6 Referências Bibliográficas

1. J. BAILEY, Y. BAKOS. **An Exploratory Study of the Emerging Role of Electronic Intermediaries**. International Journal of Electronic Commerce 1(3), Spring 1997.
2. Y. BAKOS. **The Emerging Role of Electronic Marketplaces on the Internet**. CACM, Agosto 1998.
3. M. BARBACCI. **Quality Attributes**. Technical Report, CMU/SEI-95-TR-021, Dezembro, 1995.
4. G. BOOCH, J. RUMBAUGH. **Unified Modeling Language – User Guide**. Addison-Wesley, 1999.
5. J. BRADSHAW**. An Introduction to Software Agents**. em: *Software Agent*s, J. Bradshaw (Ed.)
(ed.), American Association for Artificial Intelligence/MIT Press, 1997.
6. D. BRUGALI AND K. SYCARA. **A Model for Reusable Agent Systems**. em: *Implementing Application Frameworks – Object-Oriented Frameworks at Wor*k, M. Fayad et al. (Ed.), John Wiley & Sons, 1999.
7. F. BUSCHMANN et al. **Pattern-Oriented Software Architecture: A System of Patterns**, John Wiley & Sons, 1996.
8. L. E. BUZATO, C. M. F. RUBIRA (UNICAMP, BRAZIL) AND M. L. B. LISBOA (UFRGS, Brazil) **A Reflective Object-oriented Architecture for Developing Fault-tolerant Software** JBCS, Vol 4, Num 2, Julho 1997
9. G. CABRI, L. LEONARDI, F. ZAMBORNELLI, **MARS: A Programmable Coordination Architecture for Mobile Agents**, IEEE Internet Computing, Vol. 4, No. 4, pp. 26-35, Julho-Agosto 2000.
10. N. CARRIERO, D. GALERNTER, **Linda in Context**, Communications of the ACM, vol. 32, No. 4, (1989), pp 444-458
11. L. ERMAN, F. HAYES-ROTH, V. LESSER, D. REDDY. **The HEARSAY-II speech-understanding system: Integrating knowledge to resolve uncertainty**. Computing Surveys 12(2): 213-253.
12. L. FERREIRA, C. RUBIRA, **The Reflective State Pattern**. Proceedings of the 5th Patterns Languages of Programs Conference (PLoP'98), Agosto 98, Monticello, EUA.
13. M. FOWLER et al. **Refactoring: Improving the Design of Existing Code** Addison-Wesley Pub Co; 1st edition (Junho 28, 1999)
14. E. FREEMAN, S. HUPFER, K. ARNOLD, **JavaSpaces(TM) Principles, Patterns and Practice**, Addison-Wesley Pub Co, Junho 1999
15. A. FUGGETTA, G. PICCO, G. VIGNA **Understanding Code Mobility** IEEE Transactions on Software Engineering v 24/5, 1998
16. D. GELERNTER, **Generative Communication in Linda** ACM Transactions on Programming Languages and Systems, vol. 7 - No.1, pp 80-112, 1985
17. E. GAMMA et al. **Design Patterns: Elements of Reusable Object-Oriented Software**. Addison-Wesley, 1995.
18. A. GARCIA, C. CHAVEZ, O. SILVA, V. SILVA, C. LUCENA. **Promoting Advanced Separation of Concerns in Intra-Agent and Inter-Agent Software Engineering**. Workshop on Advanced Separation of Concerns in Object-oriented Systems (ASoC) at OOPSLA'2001, Tampa Bay, EUA, Outubro 2001

