ISBN: 978-65-986235-3-1

Aurum EDITORA

CAPÍTULO 11

MODELOS DE PROCESSOS DE ENGENHARIA DE SOFTWARE

SOFTWARE ENGINEERING PROCESS MODELS

https://doi.org/10.63330/aurumpub.005-011

Ramon Santos Fernandes
Engenharia de software
UniFatecie
E-mail: rosangelaengbio@gmail.com

RESUMO

Este trabalho tem como tema central a análise dos modelos de processos de engenharia de software, com foco nas abordagens adotadas para o desenvolvimento de sistemas computacionais e nas contribuições que essas metodologias oferecem para a eficiência, qualidade e organização dos projetos. O objetivo principal é compreender como os modelos mais tradicionais e modernos de desenvolvimento auxiliam na superação dos desafios recorrentes na criação de softwares, como alterações nos requisitos, controle de prazos, custos e satisfação do usuário final. A pesquisa foi conduzida por meio de uma metodologia bibliográfica, com base em autores como Pressman, Sommerville, IBM e o IEEE, utilizando suas obras como referência para examinar os principais modelos aplicados na engenharia de software. Entre os modelos abordados estão o modelo cascata, a prototipação, o modelo espiral, o modelo incremental e o desenvolvimento baseado em componentes. Também foi incluída a análise do guia SWEBOK (Software Engineering Body of Knowledge), que sistematiza as áreas do conhecimento necessárias ao exercício da engenharia de software. Os resultados da pesquisa demonstram que, embora cada modelo possua características próprias e limitações específicas, a escolha e a aplicação adequada de um processo de desenvolvimento têm impacto direto na qualidade e no sucesso do software produzido. Verificou-se ainda que modelos mais iterativos e adaptáveis, como o espiral e o incremental, oferecem maior flexibilidade frente à complexidade dos sistemas atuais, enquanto abordagens como o modelo cascata são mais indicadas em contextos com requisitos bem definidos. Conclui-se que a engenharia de software, ao integrar técnicas sistematizadas com foco na análise de riscos, reuso de componentes e participação do usuário, representa uma ferramenta indispensável na construção de soluções tecnológicas eficazes. Além disso, destaca-se a importância da atualização constante dos profissionais da área, diante das rápidas transformações tecnológicas e da evolução das metodologias empregadas no setor.

Palavras-chave: Engenharia de Software; Modelos de Processo; Desenvolvimento de Sistemas; Qualidade de Software.

ABSTRACT

The central theme of this work is the analysis of software engineering process models, with a focus on the approaches adopted for the development of computer systems and the contributions that these methodologies make to the efficiency, quality and organization of projects. The main objective is to understand how the most traditional and modern development models help to overcome the recurring challenges in software creation, such as changes to requirements, control of deadlines, costs and end-user satisfaction. The research was conducted using a bibliographic methodology, based on authors such as Pressman, Sommerville, IBM and the IEEE, using their works as a reference to examine the main models applied in software engineering. Among the models covered are the waterfall model, prototyping, the spiral model, the incremental model and component-based development. Also included was an analysis of the SWEBOK (Software Engineering Body of Knowledge) guide, which systematizes the areas of knowledge

necessary for software engineering. The results of the research show that although each model has its own characteristics and specific limitations, the choice and proper application of a development process has a direct impact on the quality and success of the software produced. It was also found that more iterative and adaptable models, such as the spiral and incremental, offer greater flexibility in the face of the complexity of today's systems, while approaches such as the waterfall model are more suitable in contexts with well-defined requirements. It is concluded that software engineering, by integrating systematized techniques with a focus on risk analysis, component reuse and user participation, represents an indispensable tool in the construction of effective technological solutions. In addition, the importance of constantly updating professionals in the field is highlighted, given the rapid technological changes and the evolution of the methodologies used in the sector.

Keywords: Software Engineering; Process Models; Systems Development; Software Quality.

