7
Referências Bibliográficas

[1] ORACLE. **EJB - Enterprise JavaBeans Technology**. June 2011. `http://www.oracle.com/technetwork/java/javaee/ejb/index.html`. Disponível em: <http://www.oracle.com/technetwork/java/javaee/ejb/index.html>. 1

[2] OBJECT MANAGEMENT GROUP. **CORBA Component Model Specification - Version 4.0**. Needham, EUA, abr. 2006. Document: formal/2006-04-01. 1

[3] BRUNETON, E. et al. The fractal component model and its support in java. **Software: Practice and Experience**, John Wiley & Sons, Ltd., v. 36, n. 11-12, p. 1257–1284, 2006. ISSN 1097-024X. Disponível em: <http://dx.doi.org/10.1002/spe.767>. 1

[4] MICROSOFT. **COM - Componet Object Model Technology**. August 2010. `http://www.microsoft.com/com`. Disponível em: <http://www.microsoft.com/com>. 1

[5] COULSON, G. et al. A component model for building systems software. In: HAMZA, M. H. (Ed.). **IASTED Conf. on Software Engineering and Applications**. [S.l.]: IASTED/ACTA Press, 2004. p. 684–689. ISBN 0-88986-425-X. 1

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

[7] SHAW, M.; GARLAN, D. **Software Architecture: Perspective on an Emerging Discipline**. [S.l.]: Prentice-Hall, 1996. 1

[8] MEHTA, N. R.; MEDVIDOVIC, N.; PHADKE, S. Towards a taxonomy of software connectors. In: **Proceedings of the 22nd international conference on Software engineering**. New York, NY, USA: ACM, 2000. (ICSE '00), p. 178–187. ISBN 1-58113-206-9. Disponível em: <http://doi.acm.org/10.1145/337180.337201>. 1, 3.1, 3.2.2, 4.5

[9] SHAW, M.; GARLAN, D. **Software architecture: perspectives on an emerging discipline**. Prentice Hall, 1996. (An Alan R. Apt book). ISBN 9780131829572. Disponível em: <http://books.google.com.br/books?id=5KlQAAAAMAAJ>. 1

[10] BIRRELL, A. D.; NELSON, B. J. Implementing remote procedure calls. **ACM Trans. Comput. Syst.**, ACM, New York, NY, USA, v. 2, p. 39–59, February 1984. ISSN 0734-2071. Disponível em: <http://doi.acm.org/10.1145/2080.357392>. 1

[11] OBJECT MANAGEMENT GROUP. **The Common Object Request Broker Architecture (CORBA) Specification - Version 3.1**. [S.l.], jan. 2008. Document: formal/2008-01-04. 1

[12] OBJECT MANAGEMENT GROUP. **Streams for CCM Specification - 2005-07-01 - Work in progress**. [S.l.], July 2005. Document: (ptc/2005-07-01). 1, 2.1, 3.2.3, 4.3.2

[13] DRUCKER, P. **The Age of Discontinuity: Guidelines to Our Changing Society**. Transaction Publishers, 1992. (Transaction books). ISBN 9781560006183. Disponível em: <http://books.google.com.br/books?id=1Zp7_rJ1vcMC>. 1

[14] LIMA, M. Julia de et al. Csbase: A framework for building customized grid environments. In: **Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises**. Washington, DC, USA: IEEE Computer Society, 2006. p. 187–194. ISBN 0-7695-2623-3. Disponível em: <http://portal.acm.org/citation.cfm?id=1264357.1264723>. 1

[15] Tecgraf, PUC-Rio. **SCS - A Lightweight Software Component System**. July 2011. `http://www.tecgraf.puc-rio.br/scs`. Disponível em: <http://www.tecgraf.puc-rio.br/scs>. 1, 3.1

[16] AUGUSTO, C. E. L. et al. SCS - Sistema de Componentes de Software. Departamento de Informática - Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio). 2007. Disponível em: <https://jira.tecgraf.puc-rio.br/confluence/download/attachments/18612229/scsOverview.pdf>. 1, 3.1

