8
Referências Bibliográficas

[1] GRONE, B.; KNOPFEL, A.; TABELING, P. Component vs. component: Why we need more than one definition. In: **Proceedings of ECBS'05: 12th IEEE International Conference and Workshops on Engineering of Computer-Based Systems**. Washington, DC, USA: IEEE Computer Society, 2005. p. 550–552. ISBN 0-7695-2308-0. 1

[2] SZYPERSKI, C. **Component Software: Beyond Object-Oriented Programming**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002. ISBN 0201745720. 1

[3] SZYPERSKI, C. Component technology: what, where, and how? In: **Proceedings of the 25th International Conference on Software Engineering (ICSE'03)**. Washington, DC, USA: IEEE Computer Society, 2003. p. 684–693. ISBN 0-7695-1877-X. 1, 2.2.5, 4.1, 7

[4] EMMERICH, W. Distributed component technologies and their software engineering implications. In: **Proceedings of the 24th International Conference on Software Engineering (ICSE'02)**. New York, NY, USA: ACM, 2002. p. 537–546. ISBN 1-58113-472-X. 1, 2.1, 7

[5] MANCINELLI, F. et al. Managing the complexity of large free and open source package-based software distributions. In: **Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering (ASE'06)**. Washington, DC, USA: IEEE Computer Society, 2006. p. 199–208. ISBN 0-7695-2579-2. 1, 4.1

[6] FLISSI, A. et al. Deploying on the grid with DeployWare. In: **Proceedings of the 8th International Symposium on Cluster Computing and the Grid (CCGRID'08)**. IEEE Computer Society, 2008. p. 177–184. Disponível em: <`http://dblp.uni-trier.de/db/conf/ccgrid/ccgrid2008.html#FlissiDDM08`>. (document), 1, 1, 2.3.7, 2.13, 5, 5.5

[7] LACOUR, S.; PéREZ, C.; PRIOL, T. Deploying CORBA components on a computational grid: General principles and early experiments using

the Globus Toolkit. In: **Proceedings of the 2nd International Working Conference on Component Deployment (CD'04)**. Springer-Verlag, 2004. (Lecture Notes in Computer Science, 3083), p. 35–49. Disponível em: <http://www.irisa.fr/paris/Biblio/Papers/Lacour/LacPerPri04CD.pdf>. 1

[8] FOSTER, I. T. The anatomy of the grid: Enabling scalable virtual organizations. In: **Euro-Par '01: Proceedings of the 7th International Euro-Par Conference on Parallel Processing**. London, UK: Springer-Verlag, 2001. p. 1–4. ISBN 3-540-42495-4. 1

[9] SUN MICROSYSTEMS. **Enterprise JavaBeans v3.0**. Santa Clara, USA, maio 2006. Document: JSR/220. 1, 2.3.1, 2.3.1

[10] OBJECT MANAGEMENT GROUP. **CORBA Components - Version 3.0**. Needham, USA, jun. 2002. Document: formal/2002-06-65. 1, 2.3.3, 2.3.3, 2.3.3, 2.3.4, 2.4, 3.1

[11] COULSON, G. et al. A generic component model for building systems software. **ACM Transactions on Computer Systems**, ACM, New York, NY, USA, v. 26, n. 1, p. 1–42, 2008. ISSN 0734-2071. (document), 1, 2.3.2, 2.3, 2.3.2

[12] BRUNETON, E. et al. The FRACTAL component model and its support in java: Experiences with auto-adaptive and reconfigurable systems. **Software: Practice and Experience**, John Wiley & Sons, Inc., New York, NY, USA, v. 36, n. 11-12, p. 1257–1284, 2006. ISSN 0038-0644. 1, 2.3.7, 2.3.7

[13] DENG, G. et al. DAnCE: A QoS-enabled component deployment and configuration engine. In: **Proceedings of the 3rd Working Conference on Component Deployment (CD'05)**. [s.n.], 2005. p. 67–82. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.60.36>. (document), 1, 2.3.4, 2.8, 2.3.4

[14] OBJECT MANAGEMENT GROUP. **Deployment and Configuration of Component-based Distributed Applications - Version 4.0**. Needham, USA, abr. 2006. Document: formal/2006-04-02. 1, 2.3.4, 2.4

[15] FLISSI, A.; MERLE, P. A generic deployment framework for grid computing and distributed applications. In: **Proceedings of the 2nd**

International OTM Symposium on Grid computing, high-performAnce and Distributed Applications (GADA'06). [S.l.]: Springer-Verlag, 2006. (Lecture Notes in Computer Science, v. 4279), p. 1402–1411. 1, 2.3.7