1 INTRODUÇÃO

A crescente complexidade dos sistemas computacionais e a necessidade de entregar softwares confiáveis, eficientes e dentro de prazos e orçamentos cada vez mais rígidos tornam a Engenharia de Software uma disciplina essencial no cenário atual da tecnologia. Este trabalho tem como tema central os modelos de processos de desenvolvimento de software, destacando suas contribuições para a organização, qualidade e eficácia na produção de sistemas. Fundamentado na literatura de autores como Pressman, Sommerville, IBM e no guia SWEBOK, este estudo se propõe a analisar de forma comparativa diferentes abordagens metodológicas adotadas ao longo do tempo, entre elas o modelo cascata, a prototipação, o modelo espiral, o incremental e o desenvolvimento baseado em componentes. Parte-se da hipótese de que a escolha adequada do modelo de processo impacta significativamente o sucesso do projeto, principalmente em termos de adaptabilidade a mudanças, controle de riscos, produtividade da equipe e satisfação do usuário final.

A justificativa da pesquisa reside no fato de que, mesmo com o avanço das metodologias e ferramentas de apoio, muitas organizações ainda enfrentam dificuldades na implementação eficaz de processos de desenvolvimento, o que compromete a qualidade do produto final. Assim, o estudo visa não apenas descrever os modelos, mas também evidenciar suas vantagens, limitações e contextos de aplicação mais apropriados.

É importante destacar que a Engenharia de Software não surgiu como uma disciplina técnica isolada, mas como uma resposta concreta aos problemas recorrentes enfrentados pelas indústrias de software a partir da década de 1960, quando o desenvolvimento de sistemas se tornava cada vez mais complexo e instável. Os primeiros modelos de processo, como o ciclo de vida em cascata, foram desenvolvidos para trazer uma abordagem mais formal e previsível ao desenvolvimento de software, buscando aplicar princípios da engenharia tradicional ao campo computacional. No entanto, com o passar do tempo, percebeu-se que os modelos rígidos apresentavam limitações diante da natureza dinâmica e imprevisível de muitos projetos, levando à criação de modelos mais flexíveis, iterativos e adaptáveis, como o modelo espiral, o incremental e o baseado em componentes.

A revisão bibliográfica conduzida neste trabalho possibilitou compreender que não existe um modelo de processo universalmente ideal, mas sim metodologias que devem ser escolhidas conforme as características do projeto, o perfil da equipe, o domínio do problema e o nível de envolvimento dos usuários. A literatura consultada, como a obra de Pressman (2006), destaca que muitos fracassos em projetos de software estão relacionados à escolha inadequada do modelo de desenvolvimento ou à aplicação ineficiente das boas práticas recomendadas. Sommerville (1995), por sua vez, reforça que a engenharia de software deve estar orientada por processos bem definidos, mas flexíveis o suficiente para acomodar mudanças e incertezas. O guia SWEBOK, por sua vez, sistematiza o corpo de conhecimentos essenciais da área e aponta

para a necessidade de formação interdisciplinar e contínua dos profissionais que atuam nesse campo.

Dessa forma, este trabalho também busca, como objetivo complementar, fomentar uma reflexão crítica sobre como os modelos de processo podem ser combinados ou adaptados de forma híbrida para atender melhor às demandas de projetos contemporâneos, como aqueles baseados em metodologias ágeis ou voltados para arquitetura orientada a serviços. A conclusão da pesquisa mostra que o uso consciente, estratégico e contextualizado dos modelos de engenharia de software pode reduzir falhas, aumentar a previsibilidade, melhorar a comunicação entre os envolvidos e, sobretudo, garantir a entrega de um produto que atenda às expectativas do cliente e da sociedade. Portanto, além de fornecer uma análise técnica sobre cada modelo, este trabalho contribui para o fortalecimento da capacidade de tomada de decisão em projetos de software, essencial para o sucesso em um mercado em constante transformação.

A metodologia adotada foi a pesquisa bibliográfica, com base em fontes técnicas e acadêmicas reconhecidas na área da Engenharia de Software. A estrutura do trabalho foi organizada em capítulos que seguem uma progressão lógica do conhecimento. O primeiro capítulo apresenta o conceito e a evolução histórica da Engenharia de Software, com foco em sua importância estratégica e nos princípios fundamentais da área. No segundo capítulo, são discutidas as técnicas e práticas da engenharia aplicadas ao desenvolvimento de software, evidenciando a necessidade de processos bem definidos. O terceiro capítulo detalha os principais modelos de processo: o modelo cascata, com sua estrutura linear e previsível; a prototipação, voltada à interação com o usuário e testes rápidos; o modelo espiral, que incorpora a análise de riscos em ciclos evolutivos; o incremental, que busca entregar partes funcionais do software ao longo do tempo; e o desenvolvimento baseado em componentes, que enfatiza a reutilização de módulos de software.