[17] JUNIOR, A. A. B. **Implantação de Componentes de Software Distribuídos Multi-Linguagem e Multi-Plataforma**. Dissertação

(Mestrado) — Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ, Brasil, ago. 2009. 1, 2.3, 3.1.3, 4.1, 5.1, 6

[18] STOINSKI, F. Towards a component model for universal data streams. In: **ISCC**. IEEE Computer Society, 2003. p. 1178–1184. ISBN 0-7695-1961-X. Disponível em: <http://doi.ieeecomputersociety.org/10.1109/ISCC.2003.1214273>. 2.1

[19] STOINSKI, F. The CORBA component model streaming extension. In: HAMZA, M. H. (Ed.). **IASTED International Conference on Software Engineering, part of the 22nd Multi-Conference on Applied Informatics, Innsbruck, Austria, February 17-19, 2004**. [S.l.]: IASTED/ACTA Press, 2004. p. 645–651. ISBN 0-88986-381-4. 2.1

[20] STOINSKI, F. Engineering streaming applications with the corba component model streaming extension. In: HAMZA, M. H. (Ed.). **IASTED International Conference on Software Engineering, part of the 22nd Multi-Conference on Applied Informatics, Innsbruck, Austria, February 17-19, 2004**. [S.l.]: IASTED/ACTA Press, 2004. p. 154–159. ISBN 0-88986-381-4. 2.1

[21] FREED, N.; BORENSTEIN, N. **Multipurpose internet mail extensions (MIME) part two: Media types**. [S.l.], 1996. 2.1, 2.2, 4.3.4

[22] OBJECT MANAGEMENT GROUP. **Audio/Video Stream Specification**. [S.l.], June 1998. Document: (formal/2000-01-03). 2.2, 4.3.2

[23] RAFAELSEN, H.; ELIASSEN, F.; MUSUNOORI, S. Towards self-organizing distribution structures for streaming media. In: MEERSMAN, R.; TARI, Z. (Ed.). **On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE**. Springer Berlin - Heidelberg, 2006, (Lecture Notes in Computer Science, v. 4276). p. 1825–1842. ISBN 978-3-540-48274-1. 10.1007/11914952_53. Disponível em: <http://dx.doi.org/10.1007/11914952_53>. 2.3, 3

[24] ELIASSEN, F.; MEHUS, S. Type checking stream flow endpoints. In: SPRINGER VERLAG. **Middleware'98: IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing**. [S.l.], 1998. p. 305. 2.3

[25] MUNGEE, S.; SURENDRAN, N.; SCHMIDT, D. The design and performance of a corba audio/video streaming service. In: **System Sciences,**

1999. HICSS-32. Proceedings of the 32nd Annual Hawaii International Conference on. [S.l.: s.n.], 1999. Track8, p. 14 pp. 2.4

[26] HENNING, M. Zeroc, the rise and fall of corba. **Component Technologies**, v. 4, n. 5, 2006. 2.4

[27] CHAPPELL, D. The trouble with corba. **Object News (May 1998) http://www. davidchappell. com/articles/article_Trouble_CORBA. html**, 1998. 2.4

[28] Tecgraf, PUC-Rio. **Openbus - Enterprise Integration Application Middleware**. December 2009. `http://www.tecgraf.puc-rio.br/openbus`. Disponível em: <http://www.tecgraf.puc-rio.br/openbus>. 3.1

[29] AUGUSTO, C. E. L. **Uma Infra-Estrutura para a Execução Distribuída de Componentes de Software**. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ, Brasil, set. 2008. 3.1.2, 6

[30] THIES, W.; KARCZMAREK, M.; AMARASINGHE, S. P. StreaMIT: A language for streaming applications. In: **Proceedings of the 11th International Conference on Compiler Construction**. London, UK: Springer-Verlag, 2002. (CC '02), p. 179–196. ISBN 3-540-43369-4. Disponível em: <http://portal.acm.org/citation.cfm?id=647478.727935>. 3.2.1