[16] MANOLIOS, P.; VROON, D.; SUBRAMANIAN, G. Automating component-based system assembly. In: **Proceedings of ISSTA'07: International Symposium on Software Testing and Analysis**. New York, NY, USA: ACM, 2007. p. 61–72. ISBN 978-1-59593-734-6. 1

[17] LACOUR, S.; PEREZ, C.; PRIOL, T. Generic application description model: toward automatic deployment of applications on computational grids. **The 6th IEEE/ACM International Workshop on Grid Computing (CCGRID'05)**, nov. 2005. (document), 1, 2.3.5, 2.9, 2.3.6, 2.4

[18] CUDENNEC, L. **CoRDAGe : Un service générique de co-déploiement et redéploiement d'applications sur grilles**. Tese (Thèse de doctorat) — Université de Rennes 1, IRISA, Rennes, France, jan. 2009. 1, 7

[19] MATEVSKA-MEYER, J.; HASSELBRING, W.; REUSSNER, R. H. Software architecture description supporting component deployment and system runtime reconfiguration. In: **Proceedings of the 9th International Workshop on Component-oriented Programming (WCP'04)**. [S.l.: s.n.], 2004. 1.1, 7

[20] BATISTA, T. V.; JOOLIA, A.; COULSON, G. Managing dynamic reconfiguration in component-based systems. In: **Proceedings of the 2nd European Workshop Software Architecture (EWSA'05)**. [S.l.]: Springer, 2005. (Lecture Notes in Computer Science, v. 3527), p. 1–17. ISBN 3-540-26275-X. 1.1, 2.3.2, 7

[21] HALL, R. S. et al. An architecture for post-development configuration management in a wide-area network. In: **Proceedings of the 17th International Conference on Distributed Computing Systems (ICDCS'97)**. Washington, DC, USA: IEEE Computer Society, 1997. p. 269. ISBN 0-8186-7813-5. 2.1, 4.1

[22] ARSHAD, N.; HEIMBIGNER, D.; WOLF, A. L. Deployment and dynamic reconfiguration planning for distributed software systems. **Software Quality Control**, Kluwer Academic Publishers, Hingham, MA, USA, v. 15, n. 3, p. 265–281, 2007. ISSN 0963-9314. 2.1

[23] COUPAYE, T.; ESTUBLIER, J. Foundations of enterprise software deployment. In: **Proceegins of CSMR'00: Conference on Software Maintenance and Reengineering**. Washington, DC, USA: IEEE Computer Society, 2000. p. 65–74. ISBN 0-7695-0546-5. 2.1

[24] LAU, K.-K.; WANG, Z. Software component models. **IEEE Transactions on Software Engineering**, IEEE Computer Society, Los Alamitos, CA, USA, v. 33, n. 10, p. 709–724, 2007. ISSN 0098-5589. 2.2.2, 2.2.3, 2, 2.4, 3.3

[25] SUN MICROSYSTEMS. **JavaBeans 1.01 Specification**. Santa Clara, USA, ago. 1997. `http://java.sun.com/javase/technologies/desktop/javabeans/docs/spec.html`. 2.3.1

[26] SUN MICROSYSTEMS. **JMX: Java Management Extensions**. Santa Clara, USA, out. 2006. `http://java.sun.com/javase/6/docs/technotes/guides/jmx/`. 2.3.1

[27] OBJECT MANAGEMENT GROUP. **Common Object Request Broker Architecture: Core Specification - Version 2.6**. Needham, USA, dez. 2001. Document: formal/2001-12-01. 2.3.2, 2.3.3, 3

[28] GARLAN, D.; MONROE, R. T.; WILE, D. Acme: Architectural description of component-based systems. In: LEAVENS, G. T.; SITARAMAN, M. (Ed.). **Foundations of Component-Based Systems**. [S.l.]: Cambridge University Press, 2000. p. 47. 2.3.2

[29] JOOLIA, A. et al. Mapping adl specifications to an efficient and reconfigurable runtime component platform. In: **Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05)**. Washington, DC, USA: IEEE Computer Society, 2005. p. 131–140. ISBN 0-7695-2548-2. 2.3.2

[30] OBJECT MANAGEMENT GROUP. **Common Object Request Broker Architecture: Core Specification - Version 3.0**. Needham, USA, dez. 2002. Document: formal/2002-12-06. 2.3.3

[31] WANG, N.; SCHMIDT, D. C.; O'RYAN, C. Overview of the CORBA Component Model. In: _____. **Component-based Software Engineering: Putting the Pieces Together**. Boston, USA: Addison-Wesley Longman Publishing Co., Inc., 2001. p. 557–571. ISBN 0-201-70485-4. 2.3.3