Por fim, o trabalho apresenta uma reflexão sobre o SWEBOK, o guia do IEEE que sistematiza os conhecimentos essenciais da área em dez áreas principais, destacando também suas limitações frente à constante evolução das práticas de desenvolvimento. Assim, este estudo contribui para a compreensão crítica dos modelos de processos em Engenharia de Software, ressaltando que a escolha da abordagem mais adequada pode representar a diferença entre o sucesso e o fracasso de um projeto de software.

2 DESENVOLVIMENTO

2.1 ENGENHARIA DE SOFTWARE

Atualmente, o software de computadores representa uma tecnologia essencial em escala global, tendo apresentado um desenvolvimento acelerado desde os anos 1950. Com esse crescimento intenso, surgiram diversos desafios, especialmente relacionados à correção de erros, adaptação, aprimoramento e, principalmente, à manutenção, que acaba demandando mais recursos humanos e financeiros do que o próprio desenvolvimento de novos sistemas. Diante dessas dificuldades, novos conceitos foram sendo formulados, dentre eles o da Engenharia de Software. Conforme definição de Fritz Bauer, Engenharia de

Software consiste na “criação e uso de princípios sólidos da engenharia com o objetivo de produzir softwares econômicos, confiáveis e eficientes em máquinas reais”. Já o IEEE apresenta uma concepção mais ampla, descrevendo-a como: “(1) a aplicação de uma abordagem sistemática, disciplinada e quantificável ao desenvolvimento, operação e manutenção do software — ou seja, a aplicação da engenharia ao software; (2) o estudo dessas abordagens”. (PRESSMAN, 2006). Essas definições estão reunidas na obra de Pressman, que enfatiza a Engenharia de Software como uma tecnologia estruturada em camadas — Ferramentas, Métodos, Processo e Foco na Qualidade —, sendo fundamental que as organizações se comprometam com a busca contínua pela qualidade.

2.1 TÉCNICAS DE ENGENHARIA DE SOFTWARE

A Engenharia de Software é reconhecida como uma abordagem eficaz para o desenvolvimento de projetos de software. No entanto, muitas de suas práticas ainda não são plenamente adotadas na produção de sistemas, o que contribui para a persistência de diversos problemas no setor. A criação de softwares complexos, capazes de atender às expectativas dos usuários e que sejam entregues dentro dos prazos e orçamentos definidos, ainda representa um grande desafio. Diante disso, alguns estudiosos, como Pressman (2006), chegaram a afirmar que a Engenharia de Software se encontra em um estado de aflição crônica. À medida que a capacidade técnica de desenvolver software avançou, também aumentou a complexidade dos sistemas demandados, especialmente com o surgimento de novas tecnologias oriundas da convergência entre sistemas computacionais e de comunicação. Essa evolução trouxe novos desafios para os engenheiros de software, que, em muitos casos, ainda enfrentam dificuldades devido à aplicação ineficiente das práticas da área pelas organizações. Apesar disso, a situação não é tão crítica quanto alguns pessimistas apontam, embora haja, de fato, muito espaço para melhorias.

Nesse contexto, destaca-se a importância dos modelos de processo de software, que consistem em representações abstratas dos processos de desenvolvimento e fornecem visões específicas e parciais sobre as atividades envolvidas (SOMMERVILLE, 1995). Entre os modelos que serão discutidos neste artigo, encontram-se: o modelo em cascata, o desenvolvimento evolucionário, o desenvolvimento iterativo (incluindo os modelos espiral e incremental) e o desenvolvimento baseado em componentes. Além desses, existem outros modelos citados por Pressman (2006), que não serão abordados com profundidade neste trabalho, tais como o modelo incremental, o modelo RAD (Desenvolvimento Rápido de Aplicações), o modelo de desenvolvimento concorrente, o modelo de métodos formais e o Processo Unificado.