[31] WELSH, M.; CULLER, D.; BREWER, E. SEDA: an architecture for well-conditioned, scalable Internet services. **Operating Systems Review**, v. 35, n. 5, p. 230–243, dez. 2001. ISSN 0163-5980. 4.1

[32] BRECH, A. C. T.; MOSBERGER, D. **μserver Project**. July 2011. `http://userver.uwaterloo.ca/`. Disponível em: <http://userver.uwaterloo.ca/>. 4.1

[33] BEHREN, R. von et al. Capriccio: scalable threads for internet services. In: **Proceedings of the nineteenth ACM symposium on Operating systems principles**. New York: ACM Press, 2003. (Operating Systems Review, v. 37, 5), p. 268–281. ISSN 0163-5980. 4.1

[34] STEVENS, W.; FENNER, B.; RUDOFF, A. **UNIX Network Programming: The Sockets Networking API**. [S.l.]: Addison-Wesley Professional, 2003. 4.1

[35] JOUBERT, P. et al. High-performance memory-based web servers: Kernel and user-space performance. In: **USENIX Annual Technical Conference, General Track**. [S.l.: s.n.], 2001. p. 175–187. 4.1

[36] SILVEIRA, P. da S. **Projeto e Implementação de Interfaces Coletivas em um Middleware orientado a Componentes de Software**. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ, Brasil, jan. 2011. 4.3.1

[37] Tecgraf, PUC-Rio. **FTC - File Transfer Channel**. July 2011. `https://jira.tecgraf.puc-rio.br/confluence/display/OpenBus/Download+FTC`. Disponível em: <http://www.tecgraf.puc-rio.br/scs>. 5.1.1, 6

[38] ALLCOCK, W. et al. The globus striped gridftp framework and server. In: IEEE COMPUTER SOCIETY. **Proceedings of the 2005 ACM/IEEE conference on Supercomputing**. [S.l.], 2005. p. 54. 5.1.1

[39] (NLANR)., D. A. S. T. D. at the National Laboratory for A. N. R. **Iperf**. April 2012. `http://iperf.sourceforge.net/`. Disponível em: <http://iperf.sourceforge.net/>. 5.2.2