[32] NARDI, A. R. **Componentes CORBA**. Dissertação (Mestrado) — Universidade de São Paulo, set. 2003. (document), 2.5, 2.6, 2.3.3

[33] OBJECTWEB. **The Open CORBA Component Model Platform**. jul. 2005. Disponível em: <`http://openccm.objectweb.org`>. 2.3.3

[34] BRICLET, F.; CONTRERAS, C.; MERLE, P. OpenCCM : une infrastructure à composants pour le déploiement d'applications à base de composants CORBA. In: IMAG/LSR (Ed.). **Conf'erence Francophone sur le D'eploiement et la (Re)Configuration de Logiciels (DECOR'04)**. [s.n.], 2004. (ISBN : 2-7261-1276-5), p. 101–112. Disponível em: <`http://hal.archives-ouvertes.fr/hal-00003287/en`>. 2.3.3, 2.3.3

[35] SCHMIDT, D. C. **CIAO: Component-Integrated ACE ORB**. out. 2006. Disponível em: <`http://www.dre.vanderbilt.edu/CIAO`>. 2.3.3, 2.3.4

[36] KEBBAL, D.; BERNARD, G. Component search service and deployment of distributed applications. **Distributed Objects and Applications**, IEEE Computer Society, Los Alamitos, USA, v. 00, p. 0125, 2001. 2.3.3

[37] OBJECT MANAGEMENT GROUP. **CORBA Component Model Specification - Version 4.0**. Needham, EUA, abr. 2006. Document: formal/2006-04-01. 2.3.4

[38] SCHMIDT, D.; KUHNS, F. An overview of the real-time CORBA specification. **IEEE Computer**, v. 33, n. 6, p. 56–63, jun. 2000. ISSN 0018-9162. 2.3.4

[39] SCHMIDT, D. C. **TAO: The ACE ORB**. jun. 2007. Disponível em: <`http://www.dre.vanderbilt.edu/TAO`>. 4

[40] PÉREZ, C. **Contribution à la définition et la mise en oeuvre dún modèle de programmation à base de composants logiciels pour la programmation des grilles informatiques**. France: [s.n.], 2006. Habilitation à diriger des recherches - Université de Rennes I. 2.3.5

[41] CUDENNEC, L.; ANTONIU, G.; BOUGé, L. CoRDAGe: towards transparent management of interactions between applications and resources. In: MICHAEL GERNDT AND JESUS LABARTA AND BARTON MILLER. **Proceedings of the International Workshop on Scalable**

Tools for High-End Computing (STHEC'08). Kos Grèce, 2008. p. 13–24. Disponível em: <`http://hal.inria.fr/inria-00288339/en/`>. (document), 2.3.6, 2.10, 2.4

[42] AUGUSTO, C. E. L. **Uma Infra-Estrutura para a Execução Distribuída de Componentes de Software**. Dissertação (Mestrado) — PUC-Rio, Brasil, 2008. 3, 3.1, 3.2, 3.3, 4.4, 4.4, 5.7, 6, 6.1, 7

[43] TECGRAF. **SCS: Software Component System**. nov. 2008. Disponível em: <`http://www.tecgraf.puc-rio.br/~scorrea/scs`>. 3

[44] BOX, D. **Essential COM**. Boston, USA: Addison-Wesley Longman Publishing Co., Inc., 1997. Foreword By-Grady Booch and Foreword By-Charlie Kindel. ISBN 0201634465. 3.1

[45] IERUSALIMSCHY, R. **Programming in Lua**. Second edition. [S.l.]: Lua.org, 2006. ISBN 8590379825. 3.1, 4.2

[46] MAIA, R.; CERQUEIRA, R.; CALHEIROS, R. OiL: An object request broker in the Lua language. In: **Proceedings of SBRC'07 - Salão de Ferramentas**. Porto Alegre, Brazil: SBC, 2006. p. 1439–1446. 3.1, 5.1.2

[47] HOEK, A. van der; WOLF, A. L. Software release management for component-based software. **Software – Practice & Experience**, John Wiley & Sons, Inc., New York, NY, USA, v. 33, n. 1, p. 77–98, 2003. ISSN 0038-0644. 4.1, 4.1, 4.5

[48] HOEK, A. van der et al. Software release management. **SIGSOFT Software Engineering Notes**, ACM, New York, NY, USA, v. 22, n. 6, p. 159–175, 1997. ISSN 0163-5948. 4.1

[49] DEBIAN. **Debian policy manual**. 2009. `http://www.debian.org/doc/debian-policy/index.html`. Último acesso em 06/2009. 4.1, 4.2

[50] BAILEY, E. **Maximum RPM**. [S.l.]: Red Hat Inc, 2000. 442 p. `http://www.rpm.org/max-rpm`. ISBN 1-888172-78-9. 4.1, 4.2

