Universidade Federal de Juiz de Fora
Programa de Mestrado em Modelagem Computacional

Nívea Bellose Oliveira de Souza

CARACTERIZAÇÃO DE SOFTWARE CIENTÍFICO: UM ESTUDO DE CASO EM MODELAGEM COMPUTACIONAL.

Juiz de Fora
2011

CARACTERIZAÇÃO DE SOFTWARE CIENTÍFICO: UM ESTUDO DE CASO EM MODELAGEM COMPUTACIONAL

Nívea Bellose Oliveira de Souza

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Modelagem Computacional da Universidade Federal de Juiz de Fora, como parte dos requisitos necessários à obtenção do grau de Mestre em Ciências em Modelagem Computacional.

Orientadora: Fernanda Cláudia Alves Campos

Juiz de Fora
Janeiro de 2011

Souza, Nívea Bellose Oliveira de.
Caracterização de software científico: um estudo de caso em modelagem computacional / Nívea Bellose Oliveira de Souza. – 2011.
61 f. : il.

Dissertação (Mestrado em Modelagem Computacional)—Universidade Federal de Juiz de Fora, Juiz de Fora, 2011.

1. Softwares. 2. Processamento eletrônico de dados. 3. Programação (computadores). I. Título.

CDU 681.3.000S

Nívea Bellose Oliveira de Souza

CARACTERIZAÇÃO DE SOFTWARE CIENTÍFICO:
UM ESTUDO DE CASO EM MODELAGEM COMPUTACIONAL

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de Mestre.

Aprovada em 27 de Janeiro de 2011.

BANCA EXAMINADORA

_____

Profª. Dra. Fernanda Cláudia Alves Campos – Orientadora
Universidade Federal de Juiz de Fora

_____

Profª. Dra. Neide dos Santos
Universidade Estadual do Rio de Janeiro

_____

Profª. Dra. Regina Maria Maciel Braga Villela
Universidade Federal de Juiz de Fora

Dedico este trabalho aos meus pais, marido e filhos.

AGRADECIMENTOS

A Deus, em primeiro lugar, pois sem Ele jamais teria chegado até aqui.
À minha orientadora Fernanda Campos, pela excelente orientação, por toda a dedicação e incentivo ao longo deste trabalho.
À prof.ª Regina Braga que acreditou em mim desde o início.
Ao professor Maicon Corrêa pelo auxílio neste trabalho.
Aos entrevistados que participaram desta pesquisa.
A todos os professores que colaboraram com meu crescimento intelectual.
Aos membros da Banca Examinadora pelo trabalho de avaliação.
Aos meus pais, Edson e Sônia, por sempre me apoiar e incentivar a realização de todos os meus sonhos.
Ao meu marido, Alfredo, pelo amor, carinho, paciência e incentivo.
Aos meus filhos, Julia e João Pedro, que muitas vezes foram privados da minha companhia por razão de meus estudos.
A todos que, direta ou indiretamente, contribuíram para a realização deste trabalho.

Ciência da Computação tem tanto a ver com o computador como a Astronomia com o telescópio, a Biologia com o microscópio, ou a Química com os tubos de ensaio. A Ciência não estuda ferramentas, mas o que fazemos e o que descobrimos com elas.

Edsger Dijkstra

Resumo da Dissertação apresentada à Universidade Federal de Juiz de Fora como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências

Na era da Internet, os Softwares Científicos são não somente o instrumento para a geração de resultados, mas também cruciais para a maior parte das pesquisas. A maioria dos cientistas, entretanto, aprende o que sabe sobre desenvolvimento de software informalmente, e esta informalidade normalmente gera um caráter aleatório aos produtos finais de software, dificultando o seu reuso e a sua interoperabilidade. Os Softwares Científicos, desenvolvidos especificamente para atender às necessidades de cada cientista em sua respectiva área de atuação, tornaram-se muito importantes tanto em seu desenvolvimento quanto em sua utilização e, dependendo da natureza da atividade científica em questão, essenciais. Outra característica da computação científica é que os Softwares Científicos também são frequentemente explorativos, desenvolvidos com o propósito de ajudar a entender um problema. Muitas práticas da Engenharia de Software podem contribuir para a geração de produtos científicos de alta qualidade. A aplicação de modelos, técnicas e ferramentas contribuem para a construção do Software Científico mais confiável, seguro e eficiente e que apresente menos falhas durante a sua execução. Estudos recentes apontam para a aplicação de métodos ágeis, pelo fato de os mesmos serem mais receptivos a mudanças e por lidarem melhor com requisitos emergenciais. Essa dissertação tem por objetivo caracterizar o desenvolvimento do Software Científico, através de um estudo exploratório com um grupo de pesquisadores do LNCC, especialistas na área de modelagem computacional. Para a caracterização do Software Científico foi construído um instrumento com as seguintes questões de investigação: identificação de características de qualidade do Software Científico; identificação dos modelos de processos adotados no desenvolvimento do Software Científico; possibilidades de adoção de práticas da Engenharia de Software; possibilidade de desenvolvimento de Software Científico por equipe especializada, que não os próprios cientistas; reutilização de artefatos científicos; compartilhamento de Software Científico e identificação de características dos workflows científicos. Com a aplicação do instrumento, os resultados foram quantificados e analisados, identificando características do Software Científico e de workflow científico. Ao final são definidas algumas propostas para a adoção de práticas de Engenharia de Software pelos cientistas avaliados. Os resultados fornecem indícios de como melhorar o Software Científico, seu processo de desenvolvimento, seu compartilhamento e reuso.

Abstract of Dissertation presented to Federal University of Juiz de Fora as a partial fulfillment of the requirements for the degree of Master of Science

In the Internet era, scientific software are not only an instrument to generate results but also crucial to most researches. A lot of scientists, however, learn what they know about software development informally, and this informality typically generates a random character to the final software products, making it difficult to reuse and interoperate. Scientific software specifically developed to meet individual scientists needs in their respective areas of work have become very important both in its development process as in its use and, depending on the nature of scientific activity, it is essential. Another feature of scientific computing is that the scientific software are also often exploited, developed with the focus of helping to understand a problem. Many software engineering disciplines can contribute to the generation of high-quality scientific products. The application of models, techniques and tools contribute to the construction of scientific software more reliable, secure and efficient and with fewer failures during its execution. Recent studies point to the application of agile methods, because they are more receptive to change and deal better with emergency requirements. This research aims to characterize the development of scientific software, through an exploratory study with a group of researchers from LNCC, experts in computational modeling area. For the characterization of scientific software we built an instrument with the following research questions: identify scientific software quality criteria, identification of development process models for scientific software, adoption possibilities for Software Engineering practices, possibility of scientific software development by specialized team instead of scientists themselves, re-use of scientific artifacts, sharing of scientific software and identification of scientific workflows characteristics. After the application of the instrument, the results were quantified and analyzed, identifying characteristics of scientific software and scientific workflow. At the end we define some proposals for software engineering practices adoption by scientists involved in this research. The results provide evidences on how to improve scientific software, its development process, sharing and reuse.

LISTA DE FIGURAS

Figura 2.1 Um modelo de desenvolvimento de Software Científico (Segal, 2008)..... 11

Figura 2.2 Processo de Desenvolvimento em ambientes de pesquisa (Sander & Kelly, 2008 apud Berni, 2010)..... 12

Figura 2.3 Resultado da validação das características da Norma ISO 9126..... 16

Figura 3.1 Uma Taxonomia de *Workflow* Científico para Computação em Grid (Yu *et. al,* 2005)..... 32

LISTA DE QUADROS

Quadro 2.1	Características de Qualidade de Software Científico.....	14
Quadro 2.2	Característica Usabilidade.....	16
Quadro 2.3	Característica Manutenibilidade.....	17
Quadro 2.4	Característica Portabilidade.....	17
Quadro 2.5	Identificação das Características dos Softwares Científicos.....	19
Quadro 3.1	Características dos *Workflows* Científicos.....	33
Quadro 4.1	Características da Qualidade do Software Científico.....	37
Quadro 4.2	Modelos de processos adotados no desenvolvimento do Software Científico.....	37
Quadro 4.3	Reutilização de artefatos científicos e compartilhamento de Software Científico.....	38
Quadro 4.4	Práticas de Engenharia de Software.....	38
Quadro 4.5	Práticas de Desenvolvimento de Software Científico.....	39
Quadro 4.6	Modelos de Processo de Desenvolvimento de Software.....	39
Quadro 4.7	Tipos de Profissionais envolvidos no projeto.....	39
Quadro 4.8	Características dos *Workflows* Científicos.....	40
Quadro 4.9	Características da Qualidade dos Softwares Científicos.....	41
Quadro 4.10	Características importantes dos *Workflows* Científicos.....	44

SUMÁRIO

1 INTRODUÇÃO 1
1.1 JUSTIFICATIVA..... 1
1.2 OBJETIVOS 2
1.3 METODOLOGIA 3
1.4 ESTRUTURA DO TRABALHO..... 4
2 SOFTWARE CIENTÍFICO..... 5
2.1 QUALIDADE DE SOFTWARE..... 6
2.2 CARACTERÍSTICAS..... 7
2.3 PROCESSO DE DESENVOLVIMENTO..... 10
2.4 QUALIDADE..... 13
2.5 CARACTERIZAÇÃO DO SOFTWARE CIENTÍFICO..... 18
3 WORKFLOW CIENTÍFICO..... 20
3.1 CARACTERÍSTICAS..... 22
3.2 REUTILIZAÇÃO..... 29
3.3 TAXONOMIA..... 31
3.4 CARACTERIZAÇÃO DO WORKFLOW CIENTÍFICO..... 33
4 ESTUDO DE CASO: CARACTERIZAÇÃO DO SOFTWARE CIENTÍFICO EM MODELAGEM COMPUTACIONAL..... 35
4.1 RESULTADOS DO ESTUDO..... 36
4.2 ANÁLISES DOS RESULTADOS..... 40
4.3 RECOMENDAÇÕES..... 44
5 CONSIDERAÇÕES FINAIS..... 48
REFERÊNCIAS..... 51
APÊNDICE..... 54
Apêndice 1 - Pesquisa de Cientistas sobre o Controle de Qualidade aos Softwares Científicos..... 54
Apêndice 2 - Formulário de Validação da Caracterização do Software Científico. 56

1 INTRODUÇÃO

1.1 JUSTIFICATIVA

Na era da internet, os Softwares Científicos são não somente o instrumento para a geração de resultados, mas também cruciais para a maior parte das pesquisas recentes (Maxville, 2009). A maioria dos cientistas, entretanto, aprende o que sabe sobre desenvolvimento de software informalmente, e esta informalidade normalmente gera um caráter aleatório aos produtos finais de software, dificultando o seu reuso e a sua interoperabilidade.

Para a criação de software confiável e de qualidade, os engenheiros de software utilizam processos de desenvolvimento de software que podem ser definidos como um arcabouço para as tarefas que são necessárias para a construção de software de alta qualidade. Um processo de software define a abordagem que é adotada quando o software é elaborado (Pressman, 2006).

Dentro deste tema, uma questão que tem merecido a atenção dos pesquisadores é o estudo do processo de desenvolvimento de Software Científico de natureza acadêmica por diversas razões. Uma delas é a existência de diversos modelos de processos para a construção de softwares, porém, dedicados a softwares convencionais. Outros tipos de software, como os criados por pesquisadores para serem utilizados em suas pesquisas, objeto deste trabalho, podem possuir particularidades e preocupações aparentemente diferentes dos sistemas convencionais e comerciais. Isso pode tornar os processos existentes inadequados para o Software Científico. Para exemplificar melhor, pode-se citar algumas diferenças importantes no processo de desenvolvimento do software convencional e do Software Científico. Uma dessas diferenças são os papéis de clientes e de desenvolvedores envolvidos no processo de desenvolvimento de um software tradicional. Normalmente, no desenvolvimento de Software Científico não existem esses dois papéis; geralmente ele é produzido por um cientista e/ou pesquisador (ou uma pequena equipe) e utilizado por ele próprio. Outra diferença importante é que muitos requisitos de um Software Científico são descobertos ao longo do seu desenvolvimento (Moura *et al.*, 2006).

Segundo Hannay *et al*. (2009), os Softwares Científicos, desenvolvidos especificamente para atender às necessidades de cada cientista em sua respectiva área de atuação, tornaram-se muito importantes tanto em seu desenvolvimento quanto em sua utilização e, dependendo da natureza da atividade científica em questão, essenciais. De acordo com Segal (2009), desenvolver o próprio software pode, entretanto, ser uma

condição altamente desejável, uma vez que os cientistas têm entendimento profundo do domínio da aplicação.

Outra característica da computação científica é que os softwares também são frequentemente exploratórios, desenvolvidos com o propósito de ajudar a entender um problema. Hannay *et al*. (2009) menciona que, em média, um cientista passa aproximadamente 30% de seu horário de trabalho desenvolvendo Softwares Científicos e 40% utilizando estes softwares, o que reforça a sua importância como ferramentas de trabalho da produção científica na atualidade. Quando o próprio cientista é também o desenvolvedor dos softwares de que necessita ele é levado a assumir uma tarefa a mais, além da sua pesquisa ou projeto propriamente dito.

Muitas práticas da Engenharia de Software podem contribuir para a geração de produtos científicos de alta qualidade. A aplicação de modelos, técnicas e ferramentas contribuem para a construção do Software Científico mais confiável, seguro e eficiente e que apresente menos falhas durante a sua execução. A utilização de um processo formal de desenvolvimento de software é de suma importância em um desenvolvimento de sucesso, principalmente quando o foco deste desenvolvimento é um software de natureza acadêmica, que geralmente apresenta, em sua essência, uma maior complexidade. Estudos recentes apontam para a aplicação de métodos ágeis, pelo fato de os mesmos serem mais receptivos a mudanças e por lidarem melhor com requisitos emergenciais. Segundo Hook & Kelly (2009), em se tratando de Software Científico é relevante citar ainda a falta de padrões para testes e o grande número de testes necessários para se atingir a confiabilidade desejada.

Essa pesquisa tem por objetivo caracterizar o desenvolvimento do Software Científico, através de um estudo exploratório no Laboratório Nacional de Computação Científica - LNCC, com um grupo de pesquisadores, especialistas na área de Modelagem Computacional.

Para a caracterização do Software Científico foi construído um instrumento cujo objetivo foi identificar as características do Software Científico, modelos de desenvolvimento e funcionalidade para o uso de ambientes de colaboração em e-Science.

1.2 OBJETIVOS

O objetivo geral deste trabalho é fazer um estudo exploratório para caracterizar Software Científico identificando quais características determinam a qualidade desejada para esse tipo de software, analisar e apontar características gerais que venham contribuir

para a definição de padrões sobre qualidade para Softwares Científicos e *workflows* científicos. Espera-se que o trabalho melhore o processo de Engenharia de Software para o desenvolvimento de Software Científico do ponto de vista do desenvolvedor e do pesquisador. Para conseguir alcançar os objetivos descritos os seguintes objetivos específicos foram definidos:

- ✓ Identificação de características de qualidade do Software Científico;
- ✓ Identificação dos modelos de processos adotados no desenvolvimento do Software Científico;
- ✓ Identificação da possibilidade de adoção de práticas da Engenharia de Software no desenvolvimento de Software Científico;
- ✓ Identificação da possibilidade de desenvolvimento de Software Científico por equipe especializada, que não os próprios cientistas;
- ✓ Identificação de características dos *workflows* científicos;
- ✓ Identificação da possibilidade de reutilização de artefatos científicos e compartilhamento de Software Científico.

1.3 METODOLOGIA

A metodologia de pesquisa inclui a revisão bibliográfica e a caracterização de um Software Científico, através de um estudo exploratório na área de Modelagem Computacional.

Na revisão bibliográfica foram apresentados os conceitos, metodologias, métodos e ferramentas que permitiram caracterizar o Software Científico. Foram revistas também as características, possibilidade de reutilização, taxonomia e caracterização de *Workflow* Científico, que viabiliza a utilização de um modelo definido como fluxo de tarefas coordenadas e encadeadas.

Após essa caracterização foi feito um estudo de caso através da aplicação de um instrumento do tipo questionário, cujo objetivo é conferir hipóteses identificadas na literatura em relação às características de um Software Científico. Um estudo de caso, segundo Gomez *et al*. (1999, pp95-96 apud Palazzi, 2010), pode ser entendido como uma “estratégia de desenho de investigação”. Por centrar em uma única situação, programa ou fenômeno concreto, os autores consideram ser um método muito útil para a análise de problemas práticos, situações ou acontecimentos que surgem no cotidiano. A utilização de desenho de

um único caso é justificada pelas seguintes razões: Pelo seu caráter crítico, permite confirmar, alterar, modificar ou ampliar o conhecimento sobre o objeto de estudo; O caráter único e peculiar de cada sujeito que intervém em um contexto educativo justifica, por si mesmo, este tipo de desenho; Pelo caráter revelador de uma situação concreta.

Após a aplicação do questionário, foi feito análise dos dados e uma avaliação qualitativa, dos resultados obtidos através do questionário aplicado aos cientistas, os entrevistados formam um grupo heterogêneo, cuja formação acadêmica mostrou-se diversificada, porém todos estão envolvidos com pesquisa em modelagem de Escoamentos em Meios Porosos.

1.4 ESTRUTURA DO TRABALHO

O presente trabalho está organizado em cinco capítulos. No Capítulo 2, é feita uma revisão sobre Software Científico. Para tanto, são apresentados conceitos, características e desenvolvimento do Software Científico e aponta também a importância da qualidade deste software. O Software Científico tratado nesta pesquisa e o seu processo de desenvolvimento são caracterizados neste capítulo.

O Capítulo 3 refere-se à revisão bibliográfica acerca de conceitos, atributos, reutilização, taxonomia e caracterização de sistemas de *workflows* científicos.

No capítulo 4 são apresentados os resultados das respostas do questionário e uma avaliação sobre a caracterização do Software Científico. O estudo de caso foi elaborado para investigar as características do desenvolvimento do Software Científico, através de um estudo exploratório com um grupo de pesquisadores do LNCC, especialistas na área de Modelagem Computacional. Neste capítulo, está descrito detalhadamente como a pesquisa foi desenvolvida e os passos seguidos para atingir o seu objetivo final.

Finalmente, no Capítulo 5 são feitas as considerações finais e sugestões para trabalho futuros.

2 SOFTWARE CIENTÍFICO

Nos dias atuais não existem mais dúvidas sobre a importância da Informática em sentido amplo. Expressões como hardware e software tornaram-se comuns; são um sinal de como a era dos computadores modificou o modo de pensar, agir e trabalhar de todas as pessoas no mundo civilizado. O campo científico não é exceção e os cientistas do século XXI não utilizam mais somente os métodos do passado. Os Softwares Científicos, desenvolvidos especificamente para atender às necessidades de cada cientista em sua respectiva área de atuação, tornaram-se com isso muito importante tanto em seu desenvolvimento quanto em sua utilização (Hannay et al., 2009) e, dependendo da natureza da atividade científica em questão, essenciais.

