UMA FERRAMENTA DE APOIO À METODOLOGIA OMT PARA INTEGRAÇÃO DE MODELOS

Marco Gonzalez
Afonso I.Orth
Instituto de Informática - PUC/RS
Av.Ipiranga, 6681 - CEP.90619 - PoA/RS - BR
gonzalez@music.pucrs.br
orth@music.pucrs.br

Abstract

While the diagramatic activity occurs, the Square tool builds a matrix model where all the OMT (Object Modeling Techinique) [RUM91] elements are present. The matrix model has sectors, to localize the modeled elements, patterns, to identify them, and filters to occult rows and columns thas are not important in a specific context. The filtered matrix visualization and consistency rules orient the integrated building of the models originating in the OMT diagrams .

Resumo

Enquanto a prática diagramática se desenvolve, a ferramenta Square constrói um modelo matricial onde são representados todos os elementos da metodologia OMT (Object Modeling Techinique) [RUM91]. O modelo matricial possui setores, que facilitam a localização do que é modelado, padrões, que auxiliam a sua interpretação, e filtros, que ocultam o que não é relevante num determinado contexto. A visualização da matriz filtrada e regras de consistência orientam a construção integrada dos modelos originados nos diagramas da OMT.

Palavras-chave: Object Modeling Techinique, integração de modelos, Square, análise orientada a objetos.

1 INTRODUÇÃO

A verificação de consistência entre os modelos de uma metodologia, no desenvolvimento orientado a objetos, é ainda muito pouco considerada [DED94], entretanto, ela dá qualidade e produtividade à especificação e, por isso, deve ser tentada. Soluções para a integração de modelos tem passado por três caminhos distintos: (1) através do uso de um mecanismo integrador que unifique os elementos da descrição: repositórios [MOR90] de informações foram os primeiros mecanismos usados para integrar as informações geradas durante o desenvolvimento de sistemas; (2) através da priorização da semântica: o conceito de evento [RAM91] tem sido abordado como agente integrador e (3) através do formalismo na descrição [HAY91]: os autores que seguem neste caminho tentam dar uma base matemática aos modelos.

A metodologia OMT utiliza três modelos que dão visões diferentes do domínio do problema. Os elementos modelados neles necessitam ser integrados numa descrição coerente. A ferramenta Square, aqui apresentada, gerencia um modelo matricial que faz parte de um repositório onde são armazenadas as informações geradas pelos diagramas da OMT. Os elementos modelados nos diagramas são, também, representados na matriz e o modelador pode visualizá-los sob uma nova perspectiva em fragmentos de matriz, obtidos através de filtros, ou ter em seu auxílio regras de consistência para orientá-lo na prática de diagramação.

2. O MODELO MATRICIAL

A matriz manipulada pela ferramenta Square baseia-se no conceito de "matriz associada a um grafo" [Mac74] e tem similaridade com "N^2Chart" [Loy93]. É uma matriz associada aos diagramas da OMT, onde os elementos-vértices são colocados na diagonal principal e os elementos-arcos, fora dela, na interface entre os primeiros. Na coluna de um elemento-vértice

são representadas as suas interfaces de entrada e nas células da sua linha, as interfaces de saída. A matriz faz parte de um repositório de informações que é construído a medida que o processo de diagramação se desenvolve. Não substitui os diagramas e nem é um modelo que dispensa outros para descrever o domínio do problema.

3. REPRESENTAÇÃO DOS ELEMENTOS DA OMT NO MODELO MATRICIAL

A figura 1 mostra as notações dos principais elementos do modelo de Objetos da OMT. Nas células da diagonal principal da matriz são representadas as classes de objetos e as operações definidas para elas. Associações (sombreamento) e agregações (sombreamento e hachura) são representadas nas células entre as classes relacionadas. Uma generalização é representada na célula com IS-A localizada na coluna da classe mais genérica e na linha da mais específica.