19. A. GARCIA, C. RUBIRA. **A Architectural-based Reflective Approach to Incorporating Exception Handling into Dependable Software**. em: *Advances in Exception Handling Techniques*, A. Romanovsky et al (Ed.). Springer-Verlag, LNCS-2022, Abril 2001, pp. 189-206.
20. A. GARCIA, C. RUBIRA. **A Unified Meta-Level Software Architecture for Sequential and Concurrent Exception Handling**. The Computer Journal, Special Issue on High Assurance Systems Engineering, Janeiro, 2002.
21. A. GARCIA, V. SILVA, C. CHAVEZ, C. LUCENA. **Engineering Multi-Agent Systems with Aspects and Patterns**. Journal of the Brazilian Computer Society, Special Issue on Software Engineering and Databases, Agosto 2002.
22. A. GARCIA, V. SILVA, C. LUCENA, R. MILIDIÚ. **An Aspect-Based Approach for Developing Multi-Agent Object-Oriented Systems**. XXI Simpósio Brasileiro de Engenharia de Software, Rio de Janeiro, Brasil, Outubro 2001, pp. 177-192.

23. A. GARCIA; C. LUCENA **Software Engineering for Large-Scale Multi-Agent Systems**. (Post-Workshop Report) ACM Software Engineering Notes, Vol. 27, Num 5, Setembro 2002, pp. 82-88

24. A. GARCIA, C. LUCENA, D. COWAN **Agents in Object-Oriented Software Engineering**. Technical Report CS-2001-07, Computer Science Department, University of Waterloo, Waterloo, Canada, Fevereiro 2001.
25. A. GARCIA; C. J. LUCENA; D. COWAN **Agents in Object-Oriented Software Engineering**. Software: Practice & Experience, Elsevier, 2003 (a ser publicado).
26 J. GOSLING, B. JOY, G. STEELE, G. BRACHA, **The Java Language Specification,** Second Edition, Addison Wesley, Junho 2000
27. R. GUTTMAN, A. MOUKAS, P. MAES. **Agent Mediated Electronic Commerce: A Survey**. Knowledge Engineering Review, Junho, 1998.
28. M. HUHNS, L. STEPHENS. **Multiagent Systems and Societies of Agents", in *Multiagent Systems – A Modern Approach to Distributed Artificial Intelligence***, ed. G. Weiss, MIT Press, 2000
29. Java Object Serialization Specification http://java.sun.com/products/jdk/1.2/docs/guide/serialization/spec/serialTOC.html
30. N. JENNINGS AND M. WOOLDRIDGE. **Agent-Oriented Software Engineering**. em:, Handbook of Agent Technology, J. Bradshaw (Ed.) AAAI/MIT Press, 2000.
31.. N. KARNIK, A. TRIPHATHI, **Security in the Ajanta Mobile Agent System**, Software - Practice and Experience, Janeiro 2001.
32. N. KARNIK, A. TRIPATHI. **Design Issues in Mobile-Agent Programming Systems**. IEEE Concurrency, vol. 6, n. 3, 1998, pp.52-61.
33. E. KENDALL, P. KRISHNA, C. PATHAK, C. SURESH, **A Framework for Agent Systems**, em *Implementing Applications Frameworks: Object Oriented Frameworks at Work*, M. Fayad et al (Ed.), John Wiley & Sons, 1999.

34. D. LANGE AND M. OSHIMA. **Programming and Developing Java Mobile Agents with Aglets**, Addison-Wesley, Agosto 1998.

35. T. LEHMAN, S. MCLAUGHRY, P. WYCKOFF. **TSpaces: The Next Wave**. Hawaii International Conference on System Sciences (HICSS-32), Janeiro, 1999.

36. D. LEA **Concurrent Programming in Java(TM): Design Principles and Patterns** Addison-Wesley Pub Co, 2nd edition (Novembro, 1999)

37. M. L. B. LISBOA **A New Trend on the Development of Fault-Tolerant Application: Software Meta-Level Architectures** JBCS Vol 4 Num 2 Julho 1997

38. P. MAES. **Concepts and Experiments in Computational Reflection**. ACM SIGPLAN Notices, 22(12), pp 147-155, 1987

39. T. MALONE, K. Crowston. **The Interdisciplinary Study of Coordination**. *ACM Computing Surveys 26*, 1 (Março), 87-119.