Como importantes instrumentos que são, os Softwares Científicos precisam se enquadrar em parâmetros de qualidade e funcionamento para que contribuam positivamente com seus usuários, ou seja, os cientistas.

Embora o critério “qualidade” seja um fator variável, o mesmo pode ser definido, como a conformidade com requisitos funcionais e de desempenho explicitamente declarados, normas de desenvolvimento explicitamente documentadas e características implícitas, que são esperadas em todo software desenvolvido profissionalmente (Pressman, 2006).

Essa definição serve para reforçar três pontos importantes sobre a qualidade dos softwares de modo geral (Pressman, 2006):

1. Os requisitos de software são a base pela qual a qualidade é medida. A falta de conformidade com os requisitos é falta de qualidade.
2. As normas especificadas definem um conjunto de critérios de desenvolvimento que guia o modo pelo qual o software é submetido à engenharia. Se os critérios não são seguidos, irá ocorrer, quase certamente, falta de qualidade.
3. Um conjunto de requisitos implícitos frequentemente não é mencionado (por exemplo, o desejo de facilidade de uso e boa manutenibilidade). Se o software satisfaz seus requisitos explícitos, mas deixa de satisfazer os requisitos implícitos, a qualidade do software é suspeita.

A qualidade deve ser uma característica fundamental de qualquer produto existente. Porém, desenvolver software de qualidade assegurada, dentro do prazo estabelecido e sem necessitar de recursos adicionais além dos que foram alocados tem sido um grande desafio para a Engenharia de Software (Pressman, 2006).

A preocupação com a qualidade de um produto de software é maior a cada dia. Essa preocupação é causada principalmente pela dificuldade em se determinar meios

através dos quais se possa planejar, avaliar e, enfim, alcançar a qualidade (Souza, 2009).

No desenvolvimento de software, o controle da qualidade envolve monitorar resultados do projeto e verificar se os mesmos estão de acordo com os padrões de qualidade previamente estabelecidos. Como consequência, é possível obter a melhoria da qualidade por meio da identificação das causas de resultados insatisfatórios e da tomada de ações para aumentar a eficiência do projeto, provendo dessa forma benefícios a todas as partes envolvidas (Hannay *et al.*, 2009).

Contudo, não é uma tarefa simples. Para assegurar a qualidade de um software é necessário primeiro identificar quais características determinam a qualidade desejada para um dado produto de software – no caso, um Software Científico. Em outras palavras, é preciso primeiramente fazer um planejamento da qualidade para que depois possa ser efetuada a avaliação. A qualidade de um software deve ser avaliada ao longo de todo o seu processo de desenvolvimento. É necessário avaliar a qualidade não só do produto final, mas dos inúmeros produtos intermediários gerados ao longo de seu desenvolvimento (Souza, 2009).

Dessa forma, mostra-se importante analisar e apontar características gerais que venham a estabelecer e esclarecer sobre uma noção abrangente de qualidade para os Softwares Científicos, a fim de se estabelecer padrões para que seja possível avaliá-los (Souza, 2009).

2.1 QUALIDADE DE SOFTWARE

Na medida em que cresce a demanda por sistemas complexos, com grande responsabilidade no contexto das organizações, a qualidade desponta como um fator essencial no desenvolvimento de software. Sendo assim, cada vez mais, há uma disposição para se investir em qualidade. Contudo, uma das primeiras dificuldades encontradas na definição e implantação de um programa de qualidade está em compreender o que, de fato, significa qualidade de software (Belchior, 1997).

Não se pode pensar em qualidade como sinônimo de perfeição. Trata-se de algo factível, relativo, substancialmente dinâmico e evolutivo, de acordo com os objetivos a serem atingidos. Considerá-la como algo absoluto e definitivo seria como igualar ao inatingível e, com base neste pensamento, dificilmente conseguiríamos produzi-la (Belchior, 1997). Para se ter uma noção mais abrangente sobre qualidade, descreve-se, a seguir, a visão conceitual de vários pesquisadores neste assunto.

Qualidade de software pode ser vista como um conjunto de características que

devem ser alcançadas em um determinado grau para que o produto atenda às necessidades de seus usuários. É por meio desse conjunto de características que a qualidade de um produto de software pode ser descrita e avaliada (Rocha *et al.*, 2001).

Cada uma dessas características pode ser detalhada em vários níveis de subcaracterísticas, chegando-se a um amplo conjunto de atributos que descrevem a qualidade de um produto de software. Torna-se, então, necessário escolher um modelo que organize os atributos e permita avaliar a qualidade do software. Esses modelos nos permitem, ainda, entender como as diferentes facetas contribuem para a qualidade do produto como um todo (Rocha *et al.*, 2001).

Em IEEE (1990), a qualidade é definida como o grau pelo qual um sistema, um componente ou um processo satisfazem seus requisitos especificados, e as necessidades ou expectativas de clientes ou usuários.

A ISO 8402 (ISO, 1994) enuncia qualidade como a totalidade de características de uma entidade, que lhe confere a capacidade de satisfazer suas necessidades explícitas e implícitas.

A ISO/IEC 9126-1 (ISO, 2003) fornece o significado de características de qualidade de software, como sendo uma referência básica à qualidade de um produto de software, utilizada em uma avaliação.

A qualidade de Software Científico é uma área que pode ser melhorada. Para Softwares Científicos, ensaios de validação são usados quase que exclusivamente. Nem inspeções, que tem se mostrado a técnica de avaliação mais eficiente, qualidade ou métodos formais, que são matematicamente rigorosos são amplamente utilizados no desenvolvimento de Software Científico, ou mesmo no desenvolvimento de software de qualquer tipo. Trabalhos futuros para identificar os métodos de avaliação de qualidade adequados para Softwares Científicos já foram identificados por outros pesquisadores (Nguyen-Hoan, 2009).

Qualidade, portanto, não significa somente excelência ou um outro atributo de certo produto final. A qualidade dever ser perseguida ao longo do processo de desenvolvimento, pois, certamente, é isto o que os usuários, no nosso caso os cientistas, esperam de um produto (Souza, 2009).

2.2 CARACTERÍSTICAS

O Software Científico é uma área que ganhou recentemente a atenção da comunidade de Engenharias e Ciências. Existem muitas oportunidades de melhorar a

maneira pelo qual o Software Científico é desenvolvido tanto pela aplicação de metodologias de engenharia, como pelo desenvolvimento de novas metodologias específicas ao domínio de desenvolvimento de Software Científico (Nguyen-Hoan, 2009).

Muitas vezes o próprio cientista é o desenvolvedor do software de que necessita. De acordo com Segal (2009), esta pode ser uma condição altamente desejável, uma vez que os cientistas têm entendimento profundo do que é necessário que o software execute e podem concluir o desenvolvimento de acordo com sua disponibilidade de prazo.

De acordo com Hannay *et al.* (2009), a maioria dos cientistas aprende o que sabe sobre desenvolvimento de softwares por conta própria, ou informalmente, com seus colegas, ao invés de aprenderem por meio de treinamento formal. Esta informalidade normalmente gera, entretanto, um caráter aleatório aos produtos finais de softwares, o que dificulta a análise e a definição de parâmetros universais de qualidade para tais softwares.

Tipicamente, os cientistas têm pouco treinamento formal em Engenharia de Software, mas podem, com freqüência, produzir softwares de alta qualidade utilizando princípios do senso comum e autodisciplina. Como resultado, muitos esforços da Engenharia de Software podem gerar produtos científicos de alta qualidade (Heroux, 2009).

É válido e importante ressaltar que, em se tratando de Softwares Científicos, os softwares em questão são veículos para se produzir resultados científicos nas diferentes áreas de aplicação. Em outras palavras, os Softwares Científicos constituem um meio e não um fim, diferente dos softwares comerciais. A adoção de certas medidas utilizadas na indústria pode dificultar de alguma forma a produção de Softwares Científicos de qualidade devido a esta diferença fundamental. Existe ainda a impressão generalizada entre os cientistas de que a aplicação de tantas formalidades oriundas das técnicas comerciais de desenvolvimento de softwares pode fazer mais mal do que bem (Heroux, 2009).

Hannay *et al.* (2009), também menciona que, em média, um cientista passa aproximadamente 30% de seu horário de trabalho desenvolvendo Softwares Científicos e 40% utilizando estes softwares, o que reforça a importância dos mesmos como ferramentas de trabalho da produção científica na atualidade. Quando o próprio cientista é também o desenvolvedor dos softwares de que necessita ele é levado a assumir uma tarefa a mais, além da sua pesquisa ou projeto propriamente dito. Isto leva frequentemente os cientistas a ficarem divididos em tempo e foco, o que pode levar também à necessidade de métodos ágeis em detrimento de métodos possivelmente mais adequados.

De acordo com Crabtree *et al.* (2009), um número de publicações e estudos de caso recentes apontam o fato de aplicações de métodos ágeis terem sido bem sucedidos em certos projetos de desenvolvimento de software. O principal argumento por trás da sustentabilidade dos métodos mais ágeis está no fato de os mesmos serem mais receptivos a mudanças, quando e se elas forem necessárias, e também no fato de que esses métodos

podem lidar melhor com requisitos emergenciais no desenvolvimento dos softwares.

Entretanto, estas descobertas podem não se aplicar a todas as situações pertinentes ao desenvolvimento de Softwares Científicos. Por exemplo, a necessidade de se ganhar tempo pode fazer com que detalhes importantes passem despercebido, o que favorece a ocorrência de *bugs* e/ou erros que deverão ser corrigidos posteriormente – o que é muito mais difícil e penoso (Crabtree *et al.*, 2009).

Dessa forma, neste contexto a qualidade do produto final pode diminuir se a atenção do cientista se dividir entre o desenvolvimento e a utilização do mesmo em pesquisa. O ideal seria, portanto, a interação dos cientistas com uma equipe de desenvolvimento que tenha sido contratada ou requisitada de alguma forma para este fim em específico. Também é importante que esta equipe seja não só qualificada academicamente falando, mas que possua entendimento da realidade e dos fatores envolvidos no desenvolvimento de softwares do ramo científico (Souza, 2009).

Assegurar a qualidade de um Software Científico é, portanto, uma tarefa que engloba todo o processo de desenvolvimento. Este processo muitas vezes precisa levar em consideração o fator tempo, mas é igualmente necessário que se utilize do bom senso para lidar com este fator. Não é desejável nem proveitoso utilizar um método de desenvolvimento veloz, mas que não gere um produto final de qualidade e que seja confiável. Um bom Software Científico deve, em última instância, ser um meio facilitador ao invés de um fator de complicação (Souza, 2009).

Dois fatores contribuem para a dificuldade de se testar um Software Científico. Um deles é a falta de padrões definidos para tais testes – meios de se comparar o desempenho do software com o que se espera dele, ou seja, resultados certos. O segundo fator é o grande número de testes necessários quando se segue qualquer técnica de teste padrão descrita na literatura de Engenharia de Software. Por causa dessa falta de padrões, os cientistas utilizam um julgamento baseado na experiência ao invés da precisão para se analisar a confiabilidade de seus softwares (Hook & Kelly, 2009).

Dessa forma, pode-se definir a confiabilidade de um software também como sendo o fato de ser possível recorrer a um determinado software sem dúvidas, ou seja, com possibilidades muito altas de que o mesmo funcione corretamente. O Software Científico mais confiável será, portanto, aquele com o qual um cientista se sentir mais seguro como usuário e mais eficiente como ferramenta, e que menos apresente falhas durante a execução das tarefas necessárias à pesquisa ou projeto (Hook & Kelly, 2009).

Falhas de software são relativamente comuns. Para sistemas críticos, cujas falhas podem significar perdas econômicas significativas, danos físicos ou ameaças a vida humana, medidas específicas devem ser tomadas para garantir confiabilidade, disponibilidade e segurança. Para tanto utilizam-se técnicas de Engenharia de Software,

principalmente relacionadas a testes de validação, para assegurar a qualidade dos sistemas (Sommerville, 2007 apud Berni, 2010).

2.3 PROCESSO DE DESENVOLVIMENTO

Como o desenvolvimento de Software Científico pode ser melhorado? Explorar esta pergunta requer estudo fundamentado na prática, nas características específicas do desenvolvimento de Software Científico e em relevantes técnicas de Engenharia de Software, métodos e ferramentas (Segal, 2004).

Poderíamos também fazer outra pergunta: como engenheiros de software podem dar melhor apoio aos cientistas? Cientistas são pessoas que trabalham em profissões altamente técnicas, têm conhecimento rico; normalmente são matemáticos e engenheiros, que desenvolvem seus próprios softwares, a fim de defender seus próprios objetivos profissionais. Ao contrário dos engenheiros de software, os cientistas usam linguagens formais e, portanto, tendem a ter alguns problemas com codificação. Na maioria das vezes, grande parte dos cientistas têm pouca educação formal ou treinamento em desenvolvimento de software (Segal, 2004).

Existem duas categorias de pessoas que desenvolvem Software Científico. Os profissionais de software e os usuários finais e desenvolvedores, os quais não têm nenhum treinamento formal e habilidade com a Engenharia de Software. Este usuário final tende a existir em uma comunidade tipicamente científica, tendo aprendido as suas habilidades de programação através de protótipo, bem no início de sua carreira de pesquisa. Uma questão importante para observar é que o foco principal desses desenvolvedores deveria ser a execução da ciência e não o desenvolvimento do código (Nguyen-Hoan, 2009).

Desenvolvimento de Software Científico é fundamentalmente diferente de desenvolvimento de software comercial. Um cientista (especialista de domínio) deve estar fortemente envolvido em desenvolver Software Científico - o desenvolvedor médio simplesmente não entende o domínio do aplicativo. Por este motivo, o cientista é frequentemente o desenvolvedor. Outra diferença existente entre eles tem a ver com as exigências de cada um (Segal, 2008).

Os estudos de campo de cientistas desenvolvendo seus próprios softwares revelaram o modelo de desenvolvimento de software mostrado na Figura 2.1 (Segal, 2008). Nessa proposta os requisitos são principalmente emergentes, o surgimento de exigências está interligado com a avaliação, e o teste é superficial.

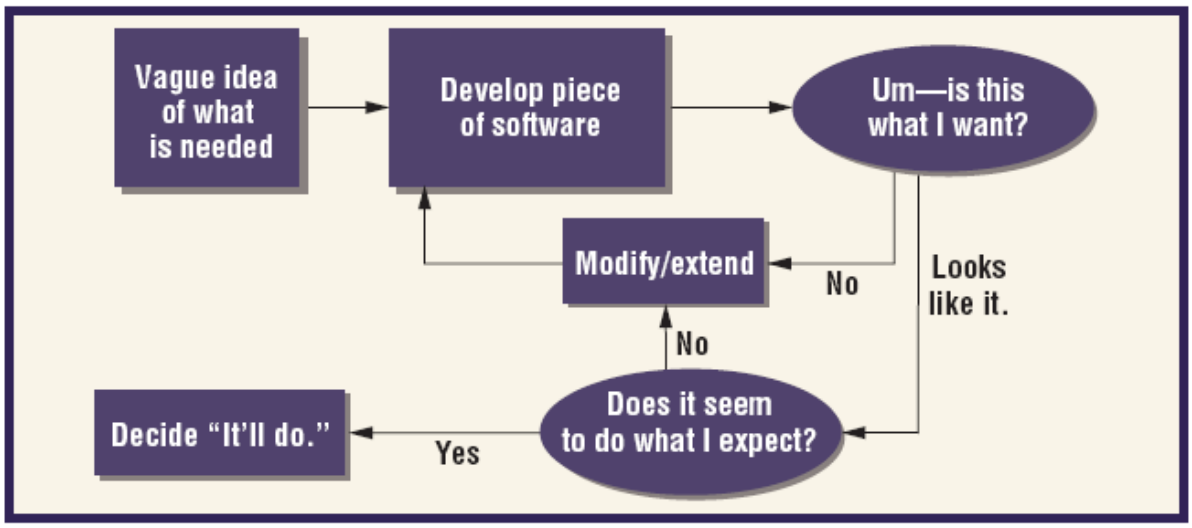

Figura 2.1. Um modelo de desenvolvimento de Software Científico (Segal, 2008).

Outros estudos de campo têm demonstrado que os esforços para aplicar técnicas de Engenharia de Software com cientistas apresentam problemas. Existe um confronto entre os engenheiros de software, que esperam uma especificação inicial de requisitos, e os cientistas, que esperam que os requisitos surjam (Segal, 2008).

Podemos observar algumas diferenças fundamentais entre o desenvolvimento de software comercial e o desenvolvimento de Software Científico. Estas diferenças incluem a maior complexidade e incerteza do domínio científico, a impossibilidade de requisitos e especificação neste sentido e as dificuldades de testar Software Científico. Tal disparidade conta para o fato que nem toda a prática usada no desenvolvimento de software comercial também é útil para o desenvolvimento de Software Científico (Segal, 2009).

Segundo Moura *et al.* (2006), o software é uma tecnologia indispensável na vida cotidiana das pessoas, ele está presente em sistemas de toda espécie: médico, transporte, entretenimento, telecomunicações e muitas outras. Essa dependência está também na sociedade científica, que cada vez mais necessita desenvolver algum tipo de software para auxiliar, validar ou comprovar suas pesquisas científicas. O processo de desenvolvimento de um software, como já se sabe, é um fator decisivo para a construção de um software confiável e de qualidade.

A construção de um Software Científico através de um processo de desenvolvimento específico para este tipo de software trará as vantagens citadas anteriormente, além de: atender a aspectos característicos deste tipo de software, definir atividades especiais para Software Científico e facilitar o entendimento do projeto desenvolvido por outros pesquisadores que por algum motivo utilizarão o Software Científico construído (Moura *et al.*, 2006).

Se a comunidade científica conseguir padronizar o processo de construção de Software Científico, mais fácil e rapidamente os projetos de pesquisas serão divulgados e expandidos por outros pesquisadores (Moura *et al.*, 2006).

Nguyen-Hoan *et al*. (2010) fizeram um estudo com 60 desenvolvedores de Software Científico com o objetivo de identificar onde podem ocorrer melhorias nas práticas de desenvolvimento de Software Científico. O estudo se concentrou em sete áreas; localização, linguagem de programação, ferramentas, equipe de desenvolvimento, documentação, teste, verificação e requisitos não funcionais. Segundo os autores o estudo confirmou hipóteses anteriores, mas também mostrou diferenças significativas. Em termos dos requisitos não funcionais a confiabilidade e a funcionalidade foram consideradas as características mais importantes. Apesar das ferramentas de controle de versões e ambientes de desenvolvimento ser usados por 50% dos entrevistados, ainda há necessidade de melhorias nessas ferramentas. A rastreabilidade não é um fator tão importante para desenvolvedores como é para usuários de Software Científico. A documentação parece estar sendo mais usada, porém a questão tempo e benefício ainda é muito questionado. Alguns testes e atividades de verificação poderiam ser mais usados, como *peer review* e testes de integração.