A
Listagem de Códigos

A.1
IDL do SCS-STREAMS consolidada

```
#ifndef SCS_STREAMS_IDL
#define SCS_STREAMS_IDL

#include <scs.idl>

module scs {
    module streams{
        typedef scs::core::ConnectionDescriptions ConnectionDescriptions;
        typedef scs::core::ConnectionDescription ConnectionDescription;
        typedef scs::core::ConnectionId ConnectionId;
        typedef unsigned long PortId;
        typedef sequence<string> StringList;

        struct NameValue {
          string name;
          any value;
        };
        typedef sequence<NameValue> NameValueList;

        enum ConfigureEvent { CONNECTION_REQUEST,
                              DISCONNECTION_REQUEST,
                              START_REQUEST,
                              STOP_REQUEST,
                              UPDATE_REQUEST,
                              CUSTOM_EVENT};

        struct SinkDescription {
            PortId id;
            string name;
            string content_type;
            string transport_type;
        };

        struct SourceDescription {
            PortId id;
            string name;
            string content_type;
            string transport_type;
            boolean is_multiplex;
        };

        /** \brief Exceção genérica do módulo scs.streams*/
```

```
43          exception SCSStreamException {};
44          /** \brief Exceção que sinaliza um erro no
45              estabelecimento do protocolo de transporte */
46          exception TransportFailed {};
47          /** \brief Exceção que sinaliza que o conector de
48              fluxo de dados não esta conectado */
49          exception NotConnected {};
50          /** \brief Exceção que sinaliza a ausência de um
51              fluxo de dados */
52          exception NoStream {};
53          /** \brief Exceção que sinaliza que a transmissao
54              de dados ja foi iniciada */
55          exception StreamAlreadyStarted{};
56          /** \brief Exceção que sinaliza que uma porta não
57              esta habilitado a receber conexões*/
58          exception SourcePortDisabled{};
59          /** \brief Exceção que sinaliza a existência de um
60              fluxo de dados semelhante */
61          exception UnsupportedStreamType {
62               /** \brief Tipos de fluxo de dados não suportados */
63              string unsupported_stream_type;
64          };
65          exception UnsupportedTransport{
66              /** \brief Tipos de fluxo de dados não suportados */
67              string unsupported_transport_type;
68          };
69
70          typedef sequence<SinkDescription> SinkDescriptions;
71          typedef sequence<SourceDescription> SourceDescriptions;
72
73          interface IStreamPort {
74              void disconnect( in ConnectionId id )
75                  raises (NotConnected, scs::core::InvalidConnection,
76                          SCSStreamException) ;
77
78              void update( in ConnectionId id, in NameValueList parameters )
79                  raises (NotConnected, scs::core::InvalidConnection,
80                          SCSStreamException) ;
81          };
82
83          interface ISource; // Forward declaration
84
85          interface ISink : IStreamPort {
86              readonly attribute SinkDescription description;
87
88              ConnectionDescription connect( in ISource port,
89                                             in NameValueList parameters )
90                  raises (scs::core::ExceededConnectionLimit,
91                          scs::core::InvalidConnection,
92                          scs::core::AlreadyConnected,
93                          UnsupportedStreamType,
94                          TransportFailed,
95                          SourcePortDisabled,
96                          UnsupportedTransport,
97                          SCSStreamException);
98
99              boolean configure( in ConfigureEvent event,
100                                 inout NameValueList parameters,
101                                 in ISource port )
```

```
102                 raises (scs::core::ExceededConnectionLimit,
103                         scs::core::InvalidConnection,
104                         scs::core::AlreadyConnected,
105                         NotConnected,
106                         UnsupportedStreamType,
107                         TransportFailed,
108                         SourcePortDisabled,
109                         NoStream,
110                         StreamAlreadyStarted,
111                         SCSStreamException);
112
113             ConnectionDescription getConnection();
114         };
115
116         interface ISource : IStreamPort {
117             readonly attribute SourceDescription description;
118             readonly attribute boolean enabled;
119
120             void start( in ConnectionId id )
121                 raises (NotConnected, scs::core::InvalidConnection,
122                         SCSStreamException, StreamAlreadyStarted);
123
124             void stop( in ConnectionId id )
125                 raises (NotConnected, scs::core::InvalidConnection,
126                         NoStream, SCSStreamException);
127
128             boolean configure( in ConfigureEvent event,
129                                inout NameValueList parameters,
130                                in ISink port )
131                 raises (scs::core::ExceededConnectionLimit,
132                         scs::core::InvalidConnection,
133                         scs::core::AlreadyConnected,
134                         NotConnected,
135                         UnsupportedStreamType,
136                         TransportFailed,
137                         SourcePortDisabled,
138                         NoStream,
139                         StreamAlreadyStarted,
140                         UnsupportedTransport,
141                         SCSStreamException);
142
143             ConnectionDescriptions getConnections();
144         };
145
146         interface IStreamComponent {
147             SinkDescriptions getSinks();
148             SourceDescriptions getSources();
149             ISink getSink(in PortId id);
150             ISink getSinkByName(in string name );
151             ISource getSource(in PortId id);
152             ISource getSourceByName(in string name );
153         };
154     };
155 };
156 #endif //SCS_STREAMS_IDL
```