[51] LEONARD, T. **Zero Install Project**. jun. 2009. Disponível em: <`http://0install.net`>. 4.2

[52] DOLSTRA, E. **Nix Project**. set. 2008. Disponível em: <`http://www.nix.org`>. 4.2

[53] STANDARD for information technology - portable operating system interface (POSIX). Shell and utilities. **IEEE Std 1003.1, 2004 Edition.**

The Open Group Technical Standard. Base Specifications, Issue 6. Includes IEEE Std 1003.1-2001, IEEE Std 1003.1-2001/Cor 1-2002 and IEEE Std 1003.1-2001/Cor 2-2004. Shell and Utilities, 2004. 4.2

[54] RED HAT. **Cygwin Project**. jan. 2009. Disponível em: <`http://www.redhat.com/services/custom/cygwin`>. 4.2

[55] HIETANIEMI, J. **CPAN Project**. jul. 2009. Disponível em: <`http://www.cpan.org`>. 4.2

[56] RUBYGEMS Manuals. 2009. Disponível em: <`http://www.rubygems.org`>. 4.2

[57] LUAROCKS Project. abr. 2009. Disponível em: <`http://www.luarocks.org`>. 4.2

[58] LUAROCKS. **Rockspec Format**. abr. 2009. Disponível em: <`http://www.luarocks.org/en/Rockspec_format`>. 4.2.1

[59] APACHE. **The Apache ANT Project**. jan. 2010. Disponível em: <`http://ant.apache.org`>. 4.3

[60] IONA TECHNOLOGIES. **The corbaloc and corbaname URLs**. [S.l.], ago. 2003. `http://www.iona.com/devcenter/corba/corbaloc.pdf`. 5.1

[61] FONSECA, E. **Monitorando o Ambiente de Execução de Componentes de Software Distribuídos**. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil, 2009. 6, 6.2

[62] DEAN, J.; GHEMAWAT, S. Mapreduce: Simplified data processing on large clusters. In: **Proceedings of OSDI 2004**. Berkeley, USA: USENIX Association, 2004. p. 137–150. (document), 6, 6.2

[63] FONSECA, E.; CORREA, S.; CERQUEIRA, R. Experimenting middleware-level monitoring facilities to observe component-based applications. In: **Proceedings of SBCARS 2008**. Porto Algre, Brasil: EDIPUCRS, 2008. p. 96–106. ISBN 978-85-7430-786-2. 7

[64] TSENG, T.-L. B. et al. Applying genetic algorithm for the development of the components-based embedded system. **Computer Standards & Interfaces**, v. 27, n. 6, p. 621 – 635, 2005. ISSN 0920-5489.

Disponível em: <http://www.sciencedirect.com/science/article/B6TYV-4F5S6C2-1/2/47d01c7228b5c5fa4f02db5227be3171>. 7

A
Apêndice

A.1
IDL do SCS integrado ao LuaRocks

O Código A.1 apresenta os tipos de dados usados pelo *framework* do SCS e retornado pelos métodos das interfaces básicas do seu modelo de componentes. O modelo de componentes, por sua vez, possui suas interfaces ilustradas no Código A.2.

Dentre os componentes da infraestrutura de execução do SCS apenas o *ExecutionNode* e *Repository* sofreram mudanças em suas IDL, que são apresentada no Código A.4 e A.5, respectivamente. As interfaces usadas pelo componente do contêiner é apresentada no Código A.3.

A.2
IDL dos Descritores Propostos

A integração com o LuaRocks e o desenvolvimento dos serviços de implantação motivaram a criação de dois descritores: um para indicar a lógica estrutural do componente e outro para indicar os artefatos que o implementam. O Código A.6 apresenta a definição integral desses descritores.

A.3
IDL dos Serviços para Implantação

Por fim, o Código A.7 apresenta as entidades virtuais que são manipuladas em tempo de planejamento e que existem através do uso do serviço *DeployManager*. Por outro lado, as interfaces dos serviços *DeployManager* e *Packager* são apresentadas integralmente no Código A.8.

A.4
Algoritmos do Mecanismo de Mapeamento Automático

O mecanismo que permite o mapeamento em mais alto nível considerando níveis graduais de detalhamento completa o mapeamento dos componentes na infraestrutura de execução do SCS e nas máquinas que serão usadas. O Código A.9 apresenta a implementação do método *autofit_container* que

associa um componente a um contêiner caso o usuário não tenha especificado qual contêiner usar no plano de implantação. O Código A.10 apresenta a implementação do método *autofit_exnode* que associa um contêiner a um nó de execução caso o usuário não tenha especificado qual nó de execução utilizar no plano de implantação. E, por sua vez, o Código A.11 apresenta a implementação do método *autofit_host* que associa um nó de execução a uma máquina caso o usuário não tenha especificado qual máquina utilizar no plano de implantação. Esses métodos são parametrizados pela lista de implementações disponíveis para um dado componente e por restrições de linguagem e plataforma. Portanto, caso deseje-se estender, ou mesmo alterar, a estratégia de mapeamento automático basta reimplementar esses métodos que fazem parte da implementação do serviço *DeployManager*.