A Engenharia de Software para o desenvolvimento de sistemas decorrentes de pesquisas científicas, deve ser focada na rápida evolução do meio e adaptável às diferenças entre os envolvidos no processo e suas atribuições dentro do projeto (Letondal & Zdun, 2003 apud Berni, 2010). O processo de desenvolvimento de software em ambientes de pesquisa ocorre, de forma geral, como o modelo demonstrado na Figura 2.2 (Sander & Kelly, 2008 apud Berni, 2010).

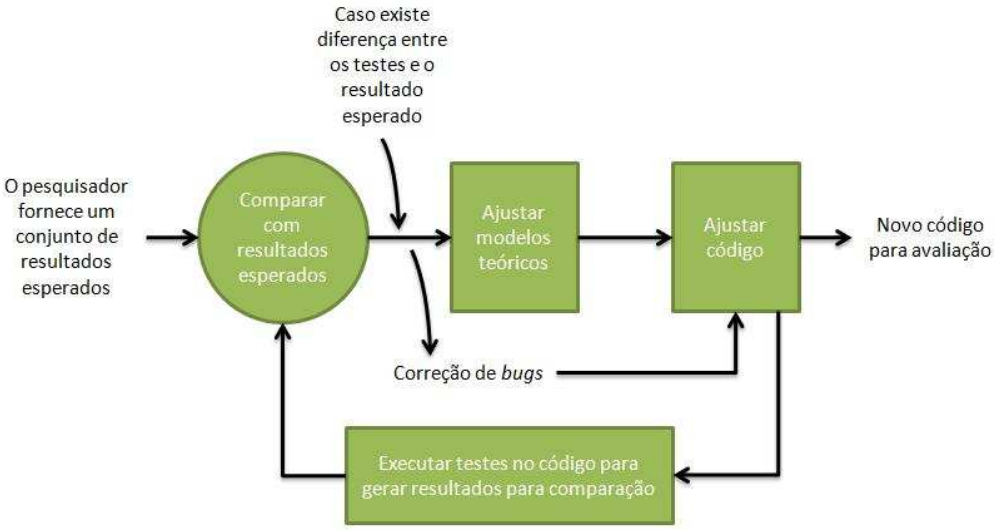

Figura 2.2 Processo de Desenvolvimento em ambientes de pesquisa (Sander & Kelly, 2008 apud Berni, 2010).

2.4 QUALIDADE DE SOFTWARE CIENTÍFICO

Provavelmente não existe hoje em dia nenhum cientista que não tenha, em alguma ocasião, utilizado um sistema de software para analisar, visualizar ou simular processos ou dados. Muitos cientistas usam tais softwares diariamente, enquanto outros os desenvolvem para uso próprio ou para uma comunidade mais ampla (Hannay *et al.*, 2009). Além disso, desde a criação dos primeiros computadores os softwares são instrumentos importantes para os cientistas e pesquisadores de diversas áreas.

Conforme indicado por muitos pesquisadores, há uma grande diferença que separa a comunidade de usuários de computação comum dos usuários de computação científica. Como resultado, tem ocorrido pouca troca de idéias significativas para os softwares de aplicação científica (Hannay *et al.*, 2009).

Uma razão para isto é que na comunidade de computação científica um desenvolvedor deve possuir conhecimento e familiaridade com o domínio de aplicação (ou seja, a ciência), quando no desenvolvimento de softwares “normais” (digamos, um sistema de planejamento de recursos para uma empresa), os desenvolvedores têm muito menos tendência a ser “*experts*” em seus domínios. Além disso, um desenvolvedor de Software Científico costuma ser também um usuário final, enquanto um desenvolvedor da Engenharia de Software “tradicional” normalmente não é. Os Softwares Científicos também são frequentemente exploratívos: o propósito do software normalmente é ajudar a entender um outro problema, o que significa que a classificação imediata desse tipo de software é difícil ou impossível. Isto pode inibir a iniciação de colaborações que rendam frutos entre engenheiros de software e cientistas (Hannay *et al.*, 2009).

É possível notar, dessa forma, que a interação entre as duas partes (desenvolvedor e cientista) pode exercer influência direta sobre a qualidade de um software que esteja sendo desenvolvido para este fim, ou seja, para uso científico. Também é importante que o desenvolvedor compreenda e seja habilidoso o bastante para atribuir funções ao software que atendam às necessidades do solicitante e do propósito para o qual este software está sendo desenvolvido (Hannay *et al.*, 2009).

A importância da qualidade dos Softwares Científicos também se deve ao fato de que, hoje em dia, quase toda pesquisa poder ser feita eletronicamente, via Internet e outros tipos de rede (Maxville, 2009). Dessa forma, a verdade é que os softwares de pesquisa, ou científicos, são não somente o instrumento para a geração de resultados como também são cruciais para a maior parte das pesquisas recentes.

Segundo Souza (2009), a preocupação com a qualidade está subentendida com o desenvolvimento de técnicas de produção, a necessidade de avaliação e o julgamento da

qualidade do produto; mas isto só é valido quando os métodos para geração do produto já estão consolidados.

O controle de qualidade surge então como uma necessidade: inicialmente são feitas verificações, adotam-se técnicas e critérios bem definidos, em alguns casos chega-se a verificação de 100% dos produtos para eliminação daqueles defeitos produzidos; finalmente certifica-se a qualidade do produto. Como alternativa busca-se melhorar o processo de produção para se adquirir maior confiança na qualidade do produto final.

A Engenharia de Software tem como objetivo a melhoria da qualidade do seu produto com propostas e modelos de desenvolvimento, métodos e técnicas para aplicação nas diversas fases de desenvolvimento. É importante a avaliação da qualidade de software nas duas visões, processo e produto, é aqui que se direciona o esforço.

Em 2009, Souza (2009), realizou um estudo exploratório para identificar características de qualidade para o Software Científico. Foram identificados na ISO 9126, atributos que foram submetidos à avaliação de onze cientistas, que as classificaram em imprescindíveis, desejáveis e sem importância de acordo com a sua visão de importância dessas características no Software Científico. O quadro 2.1 foi organizado em características e subcaracterísticas de acordo com a norma ISO 9126.

Quadro 2.1 Características de Qualidade de Software Científico.

Qualidade aos Softwares Científicos

Características	Subcaracterísticas	Atributo
Funcionalidade	Adequação	As funções devem ser adequadas às tarefas especificadas e aos objetivos dos usuários.
	Acurácia	Os resultados devem sempre ser corretos.
	Interoperabilidade	O software deve interagir com outros sistemas.
	Conformidade	O software deve estar de acordo com as normas, padrões, leis, etc.
	Segurança de acesso	O software deve possuir controle de acesso e segurança dos dados.
Confiabilidade (é imune à falhas?)	Maturidade	O software não deve apresentar falhas..
	Tolerância a falhas	O software deve ser manter o nível de desempenho mesmo em caso de falhas.
	Recuperabilidade	O software é capaz de recuperar dados em caso de falha.

Usabilidade (é fácil de usar?)	Intelegibilidade	O software deve permitir o usuário entender se ele é adequado e como ele pode ser usado para tarefas e condições de uso específicas.
	Apreensibilidade	O software deve ser fácil de aprender a usar.
	Operacionalidade	O software deve ser fácil de operar e controlar.
	Atratividade	A interface do software deve ser atrativa.
Eficiência (é rápido e enxuto?)	Tempo	O software deve ter tempo de resposta e velocidade de execução adequada.
	Recursos	O software deve necessitar de recursos adequados ao seu funcionamento.
Manutenibilidade (é fácil de modificar?)	Analisabilidade	Deve ser fácil encontrar e diagnosticar uma falha.
	Modificabilidade	Deve ser fácil modificar e adaptar o software.
	Estabilidade	As alterações no software não devem trazer grandes riscos.
	Testabilidade	Deve ser fácil testar o software quando se faz alterações.
Portabilidade (é fácil de usar em outro ambiente?)	Adaptabilidade	Deve ser fácil adaptar o software a outros ambientes.
	Capacidade para ser instalado	Deve ser fácil instalar o software em outros ambientes.
	Conformidade	O software deve estar de acordo com padrões de portabilidade.
	Capacidade para substituir	Deve ser fácil substituir o software por outro com os mesmos objetivos.

Para este estudo foi feita uma validação dessas características através do formulário constante no Apêndice I, aplicado a onze pesquisadores de diversas áreas como Física (3), Engenharia (3), Matemática (3), e Química (2), da Universidade Federal de Juiz de Fora, que opinaram sobre importância dos atributos.

Sendo assim, é possível observar por meio da Figura 2.3 que a maioria dos entrevistados considerou a seguinte ordem de prioridade das características: confiabilidade, funcionalidade, eficiência, manutenibilidade, portabilidade e usabilidade.

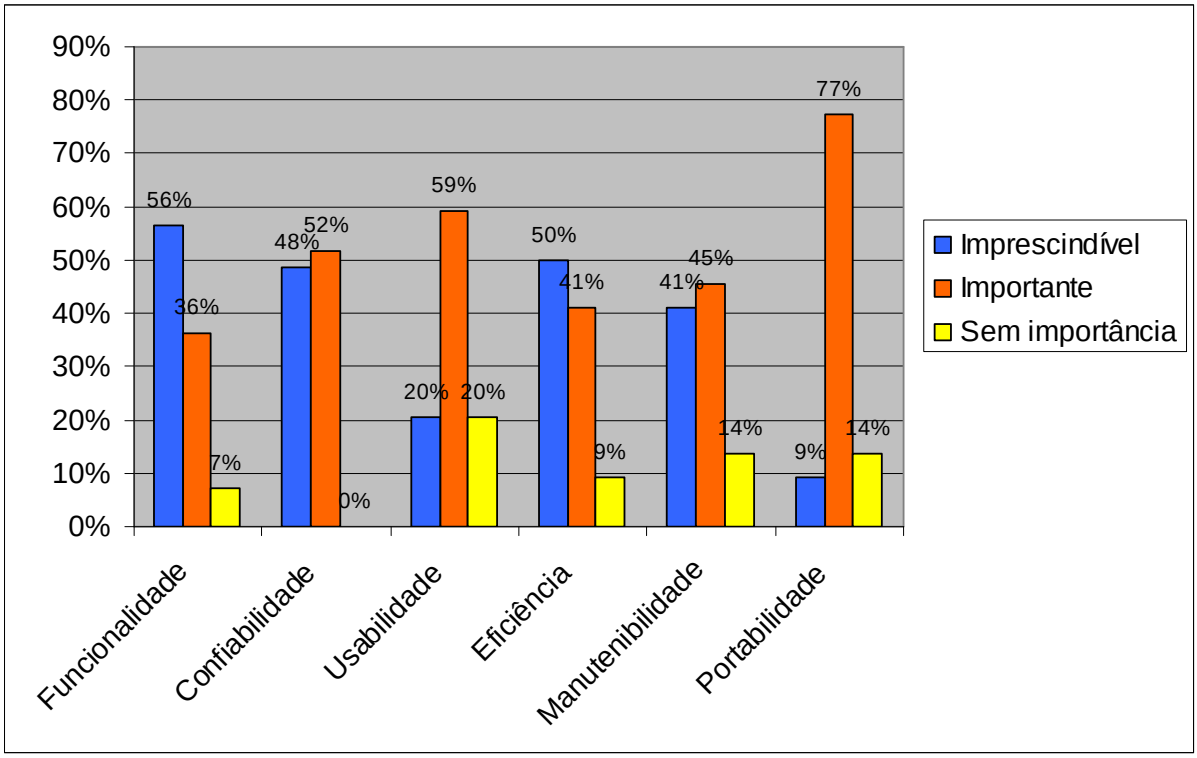

Figura 2.3. Resultado da validação das características da Norma ISO 9126.

Dos entrevistados 91% são do sexo masculino e 9% são do sexo feminino e 100% dos participantes são pesquisadores da Universidade Federal de Juiz de Fora. Em termos de *Funcionalidade* o gráfico 2.1 ilustra os resultados. O quadro 2.2 apresenta a característica Usabilidade como resultado considerado importante para estes pesquisadores. Todos eles consideraram imprescindível ou importante a Confiabilidade do software conforme gráfico 2.2 demonstrado abaixo. A maioria dos participantes consideraram como importante ou imprescindível a característica Eficiência conforme gráfico 2.3. O quadro 2.3 e 2.4 demonstra os resultados das características Manutenibilidade e Portabilidade respectivamente.

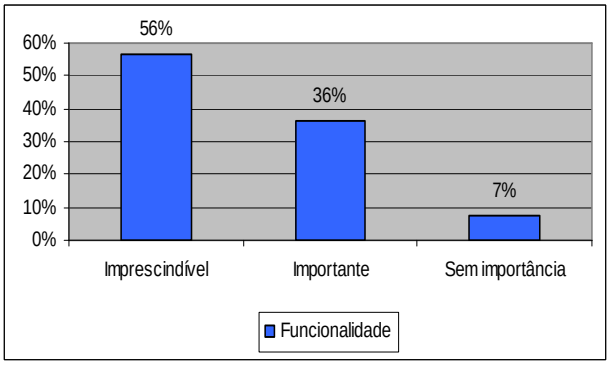

Gráfico 2.1 Característica Funcionalidade.

Quadro 2.2 Característica Usabilidade.

Atributos	Porcentagem
Imprescindível	20%
Importante	60%
Sem importância	20%

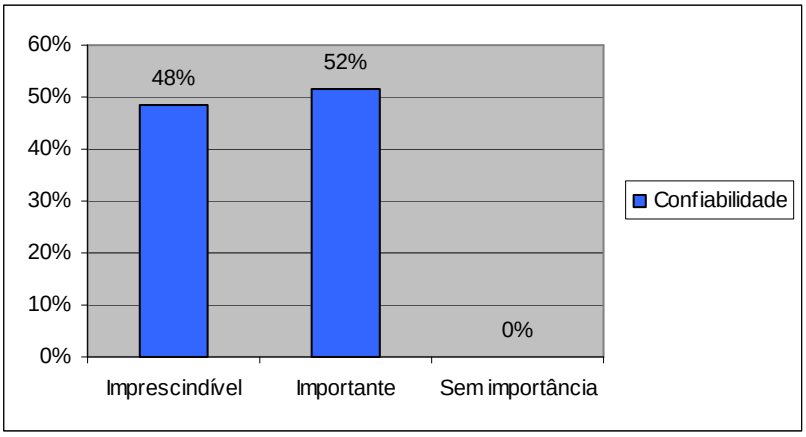

Gráfico 2.2 Característica Confiabilidade.

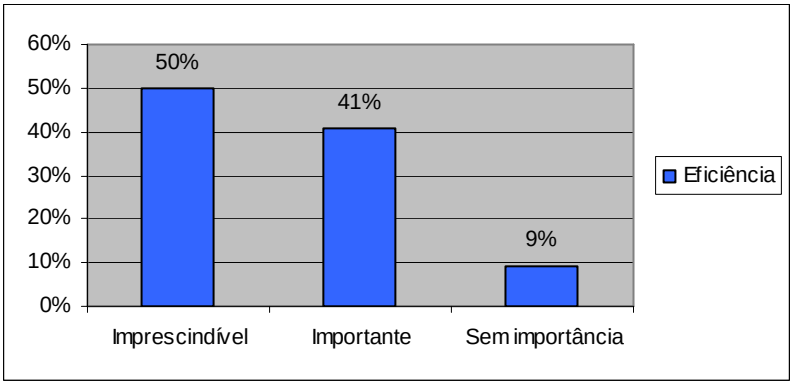

Gráfico 2.2 Característica Eficiência.

Quadro 2.3 Característica Manutenibilidade.

Atributos	Porcentagem
Imprescindível	41%
Importante	45%
Sem importância	14%

Quadro 2.4 Característica Portabilidade.

Atributos	Porcentagem
Imprescindível	9%
Importante	77%
Sem importância	14%

Verificou-se que a funcionalidade é fator imprescindível, pois está relacionada à necessidade que os cientistas têm de utilizar softwares que satisfaçam plenamente às necessidades específicas de cada projeto ou pesquisa para a qual tal software é empregado.

A usabilidade dos softwares foi considerada importante apesar de observarmos que muitos cientistas aprendem a operar os softwares que utilizam de modo informal, muito embora um software de utilização complicada não seja desejável sob pena de resultar em atrasos e transtornos em trabalho.

Observou-se também que a confiabilidade de um software foi considerada por muitos entrevistados como sendo imprescindível ou importante, uma vez que seria arriscada ou contraproducente a utilização de um software desconhecido ou cujo funcionamento não é bem dominado.

Já a relevância da eficiência como característica está baseada na necessidade de um Software Científico conseguir operar em condições plenas e de modo a se beneficiar adequadamente das feições do sistema no qual está inserido, resultando em rapidez de resposta aos comandos e precisão de resultados obtidos para cada ação.

Quanto à manutenibilidade, sua relevância entre os entrevistados se deve ao fato de que, quando a execução de algum reparo ocasional é necessária, o tempo investido nesses reparos pode ser minimizado caso o software se baseie em atributos que facilitem sua manutenção e restabelecimento de seu funcionamento normal.

Foi observado que a importância da portabilidade se deve principalmente ao fato de ser este o atributo que confere ao software a capacidade de ser transposto de um ambiente para outro. A utilidade desta característica está relacionada à possibilidade de os cientistas poderem valer-se de várias plataformas e mídias, o que pode exercer influência direta no tempo de execução de um determinado projeto ou pesquisa.

2.5 CARACTERIZAÇÃO DO SOFTWARE CIENTÍFICO

De acordo com a literatura pesquisada, no quadro 2.5 apresentamos as principais características dos Softwares Científicos em relação à qualidade do software, práticas da Engenharia de Software, equipe de desenvolvimento, processo de desenvolvimento entre outros, identificadas em Belchior (1997); Crabtree *et al.* (2009); Hannay *et al.* (2009); Heroux (2009); Hook & Kelly (2009); ISO (8402); ISO/IEC (9126-1); Letondal & Zdun, (2003 apud Berni, 2010); Maxville (2009); Moura *et al.* (2006); Nguyen-Hoan (2009); Pressman (2006);

Rocha *et al.* (2001); Sander & Kelly (2008 apud Berni, 2010); Segal (2004); Segal (2008); Segal (2009); Souza (2009). Essas características identificadas serão consideradas questões de investigação da presente dissertação.