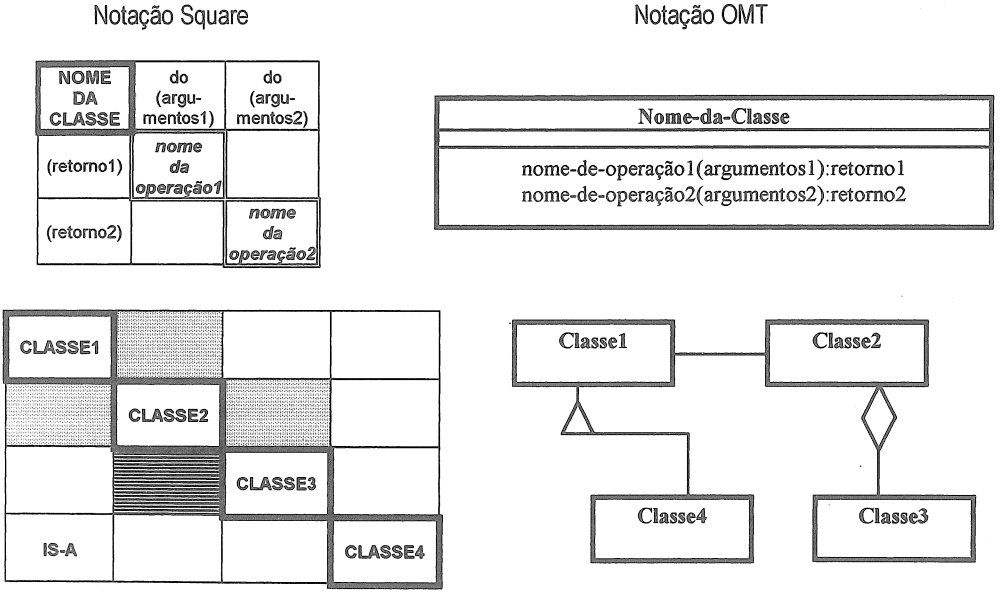

Figura 1. Notações dos elementos do modelo de Objetos

A figura 2 mostra as notações dos principais elementos do modelo Dinâmico da OMT. Um fluxo de evento é representado pelo nome do evento na célula da linha da classe de objetos de origem e da coluna da operação correspondente da classe de objetos de destino. Um estado é representado pelo nome do estado numa célula da diagonal principal da matriz, tendo na célula de sua coluna com a linha da classe correspondente a indicação "**on**".

Notação Square

CLASSE1		*nome do evento*
	CLASSE2	do (lista-de-ar-gumentos)
evento de retorno	(saída)	*operação correspon -dente*

CLASSE	on	on	do (lista-de-ar-gumentos)	
	estado1		*nome do evento*	
		estado2		
(saída)		[condição]	*operação*	[condição]
				ação

Notação OMT

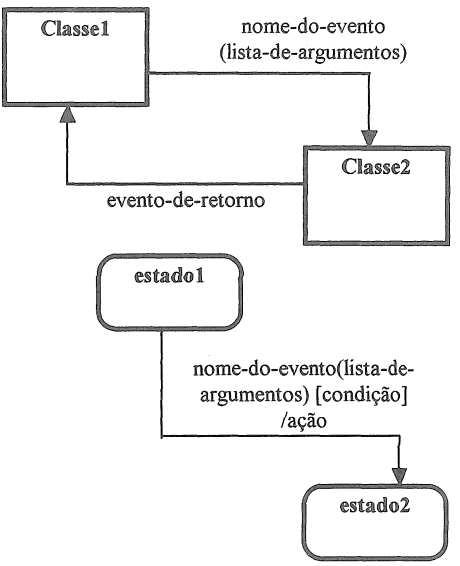

Figura 2. Notação dos elementos do modelo Dinâmico

A figura 3 mostra as notações dos principais elementos do modelo Funcional da OMT. Um processo é representado numa célula da diagonal principal da matriz. Um depósito de dados é representado como classe, assim como um ator. Um fluxo de dados é representado pela lista de dados correspondentes na célula da linha do elemento origem e da coluna do elemento destino. Um fluxo de objeto é representado pela lista de atributos do objeto. Um fluxo de controle é

representado pela condição entre colchetes. Um subprocesso é representado como um processo, tendo uma indicação "**in**" na célula de sua linha e da coluna do superprocesso. A implementação de operação por processo-folha (primitivo funcional) é representada pela indicação "**in**" na célula da linha da operação e da coluna do processo-folha. Se existir fluxo de dados ou fluxo de controle entre os processos, as células correspondentes entre as operações serão sombreadas.