40. N. MINSKY, V. UNGUREANU**. Law-Governed Interaction: A Coordination and Control Mechanism for Heterogeneous Distributed Systems.** ACM Transactions on Software Engineering and Methodology, Vol. 9, No. 3, Julho 2000, pp. 273-305.

41. A. OLIVA, I. GARCIA, L. BUZATO, **The reflexive architecture of Guaraná**. Technical Report IC-98-14, Instituto de Computação, Unicamp, Campinas, Brasil, Abril, 1998

42. OBJECT MANAGEMENT GROUP – Agent Platform Special Interest Group. **Agent Technology – Green Paper – Version 1.0,** OMG, Setembro 2000.

43. A. OMICINI, F. ZAMBONELLI, **TuCSoN: a Coordination Model for Mobile Information Agents**. 1st International Workshop on Innovative Internet Information Systems (IIIS'98), Pisa (I), Junho 1998

44. C. E. PERKINS, **Mobile IP,** IEEE Communications Magazine, vol. 35, no. 5, pp. 84-99, Maio 1997

45. P. RIBEIRO, **Modelagem e implemantação OO de sistemas multi-agentes** Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, Rio de Janeiro, Brasil, 2001

46. P. RIPPER, M. Fontoura, A. Neto, and C. Lucena. **V-Market: A Framework for e-Commerce Agent Systems.** World Wide Web, Baltzer Science Publishers, 3(1), 2000.

47. J. SARDINHA, P. RIBEIRO, R. MILIDIÚ, C. LUCENA **An Object-Oriented Framework for Building Software Agents**. Journal Of Object Technology. Estados Unidos, 2002.

48. R. SENRA **Programação Reflexiva sobre o Protocolo de Meta-Objetos Guaraná** Dissertação de Mestrado, Instituto de Computação Unicamp, Campinas, Brasil, Dezembro de 2001

49. M. SHAW, D. GARLAN. **Software Architecture: Perspectives on an Emerging Discipline**, Prentice Hall, 1996.

50. O. SILVA, A. GARCIA, C. LUCENA, **A Unified Software Architecture for System-Level and Agent-Level Dependability in Multi-Agent Object-Oriented Systems**, 7th ECOOP Workshop on Mobile Objects Systems, Budapeste, Hungria, Junho 2001

51. O. SILVA, A. GARCIA, C. LUCENA**, T-Rex: A Reflective Tuple Space Environment for Dependable Mobile Agent Systems**. III Workshop de Comunicação Sem Fio e Computação Móvel WCSF at IEEE MWCN, Recife, Brasil, Agosto 2001
52. SILVA, O.; GARCIA, A; LUCENA, C. J. **The Reflective Blackboard Architectural Pattern for Developing Large Scale Multi-Agent Systems**. 1st International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS 2002) at ICSE 2002, Orlando, EUA, Maio 2002.
53. O. SILVA; A. GARCIA; C. J. LUCENA. **The Reflective Blackboard Pattern: Architecting Large-Scale Multi-Agent Systems**. In:. *Software Engineering for Large-Scale Multi-Agent Systems*. A. Garcia, et al (Ed.) Springer-Verlag, LNCS, Janeiro 2003 (a ser publicado).
54. O. SILVA; A. GARCIA; C. J. LUCENA **The Reflective Blackboard Pattern** Monografias em Ciência da Computação, v 24 , PUC-Rio, Rio de Janeiro, Brasil, Outubro 2002
55. O. SILVA, D. ORLEAN, F. FERREIRA AND C. LUCENA. **A Shared-Memory Agent-Based Framework for Business-to-Business Applications**. Proceedings of the 2001 Information Resources Management Association International Conference IRMA'2001) - Web Engineering for E-Commerce Applications, Maio 2001
56. O. SILVA; C. LUCENA **Um Ambiente Para Aplicações de Agentes Móveis Baseado em Arquiteturas Reflexivas de Espaços de Tuplas**, Workshop de Teses em Engenharia de Software (WTES) do Simpósio Brasileiro de Engenharia de Software (SBES), Rio de Janeiro, Brasil, Outubro 2001.
57. V. SILVA, O. SILVA, A. GARCIA, C. CHAVEZ, C. LUCENA **Separation of Concerns for Multi-agent Software Engineering**. Poster Session at OOPSLA'2001, Tampa Bay, Florida, USA, Outubro 14 – 18, 2001.
58. GRUPO SOC+AGENTS http://www.teccomm.les.inf.puc-rio.br/SoCagents
59. M. TSVETOVATYY, M. GINI, B. MOBASHER, Z. WIECKOWSKI. MAGMA: **An Agent Based Virtual Market for Electronic Commerce. International Journal of Applied Artificial Intelligence**, Setembro 1997.
60. H. YOSHIOKA, Y. TAHARA, A. OHSUGA, S. HONIDEN. **Security for Mobile Agents**. In Proc. of the 1st International Workshop on Agent-Oriented Software Engineering at ICSE 2000, Limerick (IR), Junho 2000.
61. M. M. WOOLDRIDGE, N. JENNINGS. **Intelligent agents: Theory and practice**. The Knowledge Engineering Review, 10(2):115{152, 1995.]
62. F. ZAMBONELLI, G. CABRI, L. LEONARDI. **Developing Mobile Agent Organizations: A Case Study in Digital Tourism**. Proceedings of the 3rd International Symposium on Distributed Objects & Applications (DOA) 2001, Rome (I), Setembro 2001.
63. F. ZAMBONELLI, N. JENNINGS, A. OMICINI, M. WOOLDRIDGE, **Agent-Oriented Software Engineering for Internet Applications**, em Coordination of Internet Agents: Models, Technologies and Applications, Springer, 2000