Quadro 2.5 Características dos Softwares Científicos

	CARACTERÍSTICAS DOS SOFTWARES CIENTÍFICOS
1.	Os cientistas consideram importantes as características de qualidade de software da ISO 9126 - funcionalidade, confiabilidade, eficiência, manutenibilidade, portabilidade e usabilidade e seus atributos relacionados a Software Científico.
2.	Em geral o Software Científico não é desenvolvido através de um processo padronizado.
3.	Existem tecnologias e especialistas necessárias para desenvolver o Software Científico.
4.	As tecnologias para desenvolvimento de Software Científico podem ser adquiridas ou desenvolvidas.
5.	Nem sempre a organização dispõe de mão de obra para o desenvolvimento e operação do Software Científico em termos de conhecimento e domínio da tecnologia.
6.	Nem sempre o desenvolvimento de Software Científico conta com equipe especializada.
7.	Em geral a reutilização de artefatos científicos é realizada informalmente.
8.	A reutilização de artefatos científicos ocorre através de documentos, projetos e códigos.
9.	Os cientistas desconhecem as práticas da Engenharia de Software
10.	Muitas vezes o próprio cientista é o desenvolvedor do software de que necessita.
11.	A maioria dos cientistas aprende sobre desenvolvimento de software de maneira informal.
12.	De uma maneira geral os cientistas têm pouco treinamento formal em Engenharia de Software.
13.	A adoção de certas medidas utilizadas no software comercial nem sempre são necessárias no Software Científico.
14.	O processo de desenvolvimento de Software Científico em geral é caótico.
15.	A imprevisibilidade e incertezas fazem parte do processo de desenvolvimento de Software Científico.
16.	Os requisitos podem sempre sofrer alterações.
17.	Pressões de tempo sempre existem no desenvolvimento de Software Científico.
18.	Qualidade não é uma preocupação constante no desenvolvimento de Software Científico.
19.	Os recursos humanos são limitados para o desenvolvimento de Software Científico.
20.	Resolução do problema é a prioridade no desenvolvimento de Software Científico.
21.	Atualização das ferramentas não é uma prioridade no desenvolvimento de Software Científico.
22.	Para cientistas software funcionando é mais importante que documentação completa.
23.	Colaboração é muito importante no desenvolvimento de Software Científico.
24.	Adaptação às mudanças é mais importante que seguir um plano.

3 *WORKFLOW* CIENTÍFICO

A necessidade de sistemas capazes de isolar o fluxo de tarefas da automatização de processos de escritório foi identificada nos anos 70. Ao separar o fluxo do restante do sistema, ganhou-se flexibilidade e agilidade quando modificações eram necessárias. Além disso, aos profissionais especialistas em áreas de negócios puderam ser atribuídas as responsabilidades por essas tarefas, sem que fossem necessários conhecimentos de programação (Nardi, 2009).

O tema ganhou a atenção de diversas empresas, que constituíram a *Workflow Management Coalition* (*WfMC*) em 1993. A *WfMC* definiu *workflow* como a automatização de um processo de negócio ou de parte dele, englobando documentos, informações ou tarefas que são passadas entre os participantes do processo, de acordo com regras de negócios pré-definidas. Um processo de negócio é uma instância de uma tarefa bem definida que é freqüentemente repetida como parte de uma regra de negócio (WfMC, 1995 apud Silva, 2010).

O conceito de *workflow* também pode ser definido, em linhas gerais, como um modelo que define o fluxo de processos ou fluxo de tarefas coordenadas e encadeadas usando um plano sistemático. Diz respeito à organização de um conjunto de procedimentos, em que documentos, informações ou tarefas são trocadas entre os participantes. A distribuição de tarefas entre os participantes é baseada em regras e requisitos, constantes na definição do *workflow*. A colaboração de todos os participantes é necessária para alcançar um objetivo global de um processo, podendo ser realizada manualmente ou automaticamente (Fernandes e Novais, 2007 apud Silva, 2010).

Além das aplicações para sistemas de negócios, *workflows* também fazem parte do cenário científico, que acrescenta algumas complexidades, como (Nardi, 2009):

- ✓ Fluxos com grande número de etapas, comum no processamento de tarefas como ocorre em bioinformática, química, astronomia, agronomia, entre outras;
- ✓ Volatilidade dos fluxos, que devem poder ser alterados freqüentemente, na avaliação de hipóteses científicas;
- ✓ Necessidade de parametrização para grande número de tarefas, por exemplo, em buscas por padrões no genoma;
- ✓ Monitoramento e acompanhamento da execução, que deve levar em conta a dinamicidade dos fluxos;
- ✓ Execução em ambientes cujos recursos sejam desconhecidos *a priori*.

Segundo Gil *et al.* (2007), os avanços científicos significantes são cada vez mais realizados por conjuntos complexos de análises de dados e cálculos. Esses cálculos podem

compreender milhares de passos, onde cada passo pode integrar modelos diversos e fontes de dados desenvolvidos por grupos diferentes. As aplicações e os dados também podem ser distribuídos no ambiente de execução. A reunião e a gerência de cálculos distribuídos destes conjuntos complexos apresentam muitos desafios, e a pesquisa científica cada vez mais ambiciosa está empurrando continuamente os limites da tecnologia atual. Os *workflows* surgiram recentemente como um paradigma para representar e gerenciar computações científicas complexas e distribuídas e por isso aceleram o passo do progresso científico. Os *workflows* científicos capturam as transformações de dados individuais e passos de análise bem como os mecanismos para executá-los em um ambiente distribuído. Cada passo no *workflow* especifica um processo ou o cálculo para ser executado (p.ex., um programa de software a ser executado, um serviço de web a ser invocado). Os passos são ligados segundo o fluxo de dados e as dependências entre eles. A representação desses *workflows* computacionais contém muitos detalhes necessários para executar cada passo de análise, inclusive o uso de recursos de armazenamento e execução específicos em ambientes distribuídos, os sistemas de *Workflow* podem explorar essas representações explícitas dos processos computacionais complexos para dirigir o seu ciclo de vida e automatizar a sua execução. Os *workflows* podem capturar processos de análise complexos a vários níveis da abstração, e também fornecer a informação sobre proveniência necessária para reprodutividade científica, publicação de resultado, e compartilhamento de resultados entre colaboradores. Fornecendo formalismo e apoiando automação, os *workflows* têm o potencial para acelerar e transformar o processo de análise científica.

Gil *et al.* (2007) menciona ainda que os sistemas de *workflow* existentes foram demonstrados em várias aplicações científicas, foram *workflows* compostos de milhares de conjunto de dados grandes, distribuídos e processados em recursos computacionais de alta qualidade.

Os *workflows* fornecem meios sistemáticos e automatizados para a condução de análises de conjuntos de dados e integração de aplicações. É uma forma de capturar processos cujos resultados podem ser reproduzidos e métodos podem ser revisados, validados, repetidos e adaptados. O *workflow* fornece uma interface por meio da qual os cientistas podem criar fluxos de trabalho sem a necessidade de programação de baixo nível. O *workflow* consiste em uma plataforma para integração e acesso ao crescente número de recursos computacionais independentes, disponíveis para cientistas que não possuem conhecimento especializado sobre cada um dos componentes ou sobre computação (Goble e Roure, 2009 apud Silva, 2010).

O *workflow* é um elemento importante para a pesquisa científica. Mais especificamente em *e-Science*, temos os *workflows* científicos que são voltados para explicitar representações de processos experimentais científicos, geralmente de natureza

colaborativa (Fernandes e Novais, 2007 apud Silva, 2010). Para facilitar o trabalho dos pesquisadores, poupando tempo e possibilitando descobertas cada vez maiores e mais confiáveis, os processos científicos são mapeados para *workflows.* Com a utilização de *workflows,* serviços são associados a cada tarefa de experimentos *in silico*[1], por exemplo (Mattos e Mattoso, 2007 apud Silva, 2010).

O termo *e-Science* geralmente é empregado para descrever o desenvolvimento de infraestruturas de serviços de software capazes de prover acesso a facilidades remotas, recursos computacionais distribuídos, armazenamento de informações em bancos de dados dedicados, disseminação e compartilhamento de dados, resultados e conhecimento. Em essência, segundo Lauschner (2005 apud Palazzi, 2010):

> Um ambiente de *e-Science* deve permitir o compartilhamento de recursos coordenados em larga-escala entre comunidades dinâmicas de indivíduos, grupos, laboratórios e instituições, permitindo o gerenciamento cooperativo de facilidades (equipamentos, instrumentação, experimentos entre outros) e análise colaborativa de produtos (dados) oriundos dessas facilidades.

São características de *e-Science* o acesso a uma vasta coleção de dados, utilização de recursos computacionais em larga escala, utilização de recursos heterogêneos e dinâmicos de múltiplas organizações e *workflows* (Cardoso, 2007; Matos *et al.*, 2007 apud Palazzi, 2010).

3.1 CARACTERÍSTICAS

Muitas pesquisas são dirigidas e encaminhadas às questões de criação, reutilização, rastreamento de proveniência, otimização e confiança de *workflows* científicos. Contudo, para realizar totalmente a promessa de tecnologias de *workflow*, muitas exigências adicionais e desafios devem ser encontrados. As aplicações científicas são sistemas que conduzem o *workflow* a examinar questões como o apoio a análises dinâmicas orientada a eventos, manipulação de dados, interação com usuários, ajuda inteligente e suporte colaborativo para desenhos de *workflow*, permitindo assim, o compartilhamento de resultados através de todas essas colaborações. Gil *et al.* (2007) menciona que um tratamento mais abrangente dos *workflows* é necessário para satisfazer as condições específicas das aplicações científicas futuras e dos desafios que elas apresentam aos *workflows* atuais.

[1] A expressão *in silico* é, originalmente, usada para denotar simulações computacionais que modelam um processo natural ou de laboratório (Lemos, 2004 apud Silva, 2010).

Gil *et al.* (2007) ressalta ainda as características dos *workflows* científicos e analisa os contextos em que esse esforço de pesquisa deve se concentrar em quatro itens que descrevemos a seguir:

- ✓ Requisitos da Aplicação;
- ✓ Descrições de dados e *workflows*;
- ✓ Processos dinâmicos tecnológicos e direção de usuário;
- ✓ Nível de sistema e gerência de *workflow*.

I - Requisitos da aplicação

As análises científicas complexas cada vez mais necessitam de um grande esforço e coordenação humana. Os dados estão crescendo consideravelmente, mas o número de cientistas é rudemente constante. Assim os pesquisadores precisam de instrumentos exponencialmente mais eficazes para ajudar no seu trabalho, já que eles trabalham com muitos dados e tarefas associadas. Os ambientes de *workflow* que apóiam e melhoram o processo científico em todos os níveis são cruciais se quisermos manter a atual taxa de crescimento e de processamento de dados.

A capacidade de combinar dados distribuídos, computação, modelos, e instrumentos em escalas sem precedentes podem permitir a pesquisa transformável. A análise de grandes conjuntos de dados largamente distribuídos está ficando comum. Esses dados, e o aparelho experimental ou os sistemas de simulação que os produzem, tipicamente não pertencem a indivíduos, mas a colaboradores. Dentro dessas colaborações, vários indivíduos são responsáveis por aspectos diferentes de aquisição de dados, processamento, e análise, e nos quais as publicações muitas vezes são geradas por projetos inteiros. Tais ambientes exigem instrumentos que podem orquestrar os passos da descoberta científica e lançar uma ponte entre a perícia que se diferencia dos membros da colaboração.

Uma exigência de aplicação importante identificada é a reprodutividade de análises científicas e processos. Esta exigência está no núcleo do método científico, em que ele permite aos cientistas avaliar a validade de cada hipótese de outro e fornece a base para estabelecer verdades conhecidas. A reprodutividade necessita da informação rica sobre proveniência, para que os pesquisadores possam repetir técnicas e métodos de análise para obter resultados cientificamente semelhantes. Hoje, a reprodutividade é praticamente impossível para aplicações científicas complexas. Primeiro, porque muitos cientistas estão envolvidos, os registros de proveniência são altamente fragmentados e, na prática elas são

refletidas em uma variedade de elementos, incluindo e-mails, entradas de Wikipédia, consultas de banco de dados, referências a periódicos, códigos (inclusive opções de compilador), entre outros. Toda esta informação, muitas vezes fornecida em vários locais e de várias formas, tem de ser apropriadamente posta à disposição para referência. Sem seguir esta pista e integrar os resultados da análise, a reprodutividade pode ser basicamente impraticável, e mais provável, impossível, para muitas descobertas importantes que implicam em computações complexas.

Para apoiar a reprodutividade, os sistemas de gerência de *workflow* devem capturar e gerar a informação sobre proveniência como uma parte crítica dos dados gerados por *workflows*. Os sistemas de gerência de *workflow* também devem consumir a informação sobre proveniência associada com dados de entrada, e associada à informação com os produtos de dados resultantes. A proveniência deve associar-se com os novos produtos de dados e conter bastante detalhes para permitir a reprodutividade. Outra exigência importante é relacionada a repositórios interoperáveis, persistência de dados e definições de análise, com ligação para divulgar dados e publicações, bem como algoritmos e aplicações usadas para transformar os dados. Os repositórios de dados existentes devem ser completados com proveniência e repositórios de metadados que permitem a descoberta dos *workflows* e componentes de aplicação que foram usados para criar os dados.

Os ambientes fornecidos também devem ser flexíveis quanto ao apoio, tanto às análises comuns executadas por muitos, como análises individuais únicas. As análises de rotina baseadas em casos comuns devem ser fáceis de estabelecer e executar. Ao mesmo tempo, os cientistas individuais devem ser capazes de dirigir o sistema para conduzir análises únicas e criar *workflows* novos com combinações anteriormente não vistas e configurações de modelos.

De uma perspectiva operacional, há necessidade de fornecer soluções que sejam seguras, viáveis, e escaláveis. Os cientistas têm de ser capazes de confiar que os seus dados de produção e entrada são seguros para acesso de dados impróprios ou de manipulação maliciosa. A confiança e os sistemas de reputação de fornecedores de dados devem ser incorporados na infra-estrutura atual. Os instrumentos têm de ser escaláveis para apoiar análises grandes e complexas, dados de tamanho de *terabites* e grandes comunidades científicas.

Um assunto importante é como dirigir a heterogeneidade inevitável e inconsistências que surgem quando a informação vem de fontes e comunidades diferentes. Mecanismos de curadoria, validação, tradução e integração de dados são necessários para que a informação científica seja compartilhada de forma significativa e verdadeiramente integrativa.

Finalmente, os cientistas precisam usar instrumentos fáceis que forneçam ajuda

inteligente para tais capacidades de *workflows* complexos. Automação dos aspectos operacionais de baixo nível dos *workflows* é uma exigência-chave. As modalidades de interação que ocultam complexidades desnecessárias e falam a linguagem do cientista serão cruciais ao êxito. A orientação a usuários será útil para estimular as melhores práticas científicas.

II - Descrições de dados e *workflows*

Cientistas sempre confiaram em tecnologia para compartilhar informação sobre experiências, de caneta e papel para máquinas fotográficas digitais, e-mail, Web, e software de computador. Descrições de *Workflow* e capacidades de execução oferecem um modo novo de compartilhar e administrar informação: aquela em que os processos completos podem ser capturados eletronicamente e compartilhados para futura referência e reutilização. Este novo modo de compartilhar informação semântica e de chegar a um acordo sobre os próprios processos, tendo assim, uma melhora na infra-estrutura de apoio a execução da informação.

Os cientistas devem ser estimulados a trazer representações de *workflow* para as suas práticas e compartilhar as descrições de suas análises científicas e cálculos, de maneira tão formal e explícita quanto possível. No entanto, não há representações comumente aceitas e suficientemente ricas na comunidade científica. Assim, mais pesquisas nessa área são ainda necessárias.

As representações de *workflow* têm de acomodar descrições de processo científico a múltiplos níveis. Por exemplo, os cientistas de domínio podem querer uma interface gráfica sofisticada para compor passos científicos ou matemáticos relativamente de alto nível, ao passo que os cientistas de computação podem ser mais preocupados com o uso de uma linguagem de *workflow*, e com as especificações detalhadas dos dados e passos de execução. Para relacionar essas visões e fornecer as capacidades necessárias, as representações de *workflow* devem incluir descrições ricas com vários níveis de abstração, e devem incluir modelos de como fazer o mapeamento entre eles. Além disso, para apoiar a descrição multidisciplinar final e de pesquisa, definições de *workflow* e de proveniência devem ser amplos o suficiente para descrever os *workflows* que estão ligados através de dados de referência, modelos apoiados por validação de *workflows*, literatura e processos manuais em geral.

Outras dimensões importantes da abstração são críticas de experimentos contra não experimentos, onde o antigo se refere às questões científicas e o último é mais

preocupado com matérias operacionais. Sendo assim, um sistema de *workflow* deve apoiar ambos experimentos.

A informação rica de processos de análise tem de ser incorporada em representações de *workflow* para apoiar esta descoberta, ou seja, a criação, a fusão, e a execução. Essas atividades serão um modo natural de conduzir experimentos e compartilhar a metodologia científica dentro e através de comunidades científicas.

As representações de *workflow* têm de apoiar, sempre que possível, a automação da criação de *workflow* e processos de gerência. Esta capacidade vai exigir ricas representações semânticas dos requisitos e restrições em modelos de *workflows* e componentes. Com descrições semânticas do formato de dados e as exigências de tipo do componente, é possível incorporar o raciocínio automatizado e capacidade de planejamento que pode acrescentar automaticamente a conversão de dados e a transformação. Da mesma forma, com descrições ricas dos requisitos de execução de cada componente de *workflows*, a seleção de recursos automatizados e otimizações dinâmicas seriam possíveis.

Um desafio para a comunidade de Ciências da Computação é ser capaz de manipular os vários níveis de abstração do *workflow* simultaneamente e manipulá-los individualmente.

Outra questão de pesquisa importante é se os *workflows* científicos podem ou até devem explorar tecnologias de *workflow* existentes, ou se eles necessitam de aproximações fundamentalmente novas. Os *workflows* foram usados durante décadas para representar e dirigir processos de negócios. Lá estão emergindo padrões de representações de *workflow* bem como software associado (um pouco da qualidade comercial) para dirigir *workflows*. Entender as diferenças entre *workflows* científicos, as práticas e os usados no negócio podem produzir discernimentos úteis. De um lado, os *workflows* científicos e de negócios não são obviamente distinguíveis, já que ambos podem compartilhar características importantes comuns. De fato, na literatura, encontramos exemplos de *workflows* em ambos os domínios que são dados únicos, altamente paralelos, etc. De outro lado, a pesquisa científica necessita de desenho flexível e capacidades de exploração que parecem partir significativamente do uso mais consagrado do *workflow* no negócio; os *workflows* na ciência são um meio de apoiar o discurso científico detalhado bem como um modo de permitir processos repetíveis.

Outra questão distintiva de *workflows* científicos é a variedade e a heterogeneidade de dados dentro de um *workflow* e um assunto importante deve equilibrar o desejo de compartilhar a informação sobre *workflow* contra os perigos de esforços de padronização prematuros que podem forçar futuras exigências e capacidades. Neste aspecto, será crucial estimular cientistas de computador e cientistas de domínio a colaborar estreitamente no desenvolvimento de mais aplicações à base de *workflows* e discutir exigências de

representação de futuros *workflows*.