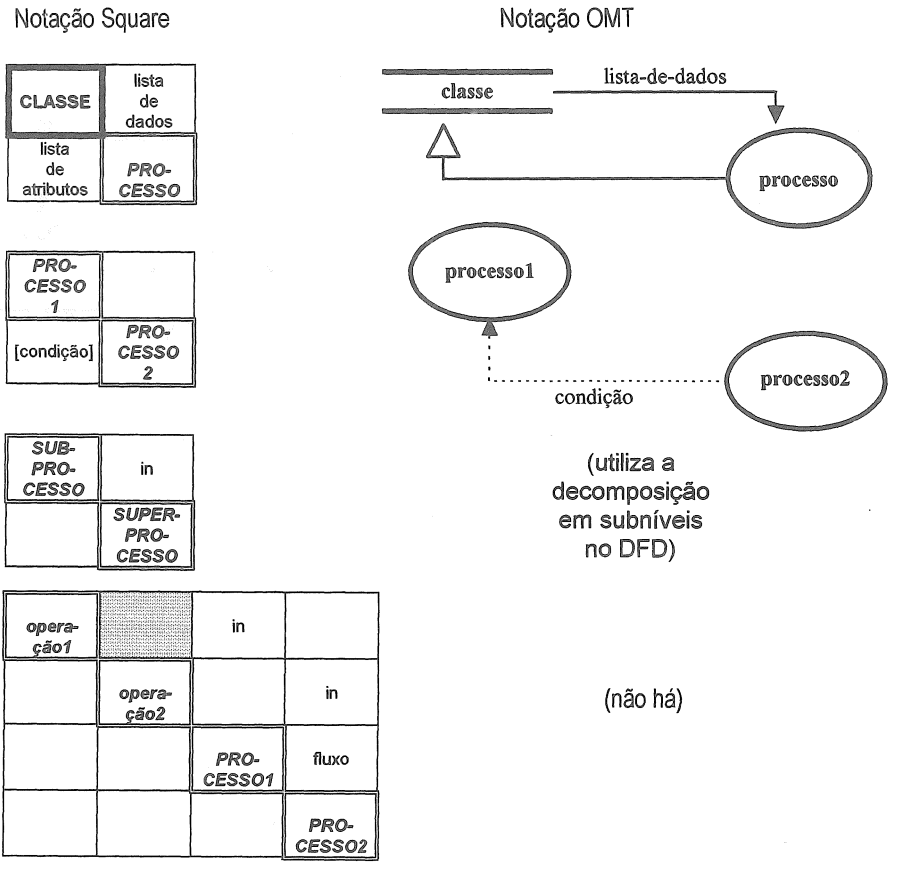

Figura 3. Notação dos elementos do modelo Funcional

4. SETORES, PADRÕES E FILTROS

Para auxiliar a visualização, a matriz possui setores (que facilitam a localização dos elementos representados), padrões (que facilitam o reconhecimento deles) e filtros (que ocultam aqueles que não são relevantes num determinado contexto).

4.1. SETORES

A matriz Square é dividida em 9 setores, conforme mostra a figura 4.

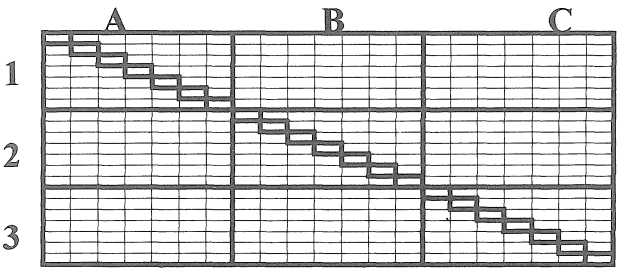

Figura 4. Setores da matriz Square

A1 é o setor dos objetos: na diagonal principal são representadas classes e estados, fora dela principalmente os relacionamentos. B2 é o setor das operações: na diagonal principal aparecem as operações e fora dela as interações. C3 é o setor dos processos: na diagonal principal estão os processos e fora dela os fluxos de dados e de controle. B1 é o setor das responsabilidades e colaborações: aparecem eventos e os argumentos de entrada das operações. A2 é o setor dos retornos: são representados os eventos de retorno, saídas das operações e condições de transições de estados. C1 é o setor dos atributos lidos: são representados os fluxos de dados com destino em processos. A3 é o setor dos atributos processados: são representados os fluxos de dados com origem em processos. C2 é o setor das implementações: são identificadas quais operações são implementadas pelos processos. O setor B3 não é tratado aqui.