Apêndice I – Código Fonte das Meta-Tuplas e Reações Presentes no Marketplace Implementado

```
public class MessageProtectionMetaTuple extends MetaTuple{
    public MessageProtectionMetaTuple( Operation op )
    {
        /*Associa a reação "MessageProtectionReaction" à operação
          específica "op" realizada por qualquer agente sobre qualquer men-
          sagem.
         A operação "op" em geral é especificada como a leitura (READ) ou
         retirada (TAKE) de mensagens.
        */
        super( new MessageProtectionReaction() ,
               op ,
               AgentID.class ,
               TRexMessage.class );
    }
}
```

Fragmento de Código 5 – Meta-tupla de proteção à leitura de mensagens por agentes que não são seus destinatários

```
public SuperTuple react( SuperTuple t , Operation op , AgentID id )
{
   SuperTuple retVal = null;
   //a tupla que origina esta reação sempre será uma mensagem
   TRexMessage msg = ( TRexMessage ) t;
   //recupera o destinatário da mensagem
   AgentID receiver = msg.getReceiver();
   //se o destinatário da mensagem for realmente o agente que
   //solicitou sua leitura ou retirada a mensagem será
   //retornada para ele
   if( receiver.equals( id ) )
   {
      retVal = msg;
   }
   //caso contrário lhe será retornado "null" indicando que a
   //tupla procurada não existe para a visão do agente
   else
   {
      retVal = null;
      //Caso a operação solicitada tenha sido a retirada da tupla
      //ela será reescrita no ambiente (espaço base)
      if ( op.equals( Operation.TAKE ) )
      {
         TRexAccessor acc = SingletonTRexAccessor.getInstance();
         acc.getBaseSpace().write( msg );
      }

   }
   return retVal;

}
```

Fragmento de Código 6 – Reação (MessageProtectionReaction) associada à Meta-Tupla de proteção à leitura

```
public class PersistenceMetaTuple extends MetaTuple
{
    public PersistenceMetaTuple( AgentID messageSender ,
                                 AgentID persistentAgent )
    {
        /*Associa a reação "PersistenceReaction" à escrita realizada de
          qualquer mensagem endereçada ao agente que se tornou persistente
          (persistentAgent) realizada por um agente específico
          (messageSender).
          Este agente específico deve ser o agente cujas mensagens devem
          acordar o agente persistente.*/
         super( new PersistenceReaction( ) ,
                Operation.WRITE ,
                messageSender ,
                new TRexMessage( persistentAgent ) );
    }

}
```