Código A.1: Definição dos tipos usados pela API básica do Modelo de Componentes do SCS

```
#ifndef SCSDESCS_IDL
#define SCSDESCS_IDL

module scs {
  module core {
    exception StartupFailed {};
    exception ShutdownFailed {};
    exception InvalidName {
      string name;
    };
    exception InvalidConnection {};
    exception AlreadyConnected {};
    exception ExceededConnectionLimit {};
    exception NoConnection {};

    typedef unsigned long ConnectionId;
    typedef sequence<string> NameList;

    struct FacetDescription {
      string name;
      string interface_name;
      Object facet_ref;
    };
    typedef sequence<FacetDescription> FacetDescriptions;

    struct ConnectionDescription {
      ConnectionId id;
      Object objref;
    };
    typedef sequence<ConnectionDescription> ConnectionDescriptions;

    struct ReceptacleDescription {
      string name;
      string interface_name;
      boolean is_multiplex;
      ConnectionDescriptions connections;
    };
    typedef sequence<ReceptacleDescription> ReceptacleDescriptions;

    struct ComponentId {
      string name;
      octet major_version;
      octet minor_version;
      octet patch_version;
    };
    typedef sequence<ComponentId> ComponentIdSeq;
  };
};
#endif
```

Código A.2: Interfaces do Modelo de Componentes do SCS

```
#include "scsdescs.idl"

module scs {  module core {
  interface IComponent {
    void startup() raises (StartupFailed);
    void shutdown() raises (ShutdownFailed);

    Object getFacet (in string facet_interface);
    Object getFacetByName (in string facet);
    ComponentId getComponentId ();
  };

  interface IReceptacles {
    ConnectionId connect (in string receptacle, in Object obj)
       raises (InvalidName, InvalidConnection, AlreadyConnected,
               ExceededConnectionLimit);
    void disconnect (in ConnectionId id)
       raises (InvalidConnection, NoConnection);
    ConnectionDescriptions getConnections (in string receptacle)
       raises (InvalidName);
  };

  interface IMetaInterface {
    FacetDescriptions getFacets();
    FacetDescriptions getFacetsByName(in NameList names)
       raises (InvalidName);
    ReceptacleDescriptions getReceptacles();
    ReceptacleDescriptions getReceptaclesByName(in NameList names)
       raises (InvalidName);
  };
};};
```

Código A.3: Interfaces do contêiner da Infraestrutura de Execução do SCS

```
#include "scs.idl"
#include "deployment_descriptions.idl"

module scs {    module container {
  typedef sequence<unsigned long> InterceptorIds;

  struct ComponentHandle {
    core::IComponent cmp;
    core::ComponentId id;
    unsigned long instance_id;
  };
  typedef sequence<ComponentHandle> ComponentHandleSeq;

  exception ComponentNotFound{};
  exception ComponentAlreadyLoaded{};
  exception LoadFailure{};
  exception InterceptorNotInstalled{};
  exception ListLockFail{};
  exception ContainerHalted{};

  interface ComponentLoader {
    ComponentHandle load (in core::ComponentId id, in StringSeq args)
       raises (ComponentNotFound, ComponentAlreadyLoaded, LoadFailure);
    void unload (in ComponentHandle handle)
       raises (ComponentNotFound);
  };

  interface ComponentCollection {
    ComponentHandleSeq getComponent (in core::ComponentId id);
    ComponentHandleSeq getComponents ();
  };

  interface ComponentInterception {
    // 0 in position counts as end of the list.
    // The same goes for positions above the list's current size.
    ComponentHandle loadInterceptor (in core::ComponentId id,
                    in StringSeq args, in unsigned long position,
                    in string type)
       raises (ListLockFail, ComponentNotFound,
             ComponentAlreadyLoaded, LoadFailure);
    void unloadInterceptor (in ComponentHandle handle)
       raises (InterceptorNotInstalled, ListLockFail, ComponentNotFound);
    void changePosition (in unsigned long instance_id,
                         in unsigned long position)
       raises (InterceptorNotInstalled, ListLockFail, ComponentNotFound);
    unsigned long getInterceptorPosition (in unsigned long instance_id)
       raises (InterceptorNotInstalled, ListLockFail, ComponentNotFound);
    InterceptorIds getClientInterceptorsOrder ();
    InterceptorIds getServerInterceptorsOrder ();
  };

  interface ComponentSuspension {
    void suspend ();
    void halt ();
    void resume ();
    long getStatus ();
    // status: 0 means no suspension
    //         positive number means a halt (launches exception)
    //         negative number means a suspension (yields coroutines)
  };
};};
```