4.2. PADRÕES

A notação utilizada na matriz produz alguns padrões que podem ser reconhecidos pelo modelador. Alguns deles são: padrão de evento - constituído por uma associação entre classes, pelo evento, sua participação numa transição de estados (se houver), pela interação entre operaçãos das classes envolvidas e pelo fluxo de dados entre os processos que implementam estas operações -, padrão de responsabilidades - constituído pelos eventos e pelas indicações "do" contidos na linha de uma classe - e padrão de colaborações - consituído pelos eventos contidos nas colunas das operações de uma classe.

4.3. FILTROS

O modelo matricial é visualizado através de filtros (figura 5). A matriz filtrada possui segmentos de algumas linhas e colunas da matriz total, sempre contendo a diagonal principal. Os filtros determinam quais linhas e colunas da matriz devem estar ocultas no momento da visualização. Este mecanismo é bastante utilizado em planilhas eletrônicas para eliminar informações não relevantes num determinado contexto de visualização de uma planilha Os principais tipos de filtros são: Filtro por evento, Filtro por colaborações, Filtro por responsabilidades e Filtro por processo.

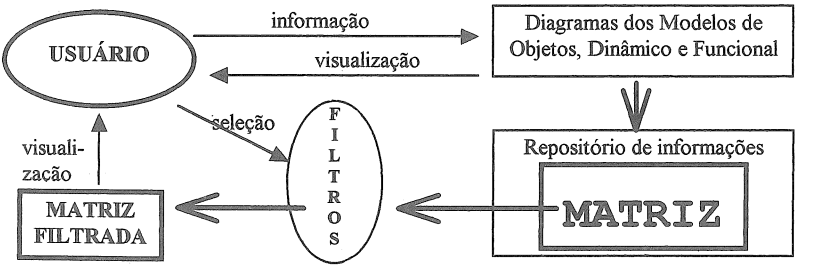

Figura 5. Construção e visualização do modelo matricial

CLASSE1				*evento*	do (args1)		
	CLASSE2	on	on	do (args2)			
		estado1		*evento*			
			estado2				
retorno	(saída2)		[condição]	*operação2*		in	
(saída1)					*operação1*		
	fluxo2					*PRO-CESSO2*	
fluxo1						fluxo3	*PRO-CESSO1*

Figura 6. Esquema de matriz filtrada por evento

CLASSE1			*eventoA*			
	CLASSE2			*eventoB*		
		CLASSE3	do (argsA)	do (argsB)		
retornoA		(saídaA)	*operação A*		in	
	retornoB	(saídaB)		*operação B*		in
		fluxoA			*PRO-CESSO1*	
		fluxoB				*PRO-CESSO2*

Figura 7. Esquema de matriz filtrada por colaborações (da CLASSE3)

A matriz filtrada por evento (figura 6) mostra elementos vinculados a um padrão de evento. Questão fundamental: Qual a relação que existe entre responsabilidades e eventos? Verifica-se a relação que eventos tem com processos, com classes de objetos e com seus estados.

A matriz filtrada por Colaborações (figura 7) mostra elementos vinculados ao padrão de colaborações de uma classe. Questão fundamental: Quais as colaborações que uma classe de objetos oferece às demais para que estas cumpram suas responsabilidades? Verificam-se as colaborações que uma classe de objetos oferece através dos serviços que presta a outras classes.

A matriz filtrada por Responsabilidades (figura 8) mostra elementos vinculados ao padrão de responsabilidades de uma classe. Questão fundamental: Quais são as responsabilidades de uma classe de objetos e o que ela necessita de outras classes para cumprir todas as suas responsabilidades? Verificam-se eventos e operações referentes a uma classe de objetos.

A matriz filtrada por Processo (figura 9) mostra elementos vinculados a um processo especificado. Questão fundamental: Quais colaborações ocorrem para que possam ser cumpridas as responsabilidades envolvidas num processo? São consideradas as operações implementadas num processo e as classes envolvidas executando ou solicitando estas operações.