Código A.1: Interfaces do SCS-STREAMS

A.2
Código gerado pela aplicação de implantação

```
require "table_show"
local oil = require "oil"
oil.verbose:level(1)
-- local of the corba idl files installation
local SCS_HOME = assert(os.getenv("SCS_HOME"),
                 "Missing the system variable called SCS_HOME")
-- client orb
orb = oil.init()
oil.orb = orb
orb:loadidlfile(SCS_HOME.."/idl/deployer.idl")


oil.main(function()
  -- obtaining the remote proxies to the DeployManager
  local manager = orb:newproxy("corbaloc::localhost:2500/Deployer_1",
            "IDL:scs/deployer/Manager:1.0")

  -- defining a host to place the repository
  local importer = require "scs.deployer.descriptorhelper"
  local repository = manager:get_public_repositories()[1]

  local deploy_result = {}

  -- creating a deployment plan in the manager
  local plan = manager:create_plan()
  deploy_result.plan_name = plan:get_nickname()
  print("[info] plan created with id='" .. deploy_result.plan_name .. "'")

  deploy_result.components = {}


  local machine1 = importer.search( "hosts", "machine1" )
  local machine1_exnode = plan:create_exnode()
  machine1_exnode:set_host( machine1 )
  machine1_exnode:add_known_repositories( repository )
  local machine1_container = plan:create_container( "java" )
  machine1_container:set_node( machine1_exnode )


  local source_machine1 = plan:create_component()
  local source_machine1_id = {
    name = "SourceTcpStream",
    major_version = 1,
    minor_version = 0,
    patch_version = 0,
    platform_spec = "linux-x86_64"
  }
  source_machine1:set_id( source_machine1_id )
  source_machine1:set_container( machine1_container )


  local machine2 = importer.search( "hosts", "machine2" )
  local machine2_exnode = plan:create_exnode()
  machine2_exnode:set_host( machine2 )
  machine2_exnode:add_known_repositories( repository )
  local machine2_container = plan:create_container( "java" )
```

```
57  machine2_container:set_node( machine2_exnode )
58
59
60  local sink_machine2 = plan:create_component()
61  local sink_machine2_id = {
62    name = "SinkTcpStream",
63    major_version = 1,
64    minor_version = 0,
65    patch_version = 0,
66    platform_spec = "linux-x86_64"
67  }
68  sink_machine2:set_id( sink_machine2_id )
69  sink_machine2:set_container( machine2_container )
70
71
72  -- deployment step
73  if not oil.pcall(plan.deploy,plan) then
74    print("[error] deployment failed!")
75    plan:undeploy()
76    os.exit(1)
77  end
78
79  -- activation step, this call startup in all components
80  print("[info] Activating plan...")
81  plan:activate()
82  print("[info] Plan activated!")
83
84
85  local source_machine1_icomp =
86  orb:narrow(source_machine1:get_facet("IComponent"), "IDL:scs/core/
        IComponent:1.0")
87  deploy_result.components['source_machine1'] = orb:tostring(
        source_machine1_icomp)
88
89
90  local sink_machine2_icomp =
91  orb:narrow(sink_machine2:get_facet("IComponent"), "IDL:scs/core/
        IComponent:1.0")
92  deploy_result.components['sink_machine2'] = orb:tostring(
        sink_machine2_icomp)
93
94
95  print("[info] Connecting components...")
96  orb:loadidlfile(SCS_HOME.."/idl/scs_streams.idl")
97
98
99  local sink_machine2_proxy =
100   orb:narrow(sink_machine2:get_facet("IStreamComponent"),
101              "IDL:scs/streams/IStreamComponent:1.0")
102
103 local sink_machine2_input_port = sink_machine2_proxy:getSinkByName("input
        ")
104
105 local source_machine1_proxy =
106   orb:narrow(source_machine1:get_facet("IStreamComponent"),
107              "IDL:scs/streams/IStreamComponent:1.0")
108
109 local source_machine1_output_port = source_machine1_proxy:getSourceByName
        ("output")
```

```
110
111   print("[info] Connecting sink_machine2-input to source_machine1-output")
112   local conn1 = sink_machine2_input_port:connect(
          source_machine1_output_port, {})
113
114   print("[info] Components ports connected!")
115   print("[info] Starting transmission on source_machine1-output ...")
116   source_machine1_output_port:start(conn1.id)
117   print("[info] Transmission started!")
118
119
120   print("[info] Components connected!")
121
122   oil.writeto('./deploy_result.lua', table.show(deploy_result, '
          deploy_result'))
123   print("[info] flow plan deployed successfully!")
124 end)
```

Código A.2: Plano de implantação.