Código A.4: Interfaces do nó de execução da Infraestrutura de Execução do SCS

```
#include "scs.idl"
#include "container.idl"
#include "deployment_descriptions.idl"

module scs {    module execution_node {
  typedef sequence<StringSeq> ArgsSeq;

  exception ContainerAlreadyExists{};
  exception InvalidContainer{};
  exception RequirementNotMet{ string reason; };
  exception InvalidProperty{ string reason; };
  exception ComponentNotInstalled{ string reason; };

  struct Property{
    string name;
    string value;
    boolean read_only;
  };
  typedef sequence<Property> PropertySeq;

  struct ContainerDescription {
    core::IComponent container;
    string container_name;
    core::IComponent execution_node;
  };
  typedef sequence<ContainerDescription> ContainerDescriptionSeq;

  interface ExecutionNode {
    core::IComponent startContainer (in string container_name,
                                     in PropertySeq props)
          raises (ContainerAlreadyExists, InvalidProperty);
    void killContainer(in string container_name)
          raises (core::InvalidName);
    core::IComponent getContainer (in string container_name);
    ContainerDescriptionSeq getContainers ();
    string getName();
    string getPlatform();
  };

  interface ContainerManager {
    void registerContainer(in string name, in core::IComponent ctr,
                           in string pid)
          raises (ContainerAlreadyExists, InvalidContainer);
    void unregisterContainer(in string name) raises (core::InvalidName);
  };

  interface Installer {
    void install(in core::ComponentId id, in string lang)
          raises (ComponentNotInstalled);
    void uninstall(in core::ComponentId id, in string lang);
    StringSeq getEnvVars(in string lang);
    string getRootPath(in string lang);
    StringSeq getDependenciesPath(in core::ComponentId id,
                                  in string lang);
    string getInstallPath(in core::ComponentId id, in string lang);
    core::ComponentIdSeq getInstalledComponents(in string lang);
    deployer::ComponentDescription getDescription(in core::ComponentId id,
                                                  in string lang)
          raises (ComponentNotInstalled);
  };
};};
```

Código A.5: Interfaces do repositório da Infraestrutura de Execução do SCS

```
#include "deployment_descriptions.idl"

module scs {    module repository {
  exception ComponentAlreadyUploaded{};
  exception ComponentNotUploaded{};

  struct Constraint {
   string key;
   string value;
  };
  typedef sequence<Constraint> ConstraintSeq;

  interface ComponentRepository {
    void upload (in deployer::Package pkg, in string help_info)
            raises (ComponentAlreadyUploaded, ComponentNotUploaded);

    void delete (in core::ComponentId id, in string lang, in string plat)
            raises (ComponentNotUploaded);

    void copy (in deployer::ComponentDescription desc,
             in ComponentRepository repo)
            raises (ComponentAlreadyUploaded, ComponentNotUploaded);

    deployer::Package download (in core::ComponentId id,
                            in string lang, in string plat)
            raises (ComponentNotUploaded);

    deployer::ComponentDescriptionSeq search (
                                    in ConstraintSeq constraints);
    deployer::ComponentDescriptionSeq searchById (
                                    in core::ComponentId id);

    deployer::ComponentDescription getDescription (
                                  in core::ComponentId id,
                                  in string lang, in string plat)
            raises (ComponentNotUploaded);

    deployer::ComponentDescriptionSeq getAllDescriptions();
  };
};};
```

Código A.6: Definição dos descritores do Modelo e Implementação

```
#include "scs.idl"

module scs {
  typedef sequence<string> StringSeq;
  typedef sequence<octet> OctetSeq;

  module deployer {
    struct PhysicalResources {
      unsigned long cpu;
      unsigned long mem;
      string os;
      string arch;
      unsigned long bandwidth;
    };

    struct HostDescription {
      string name;
      string ip;
      unsigned long port;
      PhysicalResources resources;
      StringSeq languages;
    };
    typedef sequence<HostDescription> HostSeq;

    struct Facet {
      string name;
      string interface_name;
    };
    typedef sequence<Facet> FacetSeq;

    struct Receptacle {
      string name;
      string interface_name;
      boolean is_multiplex;
    };
    typedef sequence<Receptacle> ReceptacleSeq;

    struct ComponentTemplate {
      core::ComponentId id;
      FacetSeq facets;
      ReceptacleSeq receptacles;
    };
    typedef sequence<ComponentTemplate> ComponentTemplateSeq;

    struct Artefact {
      string name;
      string class_name;
    };
    typedef sequence<Artefact> ArtefactSeq;

    struct ComponentDescription {
      core::ComponentId template_id;
      string lang;
      string plat;
      string entry_point;
      ArtefactSeq facets;
      ArtefactSeq receptacles;
    };
    typedef sequence<ComponentDescription> ComponentDescriptionSeq;

    struct Package {
      ComponentDescription info;
      OctetSeq file;
    };
    typedef sequence<Package> PackageSeq;
  };
};
```

