7
Referências Bibliográficas

[1] SZYPERSKI, C.. **Component software: beyond object-oriented programming**. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[2] GROUP OF DISTRIBUTED SYSTEMS - PUC-RIO. **OiL - The Lua Object Request Broker**. http://oil.luaforge.net/.

[3] **OMG's CORBA Website**. http://www.corba.org/.

[4] IERUSALIMSCHY, R.. **Programming in Lua, Second Edition**. Lua.Org, 2006.

[5] IERUSALIMSCHY, R.; CELES, W. ; DE FIGUEIREDO, L. H.. **Lua - The programming language**. http://www.lua.org/.

[6] MIDDLEWARE LABORATORY, PUC-RIO. **SCS - Software Component System**. http://www.tecgraf.puc-rio.br/∼scorrea/scs/.

[7] OMG: OBJECT MANAGEMENT GROUP. **CORBA Component Model**. http://www.omg.org/technology/documents/formal/components.htm, April 2006.

[8] MICROSOFT CORPORATION. **COM - Component Object Model**. http://www.microsoft.com/com/.

[9] HOARE, C. A. R.. **An axiomatic basis for computer programming**. Communications of the ACM, 12(10):576–580, 1969.

[10] LISKOV, B.; ZILLES, S.. **Programming with abstract data types**. SIGPLAN Notices, 9(4):50–59, 1974.

[11] MEYER, B.. **Applying design by contract**. Computer, 25(10):40–51, 1992.

[12] MEYER, B.. **Eiffel: The language**. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992.

[13] BEUGNARD, A.; JÉZÉQUEL, J.-M.; PLOUZEAU, N. ; WATKINS, D.. **Making components contract aware**. Computer, 32(7):38–45, 1999.

[14] CASTAGNA, G.. **Covariance and contravariance: Conflict without a cause**. ACM Trans. Program. Lang. Syst., 17(3):431–447, 1995.

[15] MEYER, B.. **Design by contract**. Em: Mandrioli, D.; Meyer, B., editors, ADVANCES IN OBJECT-ORIENTED SOFTWARE ENGINEERING, p. 1–50. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992.

[16] MITCHELL, R.; MCKIM, J. ; MEYER, B.. **Design by contract, by example**. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2002.

[17] **JML - The Java modeling language**. http://www.eecs.ucf.edu/∼leavens/JML/index.shtml.

[18] KARAORMAN, M.; HOLZLE, U. ; BRUNO, J.. **jContractor: A reflective Java library to support design by contract**. Relatório técnico, University of California at Santa Barbara, Santa Barbara, CA, USA, 1999.

[19] OMG: OBJECT MANAGEMENT GROUP. **UML 2.0 object constraint language specification**. http://www.omg.org/docs/formal/06-05-01.pdf, May 2006.

[20] MEYER, B.. **Object-oriented software construction (2nd ed.)**. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.

[21] PLÖSCH, R.. **Contracts, scenarios and prototypes**. Springer-Verlag, New York, NY, USA, 2004.

[22] FINDLER, R. B.; FELLEISEN, M.. **Contract soundness for object-oriented languages**. SIGPLAN Not., 36(11):1–15, 2001.

[23] LISKOV, B. H.; WING, J. M.. **A behavioral notion of subtyping**. ACM Trans. Program. Lang. Syst., 16(6):1811–1841, 1994.

[24] NUNES, I.. **Method redefinition - ensuring alternative behaviours**. Information Processing Letters, 92(6):279–285, 2004.

[25] **SPEC# - The Spec# programming system**. http://research.microsoft.com/specsharp/.

[26] KRAMER, R.. **iContract - The Java(tm) design by contract(tm) tool**. Em: TOOLS '98: PROCEEDINGS OF THE TECHNOLOGY OF OBJECT-ORIENTED LANGUAGES AND SYSTEMS, p. 295, Washington, DC, USA, 1998. IEEE Computer Society.

[27] PLÖSCH, R.. **Design by contract for Python**. Em: APSEC '97: PROCEEDINGS OF THE FOURTH ASIA-PACIFIC SOFTWARE ENGINEERING AND INTERNATIONAL COMPUTER SCIENCE CONFERENCE, p. 213, Washington, DC, USA, 1997. IEEE Computer Society.

[28] **Implementing design by contract for Java using dynamic proxies**. http://www.javaworld.com/javaworld/jw-02-2002/jw-0215-dbcproxy.html.

[29] BELHAOUARI, H.; PESCHANSKI, F.. **A lightweight container architecture for runtime verification**. Em: RV'08: EIGHTH WORKSHOP ON RUNTIME VERIFICATION, Budapest, Hungary, 2008.

[30] TEINIKER, E.; LECHNER, R.; SCHMOELZER, G.; KREINER, C.; KOVACS, Z. ; WEISS, R.. **Towards a contract aware corba component container**. Em: COMPSAC '05: PROCEEDINGS OF THE 29TH ANNUAL INTERNATIONAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE (COMPSAC'05) VOLUME 1, p. 545–550, Washington, DC, USA, 2005. IEEE Computer Society.

