experimentações, a ASR foi testada no desenvolvimento: (i) de um sistema de informação de Gestão, para controle e gestão de projetos por parte da TI do Órgão investigado; (ii) de um sistema de informação para Ouvidoria desse Órgão. Esse encontra-se na primeira *release*; e (iii) de um dos maiores sistemas de informação associados ao Órgão. Esse encontra-se na primeira *release*. Para que o desenvolvimento fosse orientado pela ASR, as terceirizadas de desenvolvimento e de infraestrutura foram capacitadas. Ambientes de desenvolvimento, homologação, teste e produção foram configurados, em parceria com os membros de infraestrutura do Órgão. A ASR demonstrou robustez nos testes de carga e adequação nos testes de vulnerabilidade realizados nos ambientes de homologação do Órgão. Percebeu-se maior produtividade da equipe; satisfação dos clientes, e qualidade no código e nos entregáveis providos com base nas especificações arquiteturais.

5. CONSIDERAÇÕES FINAIS

Foram apresentados os esforços na especificação de uma ASR para novas contratações de desenvolvimento de sistemas de informação governamentais, especialmente desenhado para um Órgão Público Brasileiro. Em resumo, a arquitetura apoia-se: (i) no Paradigma Orientado à Convenção, usando a plataforma Grails, e aumentando a produtividade bem como a flexibilidade e a extensibilidade dos sistemas desenvolvidos; (ii) em serviços RESTFUL, visando facilitar a integração com sistemas terceiros, respeitando o ePING; (iii) em *plugins* específicos e reutilizáveis para definição de perfis, *logging* e outras necessidades tipicamente encontradas em sistemas de informação governamentais; (iv) em aplicações específicas para lidar com segurança e monitoramento, o que confere aos sistemas a possibilidade de serem auditados e controlados pelos interessados; (v) em padrões estabelecidos na Engenharia de Software como boas práticas de *design*, implementação, teste e manutenção, no caso: práticas ágeis; código anotado; testes unitários/funcionais/integração, cobertura e outros. Como trabalho em andamento salientam-se esforços na aplicação da ASR em outros sistemas de interesse do Ministério; refinamentos na especificação arquitetural, os quais são pontuais, mas colaboram no ajuste fino de alguns aspectos; e desenvolvimento de *plugins* específicos, visando melhorar ainda mais a reutilização de boas práticas. Como trabalho futuro, pretende-se refinar os módulos da ASR que lidam diretamente com questões de *Web Design*. Nesse sentido, estudos sobre usabilidade, tecnologias e tópicos associados corresponderão aos próximos esforços do grupo.

6. AGRADECIMENTOS

Especiais agradecimentos ao Órgão Público, terceirizadas associadas e demais frentes de pesquisa do projeto, os quais colaboraram na condução da pesquisa apresentada nesse artigo.

7. REFERENCES

[1] ePING. Acesso: http://governoeletronico.gov.br/acoes-e-projetos/e-ping-padroes-de-interoperabilidade (Abril 2015)

[2] Struts. Acesso: https://struts.apache.org/ (Abril 2015)

[3] Miller, J. (2009). *Design For Convention Over Configuration*. Acesso: http://msdn.microsoft.com/en-us/magazine/dd419655.aspx (Abril 2015)

[4] Grails. Acesso: https://grails.org/ (Abril 2015)

[5] RubyOnRails. Acesso: http://rubyonrails.org/ (Abril 2015)

[6] Play. Acesso: https://www.playframework.com/ (Abril 2015)

[7] RebelLabs. (2013) Curious Coder's Java Web Frameworks.

[8] SpringSecurity, Grails Plugin. Acesso: http://grails.org/plugin/spring-security-core (Abril 2015)

[9] Log4J. Grails Plugin. Acesso: http://logging.apache.org/log4j/1.2/index.html (Abril 2015)

[10] AuditLogging, Grails Plugin. Acesso: http://grails.org/plugin/audit-logging (Abril 2015)

[11] JavaMelody, Grails Plugin. Acesso: http://grails.org/plugin/grails-melody (Abril 2015)

[12] Pautasso et al. (2008). Restful Web Services vs. Big'Web Services. 17th Int. Conf. on World Wide Web (pp. 805-814).

[13] BootStrap. Acesso: http://getbootstrap.com/ (Abril 2015)

[14] German Federal Government for IT. (2003) Standards and Architectures for e-Government Applications.

[15] Al-Khanjari et al. (2014) Developing a Service Oriented E-Government Architecture. Journal of Software Engineering and Its Applications. Vol.8(5), pp. 29-42.

[16] Nakagawa, E. Y.; Oquendo, F.; Becker, M. (2012) RAModel: A Reference Model for Reference Architectures. 6th European Conference on Software Architecture.