Um desafio importante na ciência é a representação de variantes de *workflow*, que aponta para a compreensão do impacto que uma modificação tem nos produtos de dados resultantes como uma ajuda ao discurso científico.

Em alguns casos, é conveniente compartilhar *workflows*, mas não dados. Em outros casos, os cientistas querem compartilhar uma descrição abstrata do protocolo científico, sem comunicar de fato detalhes, parâmetros e configurações. Em outras situações, ele é uma descrição de uma execução prévia específica (a proveniência) que é desejável, com ou sem fornecer detalhes de execução.

III - Processos dinâmicos tecnológicos e direção de usuário

A visão de apoiar *workflows* dinâmicos, adaptáveis e dirigidos por usuário deve permitir e acelerar a distribuição e metodologia científica colaborativa via rápida reutilização, exploração, adaptação e melhoria contínua.

A prática científica costumeiramente dará origem a *workflows* que são dinâmicos, onde as decisões tomadas sobre os passos que serão seguidos são baseados em informações mais recentes disponíveis. Um *workflow* pode ter de ser dinamicamente projetado no tocante a ver os resultados das etapas iniciais antes que uma decisão possa ser tomada sobre como executar etapas de análise posteriores. Por exemplo, examinando os resultados de algum reprocessamento inicial de uma imagem, etapas subseqüentes poderão ser necessárias para áreas específicas identificadas por algum pré-processamento. Um *workflow* dinâmico também pode ser aquele onde a estrutura básica ou a semântica se modificam por causa de algum evento externo. Outro cenário que inclui *workflows* dinâmicos é onde dois *workflows* podem afetar um a outro, por exemplo, compartilhando resultados. Eles podem ser classificados tão dinâmicos como eles respondem a eventos que surgem em cada execução de outro. Finalmente, algumas diligências científicas são amplas. Eles implicam em grandes equipes de cientistas e técnicos, e ocupam-se em métodos experimentais ou procedimentos que tomam longos tempos para concluir e necessitar da intervenção humana e da direção dinâmica em todas as partes do processo.

A gerência de *workflows* dinâmicos é complexa devido à sua evolução e ciclo de vida. Não há nenhum começo ou fim do processo de ciclo de vida de um *workflow* – os cientistas podem começar em qualquer ponto e fluir pela figura em qualquer direção.

Apoiar os cientistas em complexos processos exploratórios de *workflows* dinâmicos é um desafio importante. Um sistema de apoio à decisão com acesso humano acomoda as

necessidades de informação de um cientista para monitoramento e compreensão de processos complexos terão de ser concebidos. Interfaces adequadas que permitam aos cientistas navegar no que é necessário, consultar, re-capitular e entender esta informação serão necessárias. Simplificação do processo exploratório também necessita de meios novos e escaláveis para que cientistas manipulem os *workflows*, explorem fatias do espaço de parâmetro, e comparem os resultados de configurações diferentes. É fundamental que os *workflows* se reúnam, encontrando serviços e façam a adaptação prévia desses *workflows*.

IV - Nível de sistema e gerência de *workflow*

Um desafio-chave em *workflows* científicos é assegurar a reprodutividade de engenharia para permitir a re-execução de análises, e a réplica de resultados. A reprodutividade científica implica que alguém pode seguir a metodologia geral, confiando nos mesmos dados iniciais, e obter resultados equivalentes. A reprodutividade de engenharia necessita mais conhecimento das manipulações de dados, do software real e do ambiente de execução (hardware, bibliotecas específicas), etc., para que os resultados possam ser bit-a-bit duplicados. A capacidade anterior é necessária quando os pesquisadores querem validar cada uma das hipóteses do outro, considerando que o posterior é benéfico quando são encontrados resultados incomuns ou erros e a pesquisa precisa ser localizada e compreendida. As informações necessárias para apoiar ambos os tipos de reprodutibilidade são desafios para a captura. Ao apoiar a reprodutividade científica, em alto nível, contudo, significante, a descrição do processo deve ser fornecida. A reprodutividade da engenharia também exige a informação de baixo nível como que *flags* de compilador foram usadas para compilar um determinado código e os detalhes do ambiente de execução e arquitetura de computador.

Um desafio importante deverá fornecer uma visão estável no sistema apesar de modificações contínuas em tecnologias e plataformas ao nível de sistema. O sistema de execução subjacente deve ser projetado para que ele forneça um ambiente estável das camadas de software que dirigem o processo científico de alto nível. Deve ser possível re-executar *workflows* muitos anos depois e obter os mesmos resultados. Esta exigência coloca desafios em termos de criação de uma camada estável de abstração sobre uma infra-estrutura em constante evolução, proporcionando a flexibilidade necessária para atender à evolução das necessidades e aplicações para suportar as novas capacidades. Para fornecer o acesso consistente e eficiente a recursos, a gerência destes deve considerar ambos os recursos físicos (p.ex., computadores, redes, servidores de dados) e

recursos lógicos (p.ex., repositórios de dados, programas, componentes de aplicação, *workflows*). Ambos devem ser expostos por interfaces uniformes. Realçando descrições de recurso com anotações semânticas, o fornecimento, a proveniência, a configuração e o desdobramento de novos recursos podem ser organizados mais facilmente e possivelmente até automatizados. Aumento de serviços de informações atuais com a descrição semântica significativa de recursos deveria permitir a descoberta semi-automática, a intermediação e a negociação. A interação humana deve ser minimizada por configuração dinâmica e gerência de recursos de ciclo de vida, alguns esforços foram feitos para fornecer a descoberta semi-automática e fazer intermediação de recursos físicos e gerência de componentes de software que podem ficar à parte de ambientes de *workflows* científicos. Contudo, há ainda muita oportunidade de melhoras, desde que a maior parte de sistemas existentes necessitam do desdobramento manual ou semi-manual de componentes de software e forcem construtores de aplicação a posições de componente de software *hardcode* em recursos específicos nos seus *workflows*. Adicionalmente, os serviços de informações atualmente disponíveis não são bem adaptados para fornecer a descrição completa de componentes de software, forçando o construtor de aplicação a usar somente (nome, posição) - informação sobre estilo de serviços disponíveis e recursos. Por conseguinte essas aplicações são sensíveis a modificações dinâmicas na infra-estrutura de recursos, e, muitas vezes, falham durante a execução devido a fracassos evitáveis.

3.2 REUTILIZAÇÃO

Os *workflows* científicos estão emergindo rapidamente como um paradigma para representar e gerenciar dados complexos de computação, ajudando os cientistas a conceituar e gerenciar a análise, a visualização e a integração de dados. As bases para permitir tais *workflows* são definidas pela adoção de SOA (*Service Oriented Architecture*), no qual as funções são separadas em unidades distintas (serviços), e as interfaces e protocolos padronizados tais como WSDL (*Web Services Description Language*) e SOAP (*Simple Object Access Protocol*) que fornecem a base de sistemas livremente ligados e distribuídos. Esses serviços então podem ser acessados por qualquer cliente desde que estejam compatíveis com os padrões, e possam ser combinados e reutilizados como criação de blocos de *workflows* mais complexos (Langguth *et al.,* 2009).

Desde que um *workflow* é essencialmente uma agregação de múltiplas chamadas de serviço, a realização de um *workflow* depende diretamente da realização (de um todo) de suas atividades (Langguth *et al.,* 2009).

De acordo com Xiang *et al.* (2007), com um grande número de dados, os instrumentos de análise, e outros recursos são entregues como serviços na web, cujo benefício principal é de adotar arquitetura orientada a serviços na e-Science, que permite que os cientistas descrevam e ordenem o seu processo experimental, supervisionando os serviços distribuídos e locais em um *workflow*.

A maioria destes sistemas e métodos fornecem aos usuários um ambiente para compor serviços, desde o início, de forma mais exata na escolha dos serviços apropriados com a composição de correspondência semântica e qualidade de serviços. Alguns deles consideram o potencial de reutilização e compartilhamento do conhecimento adquirido durante o processo de composição de serviços e a reutilização de todo ou parte de *workflows* existentes. Ao proporcionar capacidade de reutilizar os conhecimentos, os *workflows* tornam-se um componente importante nesse sistema. Essa reutilização conduzirá a um processo de composição mais eficaz e mais estruturado que vai acelerar o desenvolvimento rápido de aplicações. Ele irá fornecer orientações mais valiosas para auxiliar os usuários na sua criação de *workflow* usando o conhecimento que tem sido adquirido e verificado por outros (Xiang *et al.*, 2007).

As metodologias práticas mais atuais de composição do serviço ou criação de *workflow* empregam um desenho semi-automático que permite que usuários descubram e selecionem serviços apropriados para incluir em um *workflow* baseado em definições de serviços semânticos e conceituais. Isto parcialmente aumenta a capacidade dos usuários e a necessidade do conhecimento detalhado para entender cada instrumento, serviço e tipo de dados. Entretanto, ele desdobra a complexidade de construir tal *middleware* para suportar a criação de *workflow* em um nível alto de abstração. Vários sistemas de gerência de *workflow* e orientados a serviços são desenvolvidos com a intenção de modernizar o projeto do *workflow*, a execução, a monitorização e documentação do *workflow* (Xiang *et al.*, 2007).

Vários benefícios são vistos com esta definição de estrutura hierárquica de *workflow* (Xiang *et al.*, 2007):

- ✓ Permite que usuários definam o *workflow* em níveis de abstração diferentes. Os usuários menos experientes podem definir um *workflow* quanto a funções que eles desejam que uma tarefa deva executar. Os usuários podem definir um *workflow* com propriedades mais detalhadas de cada tarefa, como o algoritmo e fonte de dados que eles podem querer usar. Os usuários experientes podem ser capazes de definir um *workflow* para este caso de aproximação selecionando serviços apropriados executáveis e formar um *workflow* com lógica apropriada.
- ✓ Permite a reutilização completa ou parcial de *workflows* definidos a níveis diferentes. A reutilização de um *workflow* pode ocorrer quando os usuários têm de duplicar os

seus conjuntos de dados ou definir o mesmo *workflow* que usa dados de entrada diferentes.

- ✓ A reutilização de um *workflow* também pode ocorrer durante o projeto de *workflow*. Por exemplo, um cientista pode ter uma representação de alto nível ou a representação parcial de um *workflow*, a pesquisa e o repositório de *workflow* podem devolver um número de *workflows* semelhantes em um nível abstrato e/ou nível concreto. Este cientista pode escolher um candidato para reutilizar ou modificar o *workflow* para encontrar o resultado.

3.3 TAXONOMIA

Segundo Yu *et. al* (2005), com o advento dos Grids e tecnologias de aplicação, os cientistas e os engenheiros estão construindo aplicações cada vez mais complexas para dirigir e processar grandes conjuntos de dados, e executar experimentos científicos em recursos distribuídos. Tais cenários de aplicação necessitam de meios de compor e executar *workflows* complexos. Por isso, muitos esforços foram feitos em direção ao desenvolvimento de sistemas de gerência de *workflow* da computação em Grid. Sendo assim, foi proposta uma taxonomia que caracteriza e classifica a construção e execução de *workflows* em Grids. A taxonomia não só destaca o projeto e semelhanças de engenharia e diferenças "do estado da arte" em sistemas de *workflow* em Grid, mas também identifica as áreas que necessitam de mais pesquisa.

Os Grids surgiram como uma ciber-infraestrutura global da nova geração de aplicações de e-Science, integrando recursos amplos, distribuídos e heterogêneos. As comunidades científicas como de física de alta energia, física de onda gravitacional, geofísica, astronomia, e bioinformática, estão utilizando Grids para compartilhar, dirigir e processar grandes conjuntos de dados. Para apoiar experimentos científicos complexos, os recursos distribuídos como dispositivos computacionais, dados, aplicações, e instrumentos científicos têm de ser orquestrados enquanto operações de *workflow* gerenciam dentro de ambientes de Grid (Yu *et. al,* 2005).

O *workflow* científico preocupa-se com a automação de processos científicos nos quais as tarefas são estruturas baseadas no controle e dependências de dados. O paradigma de *workflow* de aplicações científicas em Grids oferece várias vantagens, como (Yu *et. al,* 2005):

(a) Capacidade de construir aplicações dinâmicas que orquestram recursos distribuídos;

(b) Utilização de recursos que são localizados em um determinado domínio para aumentar a quantidade tratada ou reduzir preços de execução;

(c) Execução abrangendo vários domínios administrativos para obter capacidades de processamento específico; e

(d) Integração de múltiplas equipes lotadas na gerência de partes diferentes do *workflow* de experimento, assim promovendo colaborações interorganizacionais.

Em um passado recente, vários sistemas de *workflow* tecnológico de Grid foram propostos e desenvolvidos para a definição, arranjando-se na execução de *workflows* científicos. Para facilitar o entendimento (Yu *et. al,* 2005) propôs uma taxonomia que principalmente captura estilos arquitetônicos e identifica projeto e semelhanças de engenharia e diferenças entre eles (figura 3.1).

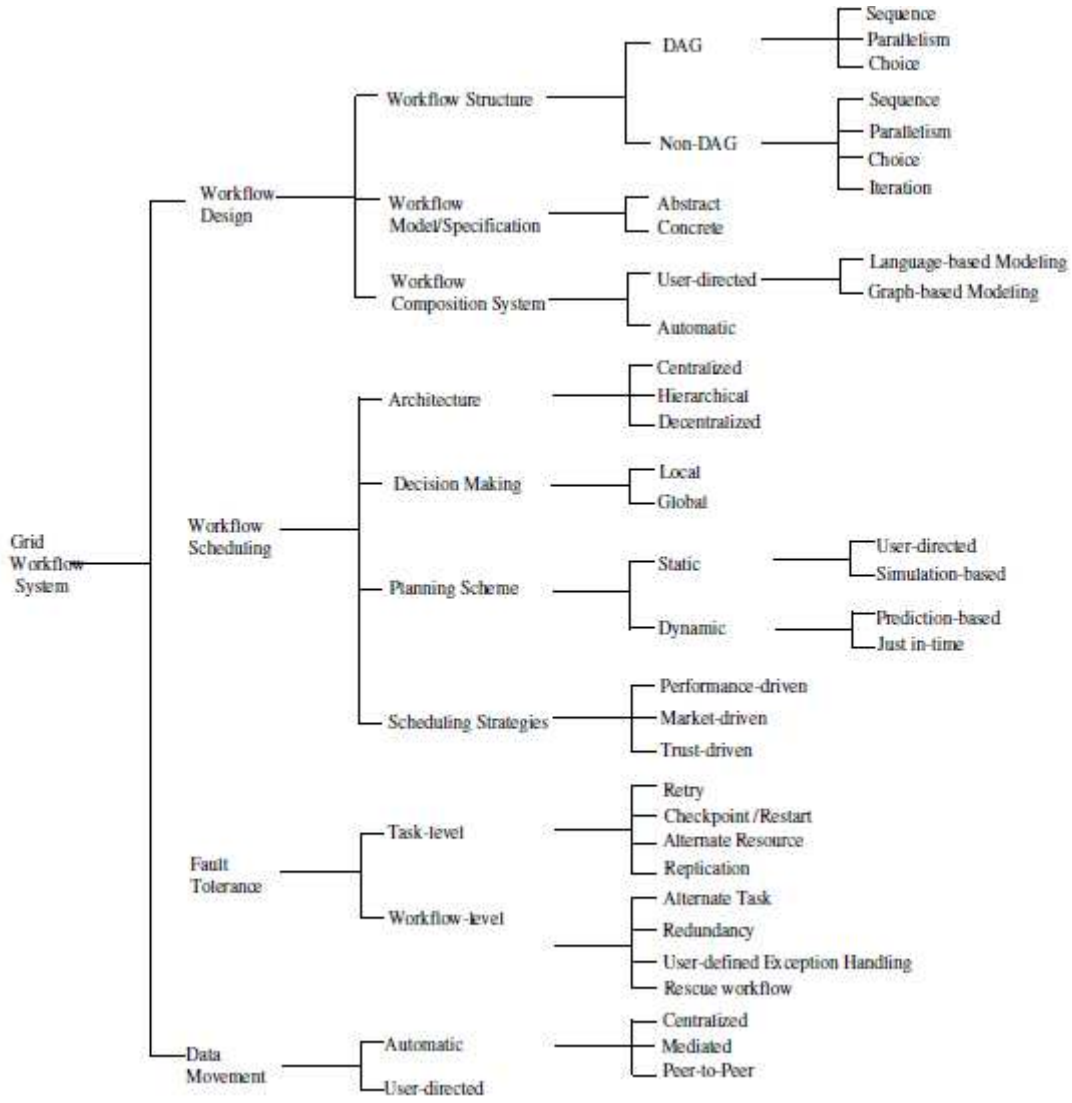

Figura 3.1 – Uma Taxonomia de *Workflow* Científico para Computação em Grid (Yu *et. al,* 2005).

A taxonomia caracteriza e classifica aproximações de sistemas de *workflow*

científicos no contexto da computação em Grid. Ela compõe-se de quatro elementos de um sistema de gerência de *workflow* em Grid (Yu *et. al,* 2005):

(a) projeto de *workflow*, que determina como os componentes de *workflow* podem ser definidos; (b) planejamento de *workflow*, concentra-se em fazer o mapa da execução e de direção de tarefas de *workflows* em recursos compartilhados que não são diretamente controlados no sistema de *workflow*; (c) falta de tolerância, em ambientes de Grid, os fracassos de execução de *workflow* podem ocorrer por várias razões como fracasso de rede, condições de recursos sobrecarregados, ou indisponibilidade de componentes de software necessários. Assim, os sistemas de gerência de *workflow* de Grid devem ser capazes de identificar e detectar/tratar fracassos e apoiar a execução confiável na presença de conformidade de opiniões e fracassos; e (d) movimento de dados, classifica como centralizado, mediado e *peer-to-peer*. Uma aproximação centralizada transfere dados intermediários entre recursos via um ponto central. Em uma aproximação mediada antes de usar um ponto central, as posições dos dados intermediários são dirigidas por um sistema de gerência de dados distribuído. Uma aproximação de *peer-to-peer* transfere dados entre o processamento de recursos.

A execução de experimentos científicos com base em simulações de computador constitui uma contribuição importante para a comunidade científica. Nesse sentido, a implementação de um *workflow* científico pode ser automatizado por sistemas de gerenciamento de *workflow* científico, cujo objetivo é proporcionar a junção de todos os processos envolvidos. Tem como objetivo capturar a semântica envolvida na implementação de *workflows* científicos usando ontologias que possam capturar o conhecimento envolvido nestes processos (Gaspar *et*. *al*, 2010) .