CLASSE1		do (args1A)	do (args1B)	*evento*	fluxo	
	CLASSE2			do (args2)		
(saída1A)		*operação 1A*			in	
(saída1B)			*operação 1B*			in
retorno	(saída2)			*operação2*		
					PRO-CESSO1	
						PRO-CESSO2

Figura 8 Esquema de matriz filtrada por responsabilidades (da CLASSE1)

CLASSE1			do (argsA)			
	CLASSE2		*eventoA*	do (argsB)	*eventoC*	fluxoB
		CLASSE3			do (argsC)	
(saídaA)	*retornoA*		*operaçãoA*			in
	(saídaB)			*operaçãoB*		in
	retornoC	(saídaC)			*operaçãoC*	in
fluxoA						*PRO-CESSO*

Figura 9. Esquema de matriz filtrada por processo

5. INTEGRANDO MODELOS

O ideal seria que o modelador pudesse abstrair alguns elementos diagramados para verificação de coerência. O problema é que os diagramas são diversos e geralmente extensos e complexos. Na prática é difícil comparar fragmentos de diagramas. Esta tarefa exige esforço e tempo na analise do que foi modelado. A visualização da matriz filtrada e as regras de consistência (baseadas no modelo único, o matricial) podem responder algumas questões importantes para a integração. As regras são aplicadas exaustivamente sobre toda a matriz. Os filtros enfatizam fragmentos dela. Durante a diagramação, o modelador pode selecionar um filtro e visualizar, no formato matricial, os elementos dos modelos de Objetos, Dinâmico e Funcional, num local único. A verificação dos padrões detectam inconsistências que prejudicam a qualidade da descrição. As regras de consistência podem ser ativadas automaticamente ao ser detectada uma incoerência durante a atividade diagramática ou, a pedido do modelador, ser gerado um relatório sobre o estado corrente de integração dos modelos.

6. CONCLUSÃO

A representação no formato matricial de elementos modelados em diagramas é apresentada aqui como uma contribuição para a solução do problema de integração de modelos da metodologia OMT. A coerência entre os modelos pode ser verificada sob uma perspectiva diferente e numa visão única. Os elementos modelados através dos diagramas são representados cada um deles num único local na matriz, unificados num só modelo. Isto favorece a ausência de ambiguidades e a verificação de consistência. A dificuldade de trabalhar um modelo único (extenso) é eliminada pela utilização de filtros. Naturalmente, como em qualquer técnica ou ferramenta, a prática conseguida pelo uso continuado irá facilitar a localização e o reconhecimento dos elementos na matriz. Ao encontro disso, setores e padrões auxiliam o modelador na tarefa de visualização. A atividade diagramática, predominante, é auxiliada pelo modelo matricial. Incoerências são detectadas visualmente pela quebra de padrões ou pela aplicação de regras de consistência sobre toda a matriz.

REFERÊNCIAS BIBLIOGRÁFICAS

[DED94] DEDENE, B.; SNOECK, M. M.E.R.O.D.E.: A Model-driven Entity-Relationship Object-oriented Development Method. ACM SIGSOFT, Software Entineering Notes 19(3), p.51-61, julho 1994.

[HAY91] HAYES, F.; COLEMAN, D. Coherent Models for Object-Oriented Analysis. OOPSLA'91, pp.171-183. 1991.

[LOY93] LOY, P. A Comparison of Object-Oriented and Structured Development Methods. ACM SIGSOFT. Software Engineering Notes, 15(1). Janeiro 1990.

[MAC74] MACIEL, J. Elementos de Teoria Geral dos Sistemas. Editora Vozes Ltda. Petrópolis. 1974.

[MOR90] MORIARTY, T. Are You Ready for a Repository? Database Programming & Design, San Francisco. V. 3(3), p.60-71. Março 1990.

[RAM91] RAMACKERS, G.J.; VERRIJN-STUART, A.A. Integrating Information System Perspectives with Objects. Object Oriented Approach in Information Systems. F.Van Assche, B.Moulin, C.Rolland (Editores). Elsevier Science Publishers B.V. (North-Holland). 1991.

[RUM91] RUMBAUGH, J.; BLAHA, M.; PREMERLANI, W.; EDDY, F.; LORENSEN, W. Object-Oriented Modeling and Deseign. Englewood Cliffs. Prentice-Hall. 1991.

Alternativas para Implementar Clases Genéricas en C++

Acerca de los Autores

Marco V. Alvarado

Labora en el área de Investigación y Experimentación de SYSDE Soluciones Financieras S.A., donde funge como el coordinador de área, a la vez que trabaja en el desarrollo de herramientas básicas de alta tecnología para el sector financiero.

Ha ocupado puestos de Administrador de Base de Datos y Director de soporte interno (software) en Proyectos y Sistemas PROYECTICA S.A., desarrollador casual en Softec S.A., y músico de fila (contrabajo) en la Orquesta Sinfónica Nacional (Costa Rica).