2.2 CASCATA, LINEAR OU CLÁSSICO

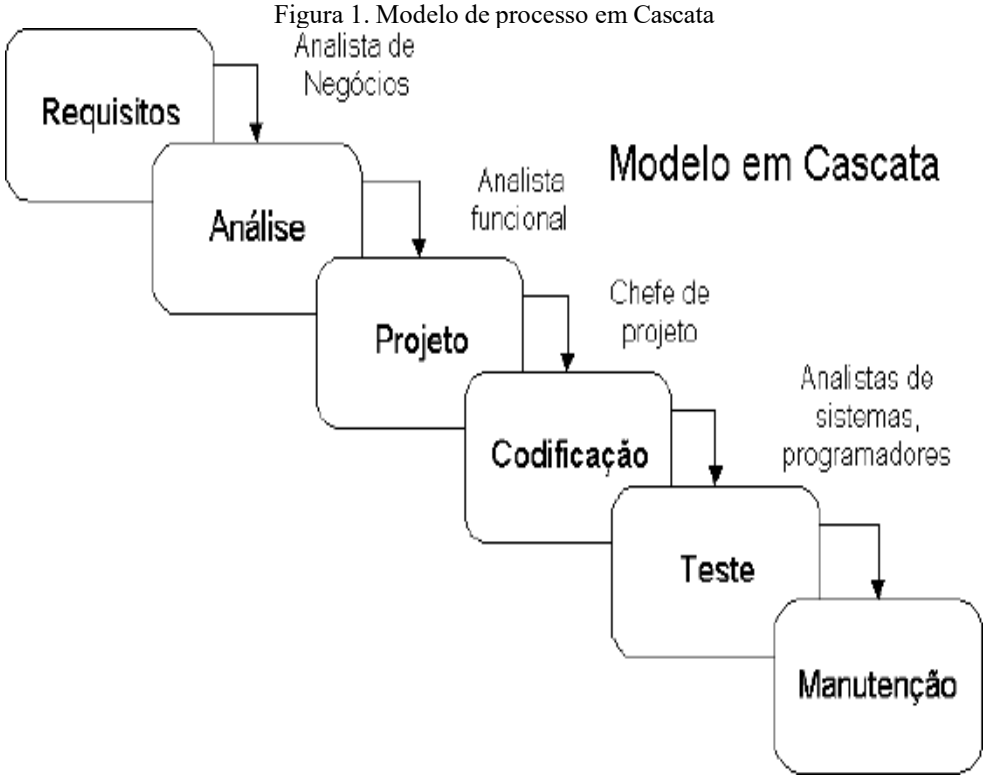
O modelo em cascata caracteriza-se por adotar uma abordagem linear e sequencial no desenvolvimento de software. Nele, cada fase do processo deve ser concluída integralmente antes que a

próxima se inicie, sem a possibilidade de retornos ou revisões entre as etapas. Essa estrutura favorece o controle organizacional e gerencial, permitindo que cada estágio tenha prazos bem definidos e contribuindo para que o produto avance de forma ordenada até sua conclusão, teoricamente dentro do cronograma previsto. O fluxo de desenvolvimento segue uma trajetória rígida, que parte do conceito inicial, passa pelo projeto (design), implementação, testes, instalação, detecção de defeitos, até chegar à operação e manutenção, sem sobreposição ou repetições entre as fases (PRESSMAN, 2006). Contudo, a principal limitação desse modelo reside na baixa flexibilidade para revisões. Assim, quando o sistema atinge a etapa de testes, torna-se difícil modificar elementos que foram mal planejados nas fases iniciais, especialmente no conceito do projeto.

O modelo em cascata parte de uma visão orientada a projetos, tratando o desenvolvimento de software como uma tarefa bem definida, cujos resultados são determinados com precisão desde o início. Entretanto, Pressman (1995) destaca alguns problemas que surgem na prática da aplicação desse modelo. Um deles é o fato de que projetos reais raramente seguem um fluxo totalmente linear, havendo sempre alguma forma de iteração, o que dificulta a aplicação rigorosa do paradigma. Além disso, é comum que o cliente tenha dificuldades para especificar todas as exigências de forma clara no início do projeto, e o ciclo de vida tradicional exige justamente essa definição precoce, o que nem sempre é viável. Outro ponto crítico é a necessidade de paciência por parte do cliente, já que uma versão funcional do software só é disponibilizada tardiamente, o que pode ser problemático caso algum erro grave só seja identificado nesse estágio, acarretando grandes prejuízos.