3.4 CARACTERIZAÇÃO DO *WORKFLOW* CIENTÍFICO

De acordo com a literatura pesquisada no quadro 3.1 apresentamos as principais características dos *workflows* científicos desejáveis, identificadas em Cardoso (2007); Matos *et al.* (2007apud Palazzi, 2010); Fernandes e Novais (2007 apud Silva, 2010); Goble e Roure (2009 apud Silva, 2010); Gil *et al.* (2007); Langguth *et al.* (2009); Lauschner (2005 apud Palazzi, 2010); Mattos e Mattoso (2007 apud Silva, 2010); Nardi (2009); Wfmc (1995 apud Silva, 2010); Xiang *et al.*(2007); Yu *et. al* (2005); Gaspar *et*. *al*. (2010). As características desejáveis serão investigadas nessa dissertação, através da validação da mesma com os pesquisadores.

Quadro 3.1 Características dos *Workflows* Científicos

CARACTERÍSTICAS DOS *WORKFLOWS* CIENTÍFICOS	
1.	Apoiar a reprodutividade das análises e processos.
2.	Permitir a captura e geração de proveniência de dados.
3.	Apoiar soluções seguras, confiáveis e escaláveis.
4.	Tratar a heterogeneidade e inconsistências de informações.
5.	Prover assistência inteligente.
6.	Prover diferentes formas de representação.
7.	Permite a representação em diferentes níveis de abstração.
8.	Possuir representações para etapas de descoberta, junção, descrição e execução do workflow.
9.	Prover suporte para automação da criação e gerenciamento de processos.
10.	Permitir reexecução de processos.
11.	Aceitar variedade e heterogeneidade de dados.
12.	Apoiar colaboração e compartilhamento de práticas.
13.	Permitir a criação de workflows dinâmicos.
14.	Permitir a especificação da qualidade dos serviços solicitados.
15.	Assegurar a escalabilidade da execução.
16.	Possuir infraestrutura para gerenciamento do workflow.
17.	Permitir a reutilização de parte ou todo workflow.
18.	Ser tolerante a falhas.

4 ESTUDO DE CASO: CARACTERIZAÇÃO DO SOFTWARE CIENTÍFICO EM MODELAGEM COMPUTACIONAL

O estudo de caso aqui apresentado foi elaborado para investigar as características do desenvolvimento do Software Científico, através de um estudo exploratório, com um grupo de pesquisadores do Laboratório Nacional de Computação Científica - LNCC, especialistas na área de Modelagem Computacional.

A presente pesquisa é de campo e do tipo quantitativa com o objetivo de conferir hipóteses identificadas na literatura. Usa técnicas de coleta de dados através de um questionário. Esse instrumento de pesquisa foi composto principalmente por perguntas fechadas. A escolha de tal instrumento justifica-se por utilizar poucas pessoas para sua execução, proporcionando economia de custo, tempo, viagens e não sofre influência do entrevistador. O modo utilizado na aplicação do questionário foi a abordagem direta e pessoal.

Para a caracterização do Software Científico foi construído um instrumento, na forma de questionário, onde constavam as seguintes questões de investigação:

- ✓ Identificação de características de qualidade do Software Científico;
- ✓ Identificação dos modelos de processos adotados no desenvolvimento do Software Científico;
- ✓ Possibilidade de adoção de práticas da Engenharia de Software;
- ✓ Possibilidade de desenvolvimento de Software Científico por equipe especializada, que não os próprios cientistas;
- ✓ Reutilização de artefatos científicos e compartilhamento de Software Científico;
- ✓ Identificação de características dos *workflows* científicos.

O estudo de caso foi aplicado como uma atividade opcional aos participantes da reunião entre os pesquisadores, participantes da pesquisa, a pesquisadora e a orientadora desse trabalho. O questionário aplicado encontra-se no Apêndice II. A reunião motivadora e de esclarecimento durou cerca de 01h30min, na sala de reunião do grupo, no LNCC. O questionário foi entregue aos participantes do grupo de pesquisa, porém não foi dado nenhum treinamento específico sobre o conteúdo do mesmo. Quatro participantes preencheram o questionário logo após a reunião, num tempo médio de 40 minutos e outros dois o fizeram depois. A maior dificuldade foi relacionada a termos específicos da Engenharia de Software, definição e caracterização de Workflow Científico e reutilização de

artefatos científicos.

4.1 RESULTADOS DO ESTUDO

a) Identificação dos Participantes

Nesse item foram levantados sexo (gráfico 4.1), formação acadêmica (gráfico 4.2) e quanto à área de pesquisa deve-se ressaltar que todos participantes estão envolvidos com pesquisa em modelagem de Escoamentos em Meios Porosos.

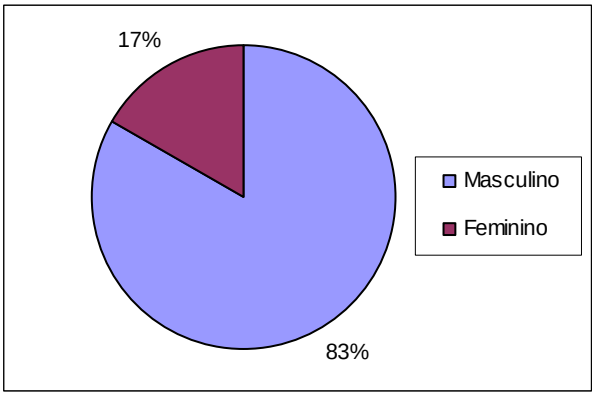

Gráfico 4.1 Sexo dos entrevistados.

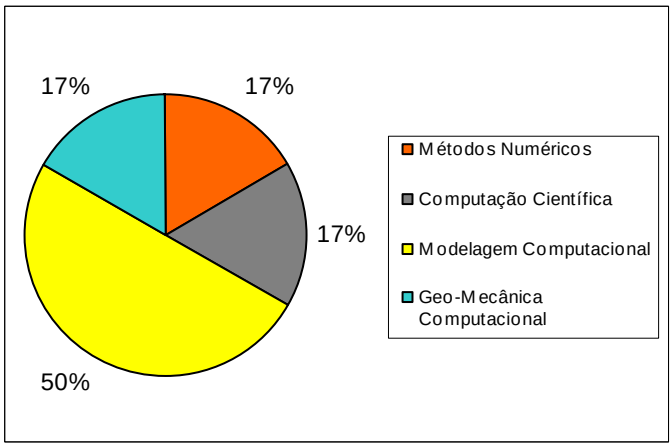

Gráfico 4.2 Área de formação dos entrevistados.

b) Características de Qualidade do Software Científico

Neste item foram observadas as características relevantes para a Qualidade do Software Científico, conforme quadro 4.1.

Quadro 4.1 Características da Qualidade do Software Científico.

Características da Qualidade do Software Científico	Resultado			
	Imprescindível	Desejável	Sem importância	Total
Funções adequadas às tarefas especificadas e aos objetivos dos usuários.	83%	17%	0%	**100%**
Resultados sempre corretos.	83%	17%	0%	**100%**
O software interagir com outros sistemas.	17%	83%	0%	**100%**
O software estar de acordo com as normas, padrões e leis.	33%	67%	0%	**100%**
O software possuir controle de acesso e segurança dos dados.	33%	50%	17%	**100%**
O software não apresentar falhas.	67%	33%	0%	**100%**
O software manter o nível de desempenho mesmo em caso de falhas.	17%	33%	50%	**100%**
O software ser capaz de recuperar dados em caso de falha.	0%	67%	33%	**100%**
O software permitir ao usuário entender se ele é adequado e como ele pode ser usado para tarefas e condições de uso específico.	75%	0%	25%	**100%**
Ser fácil aprender a utilizar o software.	50%	33%	17%	**100%**
O software ser fácil de operar e controlar.	50%	50%	0%	**100%**
A interface do software ser atrativa.	17%	33%	50%	**100%**
O software apresentar tempo de resposta e velocidade de execução adequada.	33%	50%	17%	**100%**
O software apresentar recursos necessários para o seu funcionamento.	67%	33%	0%	**100%**
Ser fácil encontrar e diagnosticar uma falha.	17%	83%	0%	**100%**
Ser fácil modificar e adaptar o software.	67%	33%	0%	**100%**
As alterações no software não trazerem grandes riscos.	17%	67%	17%	**100%**
Ser fácil testar o software quando se fizer alterações.	50%	50%	0%	**100%**
Ser fácil adaptar o software em outros ambientes.	17%	67%	17%	**100%**
Ser fácil instalar o software em outros ambientes.	17%	67%	17%	**100%**
O software estar de acordo com padrões de portabilidade.	50%	50%	0%	**100%**
Ser fácil substituir o software por outro com os mesmos objetivos.	0%	17%	83%	**100%**

c) Caracterização do Processo de Desenvolvimento de Software Científico

Neste item foram identificados os principais modelos de processos de desenvolvimento do Software Científico (quadro 4.2) e de reutilização de artefatos (quadro 4.3).

Quadro 4.2 Modelos de processos adotados no desenvolvimento do Software Científico.

Modelos de processos de desenvolvimento do Software Científico	Resultado			
	Sim	Não	Às vezes	Total
Utiliza algum processo padronizado para o desenvolvimento de Software Científico?	33%	33%	33%	**100%**
Caso utilize algum processo padronizado, ele atende às necessidades de construção de Software Científico?	17%	50%	33%	**100%**
Existe a tecnologia necessária para desenvolver o software especificado?	83%	0%	17%	**100%**
A equipe encarregada pelo desenvolvimento dispõe da tecnologia que precisa em termos de software?	83%	0%	17%	**100%**
Caso a tecnologia de software não esteja disponível, ela pode ser adquirida ou desenvolvida?	83%	0%	17%	**100%**

A organização dispõe de mão de obra para o desenvolvimento e operação do software em termos de conhecimento e domínio da tecnologia?	33%	17%	50%	**100%**
Caso não disponha de mão de obra, para o desenvolvimento e operação é possível adquiri-la por meio de treinamento?	100%	0%	0%	**100%**
Caso não disponha de mão de obra, para o desenvolvimento e operação é possível adquiri-la por meio de contratação de pessoal?	50%	0%	50%	**100%**
Uma equipe especializada ajudaria no desenvolvimento de Software Científico?	83%	0%	17%	**100%**

Quadro 4.3 Reutilização de artefatos científicos e compartilhamento de Software Científico.

Reutilização de artefatos científicos e compartilhamento de Software Científico	Resultado			
	Sim	Não	Às vezes	Total
A reutilização de artefatos científicos é realizada formalmente?	50%	17%	33%	**100%**
A reutilização de artefatos científicos é realizada informalmente?	50%	17%	33%	**100%**
A reutilização ocorre através de documentos?	17%	50%	33%	**100%**
A reutilização ocorre através de projetos?	67%	17%	17%	**100%**
A reutilização ocorre através do código?	67%	0%	33%	**100%**
Colocaria seus programas em um repositório público?	67%	0%	33%	**100%**

d) Possibilidade de Adoção de Práticas de Engenharia de Software

Neste item foram levantadas práticas de Engenharia de Software (quadro 4.4), práticas sobre o desenvolvimento do Software Científico (quadro 4.5), modelos de processo de desenvolvimento de software (quadro 4.6) e identificação de profissionais envolvidos no projeto de desenvolvimento (quadro 4.7).

Quadro 4.4 Práticas de Engenharia de Software.

Práticas de Engenharia de Software	Resultado					
	Sim	Não	Às vezes	Não conheço este termo	Gostaria de adotar	Total
Análise de riscos.	0%	50%	17%	33%	0%	**100%**
Auditorias.	0%	50%	17%	33%	0%	**100%**
Avaliação pelo cliente.	17%	33%	33%	17%	0%	**100%**
Ferramentas CASE.	0%	17%	33%	50%	0%	**100%**
Codificação.	17%	17%	17%	33%	17%	**100%**
Documentação.	0%	17%	67%	17%	0%	**100%**
Especificação do software.	0%	33%	50%	17%	0%	**100%**
Gerência de configuração.	17%	17%	0%	50%	17%	**100%**
Gerencia do ambiente.	17%	17%	0%	50%	17%	**100%**
Gerencia do projeto.	17%	17%	17%	33%	17%	**100%**
Gerenciamento e controle de mudanças.	0%	17%	33%	33%	17%	**100%**
Implantação.	17%	50%	0%	17%	17%	**100%**
Levantamento de requisitos.	33%	33%	0%	17%	17%	**100%**
Manutenção.	17%	17%	17%	33%	17%	**100%**
Medições da qualidade (Métricas).	33%	0%	17%	33%	17%	**100%**
Métodos estruturados.	33%	17%	17%	17%	17%	**100%**
Métodos orientados a objetos.	0%	33%	33%	17%	17%	**100%**

Modelo de Software Visual (UML).	17%	0%	17%	50%	17%	**100%**
Projeto da arquitetura do software.	0%	17%	17%	33%	33%	**100%**
Projeto do software.	0%	17%	33%	17%	33%	**100%**
Projeto interface com o usuário.	0%	17%	67%	17%	0%	**100%**
Prototipação.	50%	0%	0%	50%	0%	**100%**
Reuso.	17%	0%	33%	50%	0%	**100%**
Teste do sistema.	50%	17%	0%	33%	0%	**100%**
Testes de integração.	17%	33%	0%	33%	17%	**100%**
Testes de unidade.	17%	17%	0%	50%	17%	**100%**
Treinamento.	17%	17%	50%	17%	0%	**100%**
Validação.	83%	0%	0%	17%	0%	**100%**
Verificação contínua da qualidade do Software.	17%	0%	33%	33%	17%	**100%**

Quadro 4.5 Processo de Desenvolvimento de Software Científico.

Práticas sobre o desenvolvimento do Software Científico	Resultado		
	VERDADEIRO	FALSO	Total
O processo de desenvolvimento de software é caótico.	50%	50%	**100%**
A imprevisibilidade e incertezas fazem parte desse processo.	50%	50%	**100%**
Requisitos podem sempre sofrer alterações.	100%	0%	**100%**
Pressões de tempo sempre existem.	83%	17%	**100%**
Qualidade não é uma preocupação do grupo.	17%	83%	**100%**
Os recursos humanos são limitados.	100%	0%	**100%**
Resolução do problema é a prioridade.	100%	0%	**100%**
Atualização das ferramentas de desenvolvimento não é uma prioridade.	67%	33%	**100%**
Software funcionando é mais importante que documentação completa.	83%	17%	**100%**
Colaboração é muito importante.	100%	0%	**100%**
Adaptação às mudanças é mais importante que seguir um plano.	83%	17%	**100%**

Quadro 4.6 Modelos de Processo de Desenvolvimento de Software.

Modelos de processo de desenvolvimento de Software	Resultado
Processo de desenvolvimento clássico.	0%
Processo de desenvolvimento evolutivo.	20%
Processo de desenvolvimento de prototipagem.	0%
Processo de desenvolvimento ágil.	0%
Processo de desenvolvimento interativo e incremental.	80%
Total	**100%**

Quadro 4.7 Tipos de Profissionais envolvidos no projeto.

Profissionais envolvidos no projeto de desenvolvimento	Resultado			
	Sim	Não	Gostaria de ter na equipe	Total
Programador.	83%	0%	17%	**100%**
Desenvolvedor.	83%	17%	0%	**100%**
Gerente.	17%	50%	33%	**100%**
Testador de software.	50%	17%	33%	**100%**
Avaliador da qualidade.	50%	17%	33%	**100%**
Web designer.	0%	67%	33%	**100%**

Projetista de Software.	0%	50%	50%	**100%**
Engenheiro de Software.	17%	50%	33%	**100%**
Projetista de Arquitetura de Software.	17%	50%	33%	**100%**

e) Características do *Workflow* Científico

Neste item foram identificadas as características importantes de workflows científicos conforme quadro 4.8.

Quadro 4.8 Características dos *Workflows* Científicos.

Características dos *Workflows* Científicos	Resultado		
	Sim	Não	Total
Reprodutividade das análises e processos.	100%	0%	**100%**
Captura e geração de proveniência de dados.	100%	0%	**100%**
Soluções seguras, confiáveis e escaláveis.	100%	0%	**100%**
Tratar heterogeneidade e inconsistências de informações.	100%	0%	**100%**
Prover assistência inteligente.	75%	25%	**100%**
Prover diferentes formas de representação.	75%	25%	**100%**
Permitir a representação em diferentes níveis de abstração.	75%	25%	**100%**
Possuir representações para etapas de descoberta, junção, descrição e execução do workflow.	50%	50%	**100%**
Prover suporte para automação da criação e gerenciamento de processos.	50%	50%	**100%**
Permitir reexecução de processos.	100%	0%	**100%**
Aceitar variedade e heterogeneidade de dados.	100%	0%	**100%**
Apoiar colaboração e compartilhamento de práticas.	100%	0%	**100%**
Permitir a criação de workflows dinâmicos.	80%	20%	**100%**
Permitir especificação da qualidade dos serviços solicitados.	75%	25%	**100%**
Assegurar a escalabilidade da execução.	100%	0%	**100%**
Possuir infraestrutura para gerenciamento do workflow.	100%	0%	**100%**
Permitir a reutilização de parte ou todo workflow.	100%	0%	**100%**
Ser tolerante a falhas.	80%	20%	**100%**

4.2 ANÁLISE DOS RESULTADOS

Os resultados obtidos com o estudo exploratório sobre a caracterização do Software Científico com seis pesquisadores do LNCC, envolvidos em pesquisa em Modelagem Computacional, revelaram um grupo diversificado em sua formação acadêmica, porém todos envolvidos com pesquisa em Modelagem de Escoamentos em Meios Porosos.

O grupo é majoritariamente composto por pesquisadores do sexo masculino (cinco participantes) e apenas uma do sexo feminino. Um participante é da área de Computação Científica, três da Modelagem Computacional, um de Métodos Numéricos e um de Geo-mecânica Computacional, o que identifica o grupo como não desenvolvedor de software, e

com pouca formação em disciplinas específicas em desenvolvimento de software.

Como um grupo heterogêneo de cientistas, considerando nenhuma formação formal em desenvolvimento de software, o grupo retrata o que consta na literatura: o software constitui um meio para a resolução de um problema e não um fim, confirmando a característica explorativa do Software Científico.

Em relação às características de qualidade de Software Científico o grupo pesquisado identificou os seguintes atributos (quadro 4.9) como mais importantes. Neste quadro foram considerados os resultados com pelo menos 50% de imprescindível, 67% desejável e nenhum sem importância.

Quadro 4.9 Características da Qualidade dos Softwares Científicos.

Características da Qualidade dos Softwares Científicos
Funções adequadas às tarefas especificadas e aos objetivos dos usuários.
Resultados sempre corretos.
O software não apresentar falhas.
O software apresentar recursos necessários para o seu funcionamento.
Ser fácil modificar e adaptar o software.
O software ser fácil de operar e controlar.
Ser fácil testar o software quando se fizer alterações.
O software estar de acordo com padrões de portabilidade.
O software estar de acordo com as normas, padrões e leis.
Ser fácil encontrar e diagnosticar uma falha.
O software interagir com outros sistemas.