Código A.7: Definições das entidades virtuais do planejamento

```
#include "scs.idl"
#include "deployment.idl"
module scs {    module deployer {
  exception AlreadyDeployed{}; exception NotDeployed{};
  typedef sequence<ComponentEntity>     ComponentEntitySeq;
  typedef sequence<ContainerEntity>     ContainerEntitySeq;
  typedef sequence<ExecutionNodeEntity> ExecutionNodeEntitySeq;
  typedef sequence<RepositoryEntity>    RepositoryEntitySeq;
  interface Deployment {  //processo de implantação
    boolean deploy()              raises (AlreadyDeployed);
    boolean deploy_connections()  raises (AlreadyDeployed);
    boolean activate()            raises (NotDeployed);
    boolean deactivate()          raises (NotDeployed);
    boolean undeploy();
  };
  // tipos de dados da conexão planejada
  struct FacetEndpoint { BasicComponentEntity entity; string facetname; };
  struct ReceptacleConnections {
    string receptname; FacetEndpointSeq endpoints;
  };
  typedef sequence<FacetEndpoint> FacetEndpointSeq;
  typedef sequence<ReceptacleConnections> ReceptacleConnectionsSeq;
  // Entidade representante de um componente da infraestrutura
  interface BasicComponentEntity : Deployment {
    void set_args(in StringSeq args);
    void set_id(in ComponentId id);
    void set_description(in ComponentTemplate desc);
    ComponentTemplate get_description();
    void add_connection(in string recept,
                        in BasicComponentEntity provider, in string facet);
    void del_connection(in string recept,
                        in BasicComponentEntity provider, in string facet);
    ReceptacleConnectionsSeq get_connections();
    void   set_property(in string name, in string value);
    string get_property(in string name);
    string get_nickname();
    Plan   get_plan();
    Object get_facet(in string facetname) raises (NotDeployed);
  };
  // Entidade representante dos componentes do usuário
  interface ComponentEntity : BasicComponentEntity {
    void set_container(in ContainerEntity container);
    ContainerEntity get_container();
  };
  // Entidade representante do contêiner
  interface ContainerEntity : BasicComponentEntity {
    void set_node(in ExecutionNodeEntity node);
    ExecutionNodeEntity get_node();
  };
  // Entidade representante do nó de execução
  interface ExecutionNodeEntity : BasicComponentEntity {
    void set_host(in HostDescription host);
    HostDescription get_host();
    void add_known_repositories(in RepositoryEntity repo);
    void del_known_repositories(in RepositoryEntity repo);
  };
  // Entidade representate do repositório
  interface RepositoryEntity : BasicComponentEntity {
    void set_host(in HostDescription host);
    HostDescription get_host();
    void    set_privacy(in boolean private) raises (AlreadyDeployed);
    boolean get_privacy();
    void publish_desc(in ComponentTemplate structure);
    void publish_pkg(in Package pkg);
    void unpublish_desc(in core::ComponentId id);
    void unpublish_pkg(in core::ComponentId id,
                       in string lang, in string plat);
    ComponentTemplate get_desc(in core::ComponentId id);
    Package get_pkg(in core::ComponentId id,
                    in string lang, in string plat);
    ComponentDescriptionSeq search_byid(in core::ComponentId id);
};};};
```