Tiene experiencia en desarrollo de Sistemas y herramientas de alto y bajo nivel en Turbo Pascal, C y C++ (Borland C++, Microsoft Visual C++, GNU C++) sobre plataformas DOS, Windows (3.1) a 16 bits, Windows NT (3.51) a 32 bits (incluyendo uso de múltiples hilos de ejecución) y UNIX (en mayor grado con HP-UX, Solaris y Linux, y en menor grado con Digital UNIX y AIX), junto con SQL/PL-SQL (ORACLE6 y 7), Modula2, etc...

Obtuvo un Bachillerato en Música en la Universidad Nacional (C.R.) en 1987, un Bachillerato en Computación e Informática en la Universidad de Costa Rica en 1994, y actualmente cursa la Maestría en Computación e Informática en la Universidad de Costa Rica.

Dentro del área informática, sus actuales campos de estudio son los Sistemas Operativos multitarea actuales, herramientas basadas en Inteligencia Artificial, Telefonía y cualquier otra aplicación que permita combinar sonido con desarrollos computacionales. Dedica parte de su tiempo libre a la composición musical.

Ricardo Villalón

Ricardo Villalón es Gerente General de la empresa SofTec Ingeniería en Computación S.A., empresa dedicada a dar consultorías a empresas, diseño e instalación de redes de computadores y desarrollo de software a la medida. Además, se desempeña como profesor de la Escuela de Computación en la Universidad de Costa Rica desde 1989, en cursos de programación y lógica digital; también forma parte del Comité de Pruebas de Grado de Computación en la Universidad Autónoma de Centro América desde 1993. También ha laborado en la Universidad Nacional e impartido clases a nivel privado.

Obtuvo el título de Bachiller en Computación e Informática en la Universidad de Costa Rica en 1988 y el de Licenciado en Computación e Informática en la misma Universidad en 1993, también

a cursado materias del plan de Maestría en Computación en el Instituto Tecnológico de Costa Rica y actualmente es estudiante de la Maestría en Computación en la Universidad de Costa Rica.

Es miembro de la IEEE Computer Society y del Colegio de Profesional en Informática y Computación de Costa Rica.

Ha dirigido proyectos de desarrollo de software y diseño e instalación de redes de computadores en empresas como la Industria Nacional de Cemento, Corporación Peters, en empresas de servicios aduaneros, textiles, etc.

Tiene experiencia en lenguajes de programación como Ensamblador para procesadores Intel, Basic, Pascal, C, C++ para ambientes DOS y Windows, herramientas de desarrollo de software como FoxPro, Clipper, DBase, Btrieve, en bases de datos como Oracle 6, experiencia en sistemas operativos como DOS, Unix y OS/2 y amplia experiencia en el desarrollo de software orientado a objetos. Ha participado en proyectos para crear bibliotecas que permitan enlazar rutinas de software desarrollado en varios lenguajes de programación y en proyectos de software de control automático.

Actualmente trabaja en un proyecto para desarrollar aplicaciones distribuídas usando el lenguaje Java en la plataforma de desarrollo de Internet.

Marcelo Jenkins

Marcelo Jenkins es consultor independiente en informática y cuenta con más de 10 años de experiencia en consultoría en el área de sistemas de información e ingeniería de software.

El Dr. Jenkins es también profesor Asociado en la Escuela de Ciencias de la Computación e Informática de la Universidad de Costa Rica, en donde actualmente dirige el Programa de Posgrado en Computación e Informática. Obtuvo su bachillerato en Computación e Informática en la Universidad de Costa Rica en 1987, y su Maestría y Doctorado en Computación e Informática en la Universidad de Delaware, Estados Unidos, en 1989 y 1992 respectivamente.

Autor de diez publicaciones aceptadas por comités editoriales en diferentes conferencias, sus principales áreas de interés incluyen los ambientes orientados a objetos, sistemas abiertos, ingeniería de software, y planes de calidad para desarrollo de software.

Actualmente, desarrolla un proyecto de investigación en el área de estándares de calidad para desarrollo de software, cuyo objetivo es adaptar algunos estándares internacionales de calidad a procesos de desarrollo de software en empresas pequeñas y medianas.

El Dr. Jenkins es miembro del la ACM, la IEEE Computer Society, y la ASQC.