É importante observar que, em relação à ISO 9126, os pesquisadores identificaram como importantes características relacionadas à funcionalidade do software, usabilidade, eficiência, manutenibilidade, testabilidade e portabilidade e subcaracterísticas como acurácia, corretude e aderência aos padrões e normas.

Para o grupo pesquisado o controle de acesso e segurança de dados não é prioritário e a recuperação de falhas e a eficiência em relação ao tempo de resposta não são imprescindíveis. Em termos de interface a usabilidade (interface amigável e facilidade de uso) não é uma prioridade. A interoperabilidade é relevante, porém não imprescindível. Esses mesmos dados tinham sido anteriormente identificados no trabalho de Souza (2009).

Em termos de processo de desenvolvimento foi identificado por todos os participantes da pesquisa que caso não disponha de mão de obra, para o desenvolvimento e operação do software, é possível adquiri-la por meio de treinamento. A grande maioria dos participantes acredita que existe a tecnologia necessária para desenvolver o software especificado, e também concordam que a equipe encarregada pelo desenvolvimento dispõe da tecnologia que precisa em termos de software, e por fim, acreditam também que caso a

tecnologia de software não esteja disponível, ela pode ser adquirida ou desenvolvida. Não há consenso, entretanto, sobre a existência de um processo padronizado para o desenvolvimento do Software Científico. A metade dos participantes acredita que às vezes a organização dispõe de mão de obra para o desenvolvimento e operação do software em termos de conhecimento e domínio da tecnologia, entretanto é possível contratar pessoal (50% sim e 50% às vezes) caso não disponha de mão de obra para o desenvolvimento e operação.

Em relação ao processo de desenvolvimento de software, o grupo retrata o que está relatado na literatura em termos de adoção de métodos ágeis em detrimento de métodos mais específicos. Todos os participantes descrevem que adotam processo de desenvolvimento evolutivo, interativo e incremental. Isso fica claro ao caracterizarem o processo em termos de requisitos que podem sofrer alterações, foco na resolução do problema, prioridade e colaboração como sendo muito importantes, sabendo que existem maior complexidade e incerteza do domínio e que software funcionando é mais importante que documentação.

Quanto à reutilização de artefatos científicos o grupo se mostrou dividido. A maioria acredita que o Software Científico é reutilizado tanto formalmente quanto informalmente. Para a maioria essa reutilização ocorre através de projetos e código. A metade respondeu que esse reuso não se dá através de documentos. Para a maioria absoluta dos pesquisadores os mesmos colocariam seus programas em um repositório público de compartilhamento de artefatos científicos, mas 33% dos pesquisadores não colocariam seus programas em repositório público.

Em relação às práticas de Engenharia de Software os pesquisadores identificaram como possíveis de serem adotadas a prototipação, o teste do sistema e a validação. As seguintes práticas não seriam adotadas pela maioria: análise de riscos, auditorias e implantação. As práticas documentação, especificação do software, projeto de interface com o usuário e treinamentos poderiam ser adotados por pelo menos metade do grupo.

Pela análise dos resultados há evidências que o grupo desconhece boa parte das práticas da Engenharia de Software que compõem as disciplinas dos modelos mais citados na literatura como RUP (*Rational Unified Process)*, XP (*Extreme Programming*) e na pesquisa da Sepin/MCT (http://www.mct.gov.br).

Em se tratando ainda das práticas de desenvolvimento de Software Científico todos os participantes concordam que requisitos podem sofrer alterações, os recursos humanos são limitados, a resolução do problema é a prioridade e a colaboração de todos é muito importante. Entretanto, a opinião se divide quando se diz que o processo de desenvolvimento de software é caótico e também quanto à imprevisibilidade e incertezas fazerem parte desse processo. Observou-se também, que a maioria dos cientistas concorda

que pressões de tempo existem, a atualização das ferramentas de desenvolvimento não é uma prioridade, o software funcionando é mais importante que a documentação completa e que a adaptação às mudanças é mais importante que seguir um plano.

Há indícios que o grupo desconhece práticas de Engenharia de Software que poderiam auxiliar no processo de desenvolvimento de Software Científico, considerando que 83% não conhecem análise de risco; 83% não adotam as auditorias como forma de revisão e validação do software; 66% não adotam ou somente às vezes fazem validação pelo cliente; 50% não conhecem o termo Ferramenta Case; 67% só fazem documentação às vezes; 33% não fazem especificação de software e 50% às vezes fazem esta especificação. Quanto à gerência de configuração 50% não conhecem o termo e apenas 17% adotam gerência de projeto. A metade dos pesquisados não fazem a etapa de implantação do software. Apenas 33% dos entrevistados fazem levantamento de requisitos. Somente 17% dos entrevistados fazem manutenção do software e 33% não conhecem este termo. Só 33% conhecem medições de qualidade e 17% adotam às vezes porém, 33% não conhecem este termo; 33% não conhecem métodos estruturados; 66% não adotam ou adotam as vezes métodos orientados a objetos; 50% não conhecem o termo modelo de software visual, hoje representado pela UML; 33% não conhecem o termo projeto de arquitetura de software porém, 33% gostariam de adotar; 33% fazem projeto de software e 33% gostariam de adotar; 67% fazem projeto de interface com usuário as vezes; 50% conhecem prototipação e 50% não conhecem este termo; 50% desconhecem o termo reuso de software. Em termos de Software Científico 50% fazem teste do sistema; 17% fazem teste de integração e de unidade e 50% desconhecem este termo. Entretanto, 83% afirmaram fazer validação.

É claro para o grupo que o desenvolvimento de Software Científico é diferente de um software comercial, talvez isso justifique o foco no problema, entretanto o grupo pesquisado aceitaria a inclusão de uma equipe especializada, influenciando diretamente na qualidade do software e na adoção de práticas de Engenharia de Software. Para a maioria dos profissionais pesquisados, o processo de desenvolvimento atualmente adotado não atende ao desenvolvimento de Software Científico.

Em relação aos modelos de processo de desenvolvimento de software que descrevem a forma de trabalho, um dos participantes não foi considerado na pesquisa por não conhecer os termos utilizados. Dos outros, 80% considera o processo de desenvolvimento interativo e incremental e 20% avalia o processo de desenvolvimento como evolutivo.

Quanto aos tipos de profissionais que poderiam participar no projeto de desenvolvimento, é interessante observar que 83% dos pesquisados ressaltaram que poderiam incluir programadores e desenvolvedores no projeto de desenvolvimento de software; 50% gostariam de ter um projetista de software na equipe e 50% gostariam de ter

um avaliador da qualidade e o mesmo percentual gostariam de ter um web designer; 50% não incluiriam um gerente na equipe, um projetista de software, um engenheiro de software e um projetista de arquitetura de software. Há indícios de que a equipe não reconhece a importância de especialistas na área de Engenharia de Software na equipe de desenvolvimento de Software Científico.

Em relação às características importantes de *Workflows* Científicos foram considerados em termos de resultado somente cinco participantes, uma vez que um deles relatou não conhecer os termos aplicados. O grupo pesquisado identificou os seguintes atributos como importantes (quadro 4.10) em termos das características de *workflow* científico. Neste quadro foram considerados os resultados com 100% de seleção.

Quadro 4.10 Características importantes dos *Workflows* Científicos.

Características importantes de Workflows Científicos
Reprodutividade das análises e processos.
Captura e geração de proveniência de dados.
Soluções seguras, confiáveis e escaláveis.
Tratar heterogeneidade e inconsistências de informações.
Permitir reexecução de processos.
Aceitar variedade e heterogeneidade de dados.
Apoiar colaboração e compartilhamento de práticas.
Assegurar a escalabilidade da execução.
Possuir infraestrutura para gerenciamento do workflow.
Permitir a reutilização de parte ou todo workflow.

Houve uma grande unanimidade em ressaltar algumas características citadas na literatura.

4.3 RECOMENDAÇÕES

Apesar da pesquisa ser exploratória e ter focado num grupo com perfil em Modelagem Computacional, é possível inferir que os resultados obtidos, isto é, as práticas e prioridades do grupo, refletem as características identificadas na literatura para a caracterização do Software Científico e seu processo de desenvolvimento.

A seguir propomos algumas diretrizes para o grupo de pesquisadores que participaram dessa pesquisa, visando contribuir para a melhoria do processo de desenvolvimento de Software Científico pelo grupo e a definição de características de qualidade de Software Científico que norteiem o seu processo de construção e

características dos *workflows* científicos:

- ✓ Quanto às características de qualidade do Software Científico:
  - o Como não se trata de um grupo de desenvolvedores de software o desenvolvimento informal gera um caráter aleatório ao produto final do software, o que dificulta a avaliação da sua qualidade e aderência a padrões.
  - o O grupo necessita identificar como requisitos as características de qualidade do Software Científico a ser desenvolvido e considerar os atributos relacionados à recuperação de falhas, que estão diretamente relacionados a ampliação dos testes, e a usabilidade, fatores que podem trazer um diferencial para que o mesmo possa ser usado por outros pesquisadores, o que melhoraria a interoperabilidade do mesmo e as possibilidades de seu reuso.
- ✓ Identificação dos modelos de processos adotados no desenvolvimento do Software Científico:
  - o Como os Softwares Científicos constituem um meio e não um fim, a adoção rigorosa de modelos e processos de desenvolvimento pode dificultar de alguma forma a produção de Softwares Científicos. Muitas formalidades de documentação podem cercear a criatividade e o foco da resolução do problema pelo grupo.
  - o O processo pelo qual o Software Científico é desenvolvido pode ser aperfeiçoado tanto pela aplicação de metodologias de engenharia, como pelo desenvolvimento de novas metodologias específicas ao domínio de desenvolvimento de Software Científico. O grupo deveria definir um modelo de processo mínimo, com os artefatos a serem gerados, pontos de controle e etapas que garantisse a unicidade do processo entre os participantes e o acompanhamento gerencial do processo.
  - o Deveria ser estudada a possibilidade da adoção de métodos mais ágeis pelo fato de os mesmos serem receptivos a mudanças e lidarem melhor com requisitos emergenciais no desenvolvimento dos softwares.
- ✓ Possibilidade de adoção de práticas da Engenharia de Software:
  - o Há indícios de que o grupo adota ou desconhece práticas de Engenharia de Software conforme relato na literatura que podem

contribuir para a melhoria da qualidade do Software Científico e do seu processo de desenvolvimento.

- O grupo deveria considerar a adoção de inspeções, técnica de avaliação eficiente, que poderia contribuir para a melhoria da qualidade do Software Científico.
- Apesar das dificuldades de se testar um Software Científico, falta de padrões e o grande número de testes necessários quando se segue qualquer técnica de teste padrão, o grupo deveria adotar modelos de testes relatados na literatura de Engenharia de Software para que a informalidade ficasse menos presente. A confiabilidade, a eficiência, a corretude, o número de falhas durante a execução e outras características podem melhorar as medidas com a aplicação formal de um plano de teste.
- Avaliar a possibilidade de desenvolvimento de Software Científico por equipe especializada, que não os próprios cientistas.
- Apesar de ser condição altamente desejável o envolvimento dos cientistas, que têm entendimento profundo do que é necessário que o software execute, o processo de construção deveria ser compartilhado com desenvolvedores especialistas para que os cientistas se dedicassem mais aos requisitos do domínio e solução do problema e não com cumprimento de prazos e adoção de tecnologias inovadoras e mais eficientes.
- É importante que esta equipe seja qualificada academicamente e que possua entendimento da realidade e dos fatores envolvidos no desenvolvimento de Software Científico.

✓ Reutilização de artefatos científicos e compartilhamento de Software Científico:

- Hoje em dia, quase toda pesquisa poder ser feita via Internet e outros tipos de rede, portanto é fundamental que os cientistas desenvolvam os seus artefatos usando técnicas que garantam o seu compartilhamento em repositórios.

✓ Identificação de características dos *Workflows* Científicos:

- O grupo deveria adotar a criação de *workflows* dinâmicos, permitir a especificação da qualidade dos serviços solicitados, permitir também a representação em diferentes níveis de abstração e prover diferentes formas de representação.

- Possuir representações para as etapas de descoberta, junção, descrição e execução do *workflow*.
- Avaliar a possibilidade de suporte para automação da criação e gerenciamento de processos e prover a assistência de sistemas de gerenciamento.

5 CONSIDERAÇÕES FINAIS

A computação científica tem crescido rapidamente através do uso de tecnologias avançadas e resolução de problemas complexos e ela completa os pilares tradicionais da Ciência: teoria e experimentação. A computação científica também unifica três elementos distintos: modelagem e simulações, Software Científico e Ciência da Computação.

O desenvolvimento de Software Científico vem recebendo uma atenção especial dos pesquisadores desde 2008 quando a IEEE publicou duas edições especiais sobre o tema[2]. Os trabalhos recentes, na sua maioria, buscam identificar problemas na área através de estudos de caso. Esta pesquisa se propôs a contribuir com o processo de desenvolvimento de Software Científico de natureza acadêmica, com a intenção de definir para o Software Científico um padrão de qualidade similar ao software convencional. O estudo exploratório buscou caracterizar o Software Científico identificando quais características determinam a qualidade desejada para esse tipo de software, analisar e apontar características gerais que venham contribuir para a definição de padrões sobre qualidade para Softwares Científicos. Foram identificadas as práticas da Engenharia de Software que os profissionais pesquisados adotam ou gostariam de adotar para o desenvolvimento de Software Científico com o objetivo de contribuir para a definição das pesquisas e modelos para o desenvolvimento de software nessa área. Também se caracterizou *workflows* científico com o objetivo de consolidar questões de pesquisas identificadas na literatura.

O objetivo proposto no início do trabalho foi alcançado. A primeira meta pretendia revisar os conceitos, metodologias, métodos e ferramentas que permitiram caracterizar o Software Científico e também as características, reutilização, taxonomia e caracterização de *Workflow* Científico. Após essa revisão da literatura foi desenvolvido um questionário cujo objetivo é de conferir hipóteses identificadas na literatura em relação às características de um Software Científico. O instrumento foi aplicado a um grupo de pesquisadores do Laboratório Nacional de Computação Científica – LNCC.

A partir do Estudo de Caso foram quantificados e analisados os resultados, identificando as características de qualidade do Software Científico; os modelos de processos adotados no desenvolvimento do Software Científico; a possibilidade de adoção de práticas de Engenharia de Software; a possibilidade de desenvolvimento de Software Científico por equipe especializada, que não os próprios cientistas; as características dos

[2] IEEE Software, volume 25, issue 4, July 2008
IEEE computer, volume 41, number 11, November 2008

workflows científicos; a reutilização de artefatos científicos e o compartilhamento de Software Científico.

O domínio da aplicação, os objetivos do projeto, o tamanho, a complexidade e a cultura do grupo de pesquisadores têm uma grande influência na escolha do processo de desenvolvimento do Software Científico. Alguns podem ser baseados numa abordagem mais clássica, voltada para planos, mas há hoje tendência para a adoção de modelos mais ágeis, que retratam melhor o foco no problema. Esses modelos têm capacidade de absorver mudanças e lidar com requisitos emergenciais, características presentes no desenvolvimento do Software Científico além de privilegiar a exploração, a interatividade e a colaboração. Permitem um ciclo iterativo e incremental, características destacadas pelos pesquisadores. O que não deve ocorrer é não se adotar nem um processo padrão mínimo que configure as atividades, documentos e testes a serem seguidos pelo grupo de cientistas.

A maioria dos cientistas tem uma formação informal sobre desenvolvimento de software, esse cenário contribui para o desconhecimento e a pouca adoção de práticas de Engenharia de Software pelos cientistas. Isso se reflete na baixa qualidade dos produtos, na falta de padronização de testes, documentação insuficiente e baixa usabilidade. A adoção de práticas como gerenciamento de configuração e projeto arquitetural poderiam contribuir para a ampliação da qualidade, da produtividade e do reuso, diminuindo o retrabalho, aumentando o compartilhamento e a aceleração das soluções científicas.

. Os workflows emergiram como um paradigma para representação de computação complexa e distribuída e para ajudar a acelerar o progresso científico. Dessa forma, apesar do foco dessa dissertação não ser em workflow científico, a caracterização dos sistemas de gerenciamento de workflow científicos contribuiu para a identificação de abordagens práticas para estudos futuros sobre ferramentas a serem desenvolvidas para a área. Há evidências que os cientistas pesquisados concordam que os workflows fornecem uma representação formal das computações complexas e que as etapas de execução e compartilhamento são importantes. O reuso do conhecimento, através da disponibilização de artefatos foi evidenciado como de grande possibilidade pelos cientistas pesquisados, o que pode contribuir para a criação e a eficiência do workflow. Avaliações mais sistemáticas sobre todo o ciclo de vida dos workflows científicos necessitam ser conduzidas. Pelos dados levantados os cientistas pesquisados não tem conhecimentos suficientes sobre projeto, execução e compartilhamento de workflows e desconhecem as ferramentas que podem auxiliar os cientistas não especialistas em programação.

A questão dos direitos autorais dos códigos e artefatos gerados parece ser uma preocupação dos cientistas pesquisados e, de certa forma, isso restringe o uso de soluções já existentes para repositórios de serviços Web e soluções em Grid.

Essa dissertação buscou não só destacar as características do estado da arte em desenvolvimento de Software Científico e de workflows científicos, mas, também, identificar as áreas que necessitam mais pesquisas. Para estudos futuros será necessária a aplicação do instrumento desenvolvido em grupos diversificados de cientistas, representantes de outras áreas da computação científica. Esses resultados poderão fornecer indícios de como melhorar o Software Científico, seu processo de desenvolvimento, seu compartilhamento e reuso em ambientes distribuídos e cooperativos e apontar questões de pesquisas para *workflows* científicos.

REFERÊNCIAS

BELCHIOR, Arnaldo Dias. **Um Modelo Fuzzy para Avaliação da Qualidade de Software.** 1997. Tese (Doutorado em Ciências em Engenharia de Sistemas e Computação)-Universidade Federal do Rio de Janeiro, Rio de Janeiro, 1997. Disponível em:<www.lbd.dcc.ufmg.br:8080/colecoes/sbqs/2002/005.pdf>. Acesso em: 27 set. 2010.

BERNI, Jean C. A. **Gestão para o Processamento de Desenvolvimento de Software Científico utilizando uma Abordagem Ágil e Adaptativa na Microempresa**. 2010. Dissertação (Mestrado em Engenharia de Produção)-Universidade Federal de Santa Maria, Santa Maria, RS, 2010. Disponível em: <http://cascavel.cpd.ufsm.br/tede/tde_busca/arquivo.php?codArquivo=3060>. Acesso em: 27 nov. 2010.

BRASIL, Ministério da Ciência e Tecnologia. Secretaria de Política de Informática. Disponível em http://www.mct.gov.br. Acesso em 09 dez. 2010.

CRABTREE, Carlton A; *et al*. **An Empirical Characterization of Scientific Software Development Projects According to the Boehm and Turner Model: a Progress Report**. IEEE Computer Society, SECSE '09: Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering, 2009. Disponível em: http://portal.acm.org/results.cfm?h=1&cfid=10261596&cftoken=75219034. Acesso em: 21 jul. 2009.