Código A.8: Interfaces dos Serviços DeployManager e Packager

```
#include "scs.idl"
#include "deployer_entities.idl"

module scs {    module deployer {
  exception InternalProblem { string reason; };
  exception InvalidPackage { string reason; };
  exception BuildError { string reason; };
  exception PublishError {};

  // Plano de implantação
  interface Plan : Deployment {
    // planejamento
    ComponentEntity create_component();
    ContainerEntity create_container(in string lang);
    ExecutionNodeEntity create_exnode();
    RepositoryEntity create_repository();
    // inspeção
    ComponentEntitySeq get_components();
    ContainerEntitySeq get_containers();
    ExecutionNodeEntitySeq get_nodes();
    RepositoryEntitySeq get_repositories();
    Manager get_manager();
    void mark_as_admin(in boolean flag);
    ContainerEntitySeq less_used_containers();
    ExecutionNodeEntitySeq less_used_nodes();
    HostSeq less_used_hosts();
    void set_max_container_usage(in long max);
    long get_max_container_usage();
    void set_max_node_usage(in long max);
    long get_max_node_usage();
  };
  typedef sequence<Plan> PlanSeq;

  // Serviço para implantação distribuída
  interface Manager {
    Plan create_plan();
    void delete_plan(in Plan p);
    PlanSeq get_plans();
    // configuração entre componentes
    boolean connect_component(in ComponentEntity consumer,
              in string recept, in ComponentEntity provider,
              in string facet);
    // compartilhando repositórios públicos entre planos
    void add_public_repositories(in RepositoryEntity repo)
        raises (InternalProblem);
    void del_public_repositories(in RepositoryEntity repo)
        raises (InternalProblem);
    RepositoryEntitySeq get_public_repositories()
        raises (InternalProblem);
    // registro de recursos
    HostSeq get_hosts();
  };
  // Serviço para empacotamento
  interface Packager {
    Package create(in OctetSeq rockspec_stream, in OctetSeq impl_stream,
                   out ComponentTemplate tmpl)
        raises (InvalidPackage, BuildError);
    Package create_from_dir(in string rockspec_file, in string dirname,
                            out ComponentTemplate tmpl)
        raises (InvalidPackage, BuildError);
  };
};};
```

Código A.9: Algoritmo de decisão para associar um componente a um contêiner automaticamente

```
function ComponentEntity:autofit_container(available_implementations)
  local found,node,language
  local plan = self.plan
  local existent_containers = self:less_used_containers()
  for i, container in ipairs(existent_containers) do
    if plan:get_score(container) < plan:get_max_container_usage() then
      language = container:get_property("language")
      if available_implementations[language] then
        node = container:get_node()
        if not node then --nó de execução não tinha sido especificado
            node = self:autofit_exnode(available_implementations, language)
        end
        platform = node:get_property("platform")
        if available_implementations[language][platform] then
          found = container
          if not container:get_node() then
            container:set_node(node)
          end
          break
        end
      end
    else
      --continuamos buscando algum container existente que possa ser usado
    end
  end
  --não achamos nenhum container compatível, então vamos criar um novo
  if not found then
    lang_constraint = next(available_implementations)
    found = plan:create_container(lang_constraint)
    node = self:autofit_exnode(available_implementations, lang_constraint)
    found:set_exnode(node)
  end
  return found
end
```

Código A.10: Algoritmo de decisão para associar um contêiner a um nó de execução automaticamente

```
function ComponentEntity:autofit_exnode(available_implementations,
                                        lang_constraint)
  local plan = self.plan
  local found, host, platform
  local existent_nodes = self:less_used_nodes()
  for i, node in ipairs(existent_nodes) do
    if plan:get_score(node) < plan:get_max_node_usage() then
      platform = node:get_property("platform")
      if not platform then
        host = self:autofit_host(available_implementations,
                                 lang_constraint, plat_constraint)
        platform = plat_constraint or (host.resources.os..
                                       "-"..host.resources.arch)
      else
        host = node:get_host()
      end
      for j, supported_lang in ipairs(host.languages) do
        if (supported_lang == lang_constraint) and
           (available_implementations[supported_lang][platform])
        then
          found = node
          if not node:get_host() then
            node:set_host( host )
          end
          break
        end
      end
    else
      --continuamos buscando algum nó de execução que possa ser usado
    end
  end
  --não achamos nenhum nó de execução compatível, então vamos usar um outro
  if not found then
    platform_list = next(available_implementations[lang_constraint])
    plat_constraint = next(platform_list)
    found = plan:create_exnode()
    host = self:autofit_host(available_implementations,
                             lang_constraint, plat_constraint)
    found:set_host(host)
  end
  return found
end
```

Código A.11: Algoritmo de decisão para associar um nó de execução a uma máquina automaticamente

```
function ComponentEntity:autofit_host(available_implementations,
                                      lang_constraint, plat_constraint)
  local found,platform
  local existent_hosts = self:less_used_hosts()
  for i, host in ipairs(existent_hosts) do
    platform = host.resources.os .."-".. host.resources.arch
    --só existe 1 'nó de execução' por 'host'
    if platform == plat_constraint then
      for i, language in ipairs(host.languages) do
        if language == lang_constraint then
            found = host
            break
        end
      end
      if found then  break  end
    end
  end
  if not found then
      error("Não existe máquina para a plataforma solicitada:",
            plat_constraint)
  end
  return found
end
```