[31] CORONATO, A.; D'ACIERNO, A.; D'AMBROSIO, D. ; PIETRO, G. D.. **Supporting tools for designing-by-contract in component-based applications**. Em: PROCEEDINGS OF THE 3RD INTERNATIONAL METAINFORMATICS SYMPOSIUM (MIS 2004), volume 3511 de **Lecture Notes in Computer Science**, p. 1–13, Salzburg, Austria, junho 2004. Springer-Verlag. ISBN: 978-3-540-27328-8.

[32] JIN, Y.; HAN, J.. **Runtime validation of behavioural contracts for component software**. Em: QSIC '05: PROCEEDINGS OF THE FIFTH INTERNATIONAL CONFERENCE ON QUALITY SOFTWARE, p. 177–186, Washington, DC, USA, 2005. IEEE Computer Society.

[33] **LOOP - Class models for Lua**. http://loop.luaforge.net/.

[34] **Lua Fish**. http://lua-users.org/wiki/LuaFish.

[35] MEDEIROS, S.; IERUSALIMSCHY, R.. **A parsing machine for pegs**. Em: DLS '08: PROCEEDINGS OF THE 2008 SYMPOSIUM ON DYNAMIC LANGUAGES, p. 1–12, New York, NY, USA, 2008. ACM.

[36] IERUSALIMSCHY, R.. **LPeg - Parsing expression grammars for Lua**. http://www.inf.puc-rio.br/∼roberto/lpeg.html.

[37] IERUSALIMSCHY, R.; RODRIGUEZ, N.. **Side effect free functions in object-oriented languages**. Computer Languages, 21(3):129–146, 1995.

[38] DARVAS, A.; MÜLLER, P.. **Reasoning about method calls in interface specification**. http://www.jot.fm/issues/issue_2006_06/article3.pdf.

[39] **ICE - The Internet communications engine**. http://www.zeroc.com/ice.html.

[40] **Jironde: Transactions in Fractal**. http://jotm.objectweb.org/jironde.html.

[41] MILICIA, G.; SASSONE, V.. **The inheritance anomaly: Ten years after**. Em: SAC '04: PROCEEDINGS OF THE 2004 ACM SYMPOSIUM ON APPLIED COMPUTING, p. 1267–1274, New York, NY, USA, 2004. ACM.

Apêndice A

Interfaces do SCS

Listagem A.1: Arquivo scs.idl

```
#ifndef SCS_IDL
#define SCS_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;
            string receptacle_name;
        };
        typedef sequence<ConnectionDescription> ConnectionDescriptions;

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

        struct ComponentId {
            string name;
            unsigned long version;
```

```
        };

        typedef sequence<ComponentId> ComponentIdSeq;

        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);
            ConnectionDescription getConnection(in ConnectionId id)
                raises (InvalidConnection);
        };

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

#endif
```

Listagem A.2: Arquivo deployment.idl

```
#ifndef DEPLOYMENT_IDL
#define DEPLOYMENT_IDL

#include "scs.idl"

module scs {
    module container {
        typedef sequence<string> StringSeq;
        typedef sequence<octet> OctetSeq;
        typedef sequence<core::IComponent> IComponentSeq;

        struct ComponentId {
           string name;
           unsigned long version;
        };
        typedef sequence<ComponentId> ComponentIdSeq;

        struct ComponentHandle {
           core::IComponent cmp;
           ComponentId id;
           unsigned long instance_id;
        };

        typedef sequence<ComponentHandle> ComponentHandleSeq;

        exception ComponentNotFound{};
        exception ComponentAlreadyLoaded{};
        exception LoadFailure{};

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

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

    module execution_node {
        exception ContainerAlreadyExists{};
        exception InvalidContainer{};

        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);
            void killContainer(in string container_name)
                raises (InvalidContainer);
            core::IComponent getContainer (in string container_name);
            ContainerDescriptionSeq getContainers ();
            string getName();
        };

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

#endif
```

Listagem A.3: Arquivo events.idl

```
#ifndef EVENTS_IDL
#define EVENTS_IDL

#include "scs.idl"

module scs {
    module event_service {
       typedef any Event;

       exception NameAlreadyInUse {
          string name;
       };

       exception InvalidName {
          string name;
       };

       struct ChannelDescr {
          string name;
          core::IComponent channel;
       };
       typedef sequence<ChannelDescr> ChannelDescrSeq;

       interface EventSink {
          void push (in Event ev);
          void disconnect();
       };

       interface ChannelFactory {
          core::IComponent create (in string name) raises (NameAlreadyInUse);
          void destroy (in string name) raises (InvalidName);
       };

       interface ChannelCollection {
          core::IComponent getChannel (in string name);
          ChannelDescrSeq getAll();
       };
    };
};
#endif
```