GASPAR, Wander; *et al*. **SW-ONTOLOGY -** A Proposal for Semantics Modeling of a Scientific Workflow Management System. ICEIS 2010 (International Conference on Enterprise Information Systems), 2010.

GIL, Yolanda; *et. al*. **Examining the Challenges of Scientific *Workflows*.** IEEE Computer, v. 40, n. 12, 2007. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4404805>. Acesso em: 19 ago. 2010.

HANNAY, Jo Erskine; *et al*. **How Do Scientists Develop and Use Scientific Software?** IEEE Computer Society, SECSE '09: Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering, 2009. Disponível em: <http://delivery.acm.org/10.1145/1560000/1556928/05069155.pdf?key1=1556928&key2=0585044921&coll=DL&dl=ACM&CFID=4749879&CFTOKEN=11464392>. Acesso em: 21 jul. 2009.

HEROUX, Michael A.**; WILLENBRING**, James M. **Barely Sufficient Software Engineering:** 10 Practices to Improve Your CSE Software. IEEE Computer Society, SECSE '09: Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering, 2009. Disponível em: <http://delivery.acm.org/10.1145/1560000/1556930/05069157.pdf?key1=1556930&key2=6926044921&coll=DL&dl=ACM&CFID=4749879&CFTOKEN=11464392>. Acesso em: 21 jul. 2009.

ISO/IEC 9126-1. Qualidade de Software. 2003. Disponível em: < http://www.pdf-freedownload.com/pdf-folder/abnt-nbr-isoiec-9126-2-pdf.php>. Acesso em: 18 jul. 2009.

HOOK, Daniel; **KELLY**, Diane. **Testing for Trustworthiness in Scientific Software.** IEEE Computer Society, SECSE '09: Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering, 2009. Disponível em: <http://delivery.acm.org/10.1145/1560000/1556930/05069157.pdf?key1=1556930&key2=6926044921&coll=DL&dl=ACM&CFID=4749879&CFTOKEN=11464392>. Acesso em: 21 jul. 2009.

IEEE, **1990**, **Standard Glossary of Software Engineering Terminology (ANSI)**. ISBN 1-55937-067-X. Disponível em: <http://www.idi.ntnu.no/grupper/su/publ/ese/ieee-se-glossary-610.12-1990.pdf>. Acesso em: 10 ago. 2010.

ISO, **1991**, ISO/IEC 9126, ***Information Technology - Software Product Evaluation*, Quality characteristics and guidelines for their use.** Disponível em: < http://www.pdf-freedownload.com/pdf-folder/abnt-nbr-isoiec-9126-2-pdf.php>. Acesso em: 18 jul. 2009.

ISO, **1994**, ISO DIS 8402, ***Quality Vocabulary.*** Disponível em: < http://www.pdf-freedownload.com/pdf-folder/abnt-nbr-isoiec-9126-2-pdf.php>. Acesso em: 18 jul. 2009.

LANGGUTH, Christoph; **RANALDI**, Paola; **SCHULDT**, Heiko. **Improving the Reuse of Scientific *Workflows* and Their By-products.** University of Basel, Switzerland, 2009. Disponível em: <http://doi.ieeecomputersociety.org/10.1109/MC.2007.421>. Acesso em: 19 ago. 2010.

MAXVILLE, Valerie. **Preparing Scientists for Scalable Software Development**. . IEEE Computer Society, SECSE '09: Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering, 2009. Disponível em: <http://portal.acm.org/citation.cfm?id=1556904.1556939&coll=DL&dl=GUIDE&CFID=4749879&CFTOKEN=11464392>. Acesso em: 21 jul. 2009.

MOURA, Marcelle C.; **PURRI**, Souza. **Estudo e Propostas Iniciais para a Definição de um Processo de Desenvolvimento para Software Científico**. 2006. Dissertação (Mestrado em Modelagem Matemática e Computacional) - Centro Federal de Educação Tecnológica de Minas Gerais – Belo Horizonte, MG, 2006. Disponível em: <http://www.mmc.cefetmg.br/index.php?codigo=30>. Acesso em: 05 out. 2010.

NARDI, A. R. **Uma arquitetura de baixo acoplamento para execução de padrões de controle de fluxo em grades**. 2009. Tese (Doutorado em Ciência da Computação) - Instituto de Matemática e Estatística da Universidade de São Paulo, São Paulo, 2009. Disponível em: <http://www.saocamilo-sp.br/biblioteca/oai/index.php?word=%20control-flow>. Acesso em: 28 out. 2010.

NGUYEN-HOAN, Luke. **Improving Scientific Software Development.** ANU - Australian National University, 2009. Disponível em: http://cecs.anu.edu.au/files/posters09/41133440000000115.pdf. Acesso em: 05 jul. 2010.

NGUYEN-HOAN, Luke; **FLINT**, Shayne; **SANKARANARAYANA**, Ramesh. **A Survey of Scientific Software Development Proceeding.** ESEM '10 Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement ISBN:

978-1-4503-0039-1.

PALAZZI, Daniele C. **QDAontology – Abordagem para o Desenvolvimento de Ontologias em E-Science:** Um Estudo de Caso em Biologia. 2010. Dissertação (Mestrado em Modelagem Computacional) - Universidade Federal de Juiz de Fora, Juiz de Fora, MG, 2010. Disponível em: <http://www.ufjf.br/mmc/files/2010/06/dissertacao>. Acesso em: 18 jul. 2010.

PRESSMAN, Roger. **Engenharia de Software. 6 ed. São Paulo:** McGraw-Hill, 2006.

ROCHA, Ana Regina Cavalcanti da, *et* al. **Qualidade de Software. Teoria e Prática. São Paulo:** Prentice Hall, 2001.

SEGAL, Judith. **Scientists and Software engineers: a tale of two cultures**. PPIG Psycology of Programming Interest Group, Lancaster, 2008. Disponível em: <http://www.ppig.org>. Acesso em: 05 jul. 2010.

SEGAL, Judith. **Models of Scientific Software Development.** University Milton Keynes, 2004. Disponível em: <http://cs.ua.edu/~SECSE08/Papers/Segal.pdf >. Acesso em: 30 jun. 2010.

SEGAL, Judith; **MORRIS**, Chris. **Developing Scientific Software.** IEEE Software, v. 25, n. 4, 2008. Disponível em: < http://doi.ieeecomputersociety.org/10.1109/MS.2008.85>. Acesso em: 05 jul. 2010.

SEGAL, Judith; **MORRIS**, Chris. **Developing Scientific Software, Part 2.** IEEE Software, v. 26, n. 1, 2009. Disponível em:<http://doi.ieeecomputersociety.org/10.1109/MS.2009.8>. Acesso em: 05 jul. 2010.

SEGAL, Judith. **Some challenges facing software engineers developing software for scientists**. SECSE '09 Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering, 2009. ISBN: 978-1-4244-3737-5. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5069156>. Acesso em: 21 jul. 2009.

SILVA, Laryssa A. Machado da. **Composer-Science:** Um Framework para a Composição de *Workflows* Científicos**.** 2010. Dissertação (Mestrado em Modelagem Computacional) - Universidade Federal de Juiz de Fora, Juiz de Fora, MG, 2010. Disponível em: <www.ufjf.br/mmc/files/2010/08/dissertacao>. Acesso em: 18 jul. 2010.

SOUZA, Nívea. **A Importância da Qualidade nos Softwares Científicos no Atendimento às Expectativas dos Cientistas.** Relatório Técnico, 2009.

XIANG, Xiaorong; **MADEY**, Gregory. **Improving the Reuse of Scientific *Workflows* and Their By-products.** IEEE International Conference on Web Services (ICWS 2007), 2007. ISBN : 0-7695-2924-0. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4279673&tag=1>. Acesso em: 19 ago. 2010.

YU, Jia; **BUYYA**, Rajkumar. **A Taxonomy of Scientific *Workflow* Systems for Grid Computing.** University of Melbourne, Austrália, 2005. Disponível em: <http://portal.acm.org/citation.cfm?id=1084805.1084814&coll=DL&dl=GUIDE&CFID=4749879&CFTOKEN=11464392>. Acesso em: 19 ago. 2010.

APÊNDICE

Apêndice 1

Pesquisa de Cientistas sobre Qualidade aos Softwares Científicos

Apoio ao Controle de Qualidade aos Softwares Científicos

Atributo	Imprescindível	Importante	Sem importância
As funções devem ser adequadas às tarefas especificadas e aos objetivos dos usuários			
Os resultados devem sempre ser corretos			
O software deve interagir com outros sistemas			
O software deve estar de acordo com as normas, padrões, leis, etc.			
O software deve possuir controle de acesso e segurança dos dados			
O software não deve apresentar falhas			
O software deve manter o nível de desempenho mesmo em caso de falhas			
O software é capaz de recuperar dados em caso de falha			
O software deve permitir o usuário entender se ele é adequado e como ele pode ser usado para tarefas e condições de uso específicas			
O software deve ser fácil de aprender a usar			
O software deve ser fácil de operar e controlar			
A interface do software deve ser atrativa			
O software deve ter tempo de resposta e velocidade de execução adequada			
O software deve necessitar de recursos adequados ao seu funcionamento			
Deve ser fácil encontrar e diagnosticar uma falha			
Deve ser fácil modificar e adaptar o software			
As alterações no software não devem trazer grandes riscos			
Deve ser fácil testar o software quando se faz alterações			
Deve ser fácil adaptar o software a outros ambientes			
Deve ser fácil instalar o software em outros ambientes			
O software deve estar de acordo com padrões de portabilidade			
Deve ser fácil substituir o software por outro com os mesmos objetivos			

Apêndice 2

Formulário de Validação da Caracterização do Software Científico

Caro pesquisador:

Estamos pesquisando a caracterização de Software Científico: um estudo em modelagem computacional. Agradecemos a sua colaboração preenchendo esse formulário.

Nívea Bellose Oliveira de Souza (mestrando)
Fernanda Cláudia Alves Campos (orientadora)
Mestrado em Modelagem Computacional

1 - Nome (opcional): _____

2 - Sexo: ( ) masculino ( ) feminino

3 - Pós-graduação:
( ) Mestrado ( ) Doutorado Área de concentração:

4. Pesquisa:
Área de concentração:

I Caracterização do Software Científico

Assinale de acordo com a escala abaixo o grau de importância da característica de qualidade do Software Científico.

Tabela I - Escala de grau de importância da característica de qualidade.

ESCALA	Opção
I	*Imprescindível*
D	*Desejável*
SI	*Sem Importância*

OBJETIVO: Identificação de características de qualidade de Software Científico	
	I D SI
1. As funções são adequadas às tarefas especificadas e aos objetivos dos usuários?	☐ ☐ ☐
2. Os resultados são sempre corretos?	☐ ☐ ☐
3. O software interage com outros sistemas?	☐ ☐ ☐
4. O software está de acordo com as normas, padrões e leis?	☐ ☐ ☐
5. O software possui controle de acesso e segurança dos dados?	☐ ☐ ☐
6. O software não apresenta falhas?	☐ ☐ ☐
7. O software mantém o nível de desempenho mesmo em caso de falhas?	☐ ☐ ☐
8. O software é capaz de recuperar dados em caso de falha?	☐ ☐ ☐
9. O software permite ao usuário entender se ele é adequado e como ele pode ser usado para tarefas e condições de uso específico?	☐ ☐ ☐
10. É fácil aprender a utilizar o software?	☐ ☐ ☐
11. O software é fácil de operar e controlar?	☐ ☐ ☐
12. A interface do software é atrativa?	☐ ☐ ☐
13. O software apresenta tempo de resposta e velocidade de execução adequada?	☐ ☐ ☐
14. O software apresenta recursos necessários para o seu funcionamento?	☐ ☐ ☐
15. É fácil encontrar e diagnosticar uma falha?	☐ ☐ ☐
16. É fácil modificar e adaptar o software?	☐ ☐ ☐
17. As alterações no software trazem grandes riscos?	☐ ☐ ☐
18. É fácil testar o software quando se faz alterações?	☐ ☐ ☐
19. É fácil adaptar o software em outros ambientes?	☐ ☐ ☐
20. É fácil instalar o software em outros ambientes?	☐ ☐ ☐
21. O software está de acordo com padrões de portabilidade?	☐ ☐ ☐
22. É fácil substituir o software por outro com os mesmos objetivos?	☐ ☐ ☐
Comentários:	

II Caracterização do Processo de Desenvolvimento de Software Científico e Reuso de Artefatos

Assinale a sua resposta de acordo com a escala abaixo.

Tabela II - Escala de grau de uso ou importância.

ESCALA	Opção
S	SIM
N	NÃO
AV	ÀS VEZES OU TALVEZ

OBJETIVO: Identificação dos modelos de processos de desenvolvimento do Software Científico	
	S N AV
1.Utiliza algum processo padronizado para o desenvolvimento de Software Científico?	☐ ☐ ☐
2. Caso utilize algum processo padronizado, ele atende às necessidades de construção de Software Científico?	☐ ☐ ☐
3. Existe a tecnologia necessária para desenvolver o software especificado?	☐ ☐ ☐
4. A equipe encarregada pelo desenvolvimento dispõe da tecnologia que precisa em termos de software?	☐ ☐ ☐
5. Caso a tecnologia de software não esteja disponível, ela pode ser adquirida ou desenvolvida?	☐ ☐ ☐
6. A organização dispõe de mão de obra para o desenvolvimento e operação do software em termos de conhecimento e domínio da tecnologia?	☐ ☐ ☐
7. Caso não disponha de mão de obra, para o desenvolvimento e operação é possível adquiri-la por meio de treinamento?	☐ ☐ ☐
8. Caso não disponha de mão de obra, para o desenvolvimento e operação é possível adquiri-la por meio de contratação de pessoal?	☐ ☐ ☐
9. Uma equipe especializada ajudaria no desenvolvimento de Software Científico?	☐ ☐ ☐
OBJETIVO: Identificação da reutilização de artefatos científicos e compartilhamento de Software Científico	
1. A reutilização de artefatos científicos é realizada formalmente?	☐ ☐ ☐
2. A reutilização de artefatos científicos é realizada informalmente?	☐ ☐ ☐
3. A reutilização ocorre através de documentos?	☐ ☐ ☐
4. A reutilização ocorre através de projetos?	☐ ☐ ☐
5. A reutilização ocorre através do código?	☐ ☐ ☐
6. Colocaria seus programas em um repositório público?	☐ ☐ ☐
Comentários:	

III Possibilidade de Adoção de Práticas de Engenharia de Software

Assinale as práticas ou etapas de Engenharia de Software que você adota no seu dia a dia de desenvolvedor de software de acordo com a escala da Tabela III abaixo e na última coluna se gostaria de adotar (caso não adote).

Tabela III - Escala de grau de importância da característica de qualidade.

ESCALA	Opção
S	Sempre
N	Nunca
AV	Às vezes
NC	Não conheço esse termo
GA	Gostaria de adotar

OBJETIVO: Identificação de práticas de Engenharia de Software					
	S	N	AV	NC	GA
1. Análise de riscos					
2. Auditorias					
3. Avaliação pelo cliente					
4. Ferramentas CASE					
5. Codificação					
6. Documentação					
7. Especificação do software					
8. Gerência de configuração					
9. Gerencia do ambiente					
10. Gerencia do projeto					
11. Gerenciamento e controle de mudanças					
12. Implantação					
13. Levantamento de requisitos					
14. Manutenção					
15. Medições da qualidade (Métricas)					
16. Métodos estruturados					
17. Métodos orientados a objetos					
18. Modelo de Software Visual (UML)					
19. Projeto da arquitetura do software					
20. Projeto do software					
21. Projeto interface com o usuário					
22. Prototipação					
23. Reuso					
24. Teste do sistema					
25. Testes de integração					
26. Testes de unidade					
27. Treinamento					
28. Validação					
29. Verificação contínua da qualidade do Software					

Assinale Verdadeiro ou Falso para as sentenças sobre o desenvolvimento do software por você ou seu grupo:

( ) O processo de desenvolvimento de software é caótico.
( ) A imprevisibilidade e incertezas fazem parte desse processo.
( ) Requisitos podem sempre sofrer alterações.
( ) Pressões de tempo sempre existem.
( ) Qualidade não é uma preocupação do grupo.
( ) Os recursos humanos são limitados
( ) Resolução do problema é a prioridade.
( ) Atualização das ferramentas de desenvolvimento não é uma prioridade.
( ) Software funcionando é mais importante que documentação completa.
( ) Colaboração é muito importante.
( ) Adaptação às mudanças é mais importante que seguir um plano.

Qual desses modelos de processo de desenvolvimento de software você acha que melhor descreve a sua forma de trabalhar:

( ) Processo de desenvolvimento clássico
( ) Processo de desenvolvimento evolutivo
( ) Processo de desenvolvimento de prototipagem
( ) Processo de desenvolvimento ágil
( ) Processo de desenvolvimento interativo e incremental

Assinale os tipos de profissionais que trabalham com você ou que você gostaria de ter na sua equipe de acordo com a escala da Tabela IV abaixo.

Tabela IV - Escala de opção de presença de profissionais na equipe de um projeto de desenvolvimento de Software Científico.

ESCALA	**Opção**
S	Sim
N	Não
GT	Gostaria de Ter na equipe

PROFISSIONAL	S	N	GT
Programador			
Desenvolvedor			
Gerente			
Testador de software			
Avaliador da qualidade			
Web designer			
Projetista de software			
Engenheiro de Software			
Projetista de Arquitetura de Software			
Outros			

IV Caracterização de *Workflow* Científico

OBJETIVO: Identificação de características dos W*o*rkflows Científicos.

Assinale Verdadeiro ou Falso de acordo com as características importantes de *workflow*s científicos:

1. ( ) Reprodutividade das análises e processos.
2. ( ) Captura e geração de proveniência de dados.
3. ( ) Soluções seguras, confiáveis e escaláveis.
4. ( ) Tratar heterogeneidade e inconsistências de informações.
5. ( ) Prover assistência inteligente.
6. ( ) Prover diferentes formas de representação.
7. ( ) Permitir a representação em diferentes níveis de abstração.
8. ( ) Possuir representações para etapas de descoberta, junção, descrição e execução do workflow.
9. ( ) Prover suporte para automação da criação e gerenciamento de processos.
10. ( ) Permitir reexecução de processos.
11. ( ) Aceitar variedade e heterogeneidade de dados.
12. ( ) Apoiar colaboração e compartilhamento de práticas.
13. ( ) Permitir a criação de workflows dinâmicos.
14. ( ) Permitir especificação da qualidade dos serviços solicitados.
15. ( ) Assegurar a escalabilidade da execução.
16. ( ) Possuir infraestrutura para gerenciamento do workflow.
17. ( ) Permitir a reutilização de parte ou todo workflow.
18. ( ) Ser tolerante a falhas.

.