Fragmento de Código 7 – Meta-Tupla para a ativação de agentes persistentes

```
public class PersistenceReaction implements Reaction
{

    public SuperTuple react(SuperTuple t, Operation op, AgentID id)
    {

        //adquirindo acesso ao TRexAccessor
        TRexAccessor accessor = SingletonTRexAccessor.getInstance();
        //a tupla que origina esta reação é sempre uma TRexMessage
        TRexMessage msg = ( TRexMessage ) t;
        //recupera o destinatário da mensagem
        //(agente que deve ser acordado)
        AgentID receiver = msg.getReceiver();
        //ativa (acorda) o agente persistente
        accessor.activatePersistentAgent( receiver );
        return t;
    }
}
```

Fragmento de Código 8 – Reação associada à meta-tupla de ativação de agentes persistentes

```
public class EconomyMetaTuple extends MetaTuple{

    /** Creates new EconomyMetaTuple */
    public EconomyMetaTuple( String item , int maxQuantity )
    {
        /* Associa a reação “EconomyReaction” à operação de retirada (TAKE)
        de uma proposta (representada pela ProposalTuple) efetuada por
        qualquer agente. Esta operação é de fato a tentativa de compra de um
        item*/
         super( new EconomyReaction( maxQuantity ) ,
                Operation.TAKE ,
                AgentID.class ,
                new ProposalTuple( item ) );
    }

}
```

Fragmento de Código 9 - Meta-Tupla responsável pela implementação da estratégia de racionamento de itens

```
public class EconomyReaction implements Reaction
{
    private int maxQty;
    private Hashtable qtyPerAgentHashtable;

    public EconomyReaction( int maxQuantity )
    {
        maxQty = maxQuantity;
    }

    public SuperTuple react( SuperTuple t , Operation op , AgentID id )
    {
        SuperTuple retVal = null;
        //verifica se o agente possui a permissão para comprar o item
        boolean buyPermission = checkBuyPermission( id );
        //caso o agente não possua a permissão a retirada da proposta de
        //venda é impedida, sendo retornado para ele o valor "null"
        if ( buyPermission == false )
        {
            TRexAccessor acc = SingletonTRexAccessor.getInstance();
            acc.getBaseSpace().write( t );
            retVal = null;
        }
        else
        //se o agente possui a permissão de compra ele recebe a proposta
        //e a quantidade comprada do item é armazenada na Hashtable
        {
            retVal = t;
            Integer qty = qtyPerAgentHashtable.get( id );
            //caso o agente ainda não tenha comprado o item a quantidade
            //inserida na Hashtable é igual a 1
            if ( qty == null )
            {
                qtyPerAgentHashtable.put( id , new Integer( 1 ) );
            }
            else
            //caso o agente já tenha comprado o item, a quantidade
            //comprada é incrementada e atualizada na Hashtable
            {
                qtyPerAgentHashtable.put( id , qty.intValue()++ );
            }
            //a ordem de compra relativa ao item comprado
            //também é enviada para o vendedor
            sendPurchaseOrder( t );
        }
        return retVal;
    }
    private boolean checkBuyPermission( AgentID id )
    {
        /*verifica se um agente possui a permissão para comprar o item em
          questão (definido pela meta-tupla). A verificação é baseada na
          quantidade do item já comprada pelo agente. Caso esta
          ultrapasse a quantidade máxima permitida (maxQty) a permissão é
          negada*/
        boolean buyPermission = false;
        Integer qty = qtyPerAgentHashtable.get( id );
        if ( qty != null )
        {
            if ( qty.intValue() >= maxQty )
            {
                buyPermission = false;
            }
            else
            {
                buyPermission = true;
            }
        }
        else
        {
            buyPermission = true;
        }
        return buyPermission;
    }
```

Fragmento de Código 10 – Reação responsável pelo racionamento de itens. É também a responsável pelo envio das ordens de compra para os fornecedores