Complementando essa análise, Peters (2001) aponta como vantagem do modelo cascata a facilidade de gerenciamento dos "baselines", ou seja, dos conjuntos fixos de documentos gerados ao final de cada etapa do desenvolvimento. No entanto, entre as desvantagens, destaca-se a dificuldade de aplicar engenharia reversa em sistemas já existentes, bem como a ausência de mecanismos para prototipagem rápida e desenvolvimento incremental, aspectos considerados importantes nos contextos de desenvolvimento contemporâneos.

Figura 1. Modelo de processo em Cascata



Fonte: LESSA; LESSA JUNIOR, [s.d.]

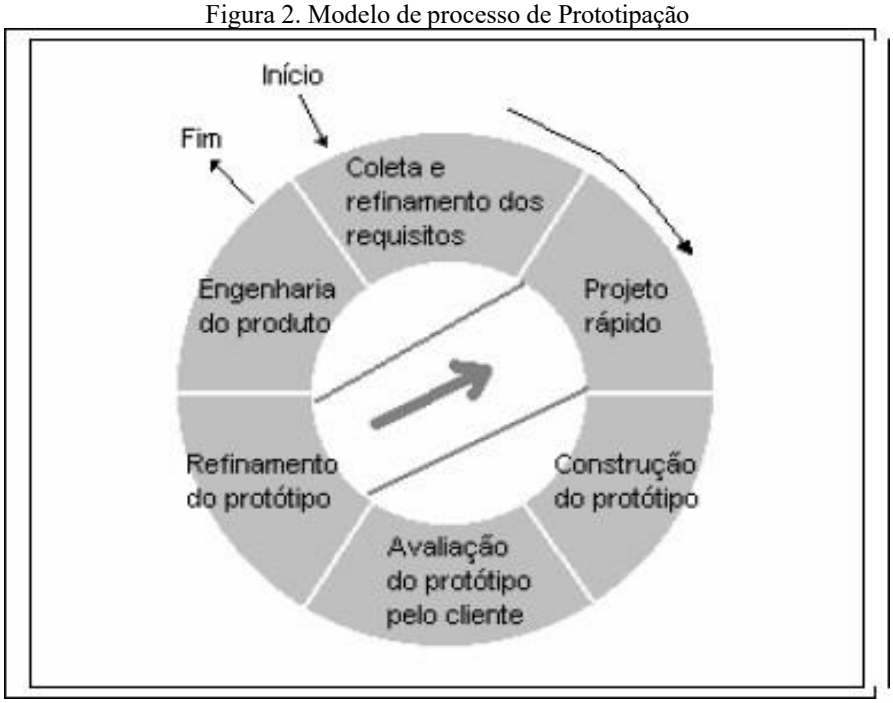
2.3 PROTOTIPAÇÃO

A prototipação consiste na construção de versões preliminares de um sistema — os chamados protótipos — e pode ser classificada com base em diferentes critérios. Essa abordagem apresenta diversas vantagens relevantes, sendo a principal delas o fato de que não é necessário definir todos os requisitos do sistema de forma completa e antecipada, pois eles podem ser modificados ao longo do desenvolvimento. Além disso, a prototipação contribui para a entrega de definições claras e compreensíveis do sistema, o que facilita a elaboração das especificações voltadas ao usuário final. Como consequência, há um aumento significativo no envolvimento e na satisfação dos usuários, uma vez que eles conseguem visualizar e interagir com versões iniciais do sistema. Outro benefício importante é a possibilidade de testar rapidamente o ambiente de desenvolvimento, avaliando aspectos como funcionalidades, desempenho e integração com banco de dados (IBM, 2002).

Apesar dessas vantagens, a prototipação também apresenta algumas limitações. Um dos problemas recorrentes é o início precoce da modelagem, muitas vezes sem uma análise adequada da situação atual e da situação desejada, bem como sem uma compreensão clara do problema a ser resolvido — aspectos que são tão importantes quanto a busca por soluções. Esse risco se intensifica no caso da prototipação evolucionária, na qual, a cada iteração, o protótipo pode ser ajustado de maneira inadequada, incorporando menos funcionalidades do que o necessário ou incluindo elementos supérfluos, comprometendo o resultado final do projeto. Outro risco significativo é relacionado à percepção do usuário final. O processo de

prototipação pode gerar uma falsa expectativa de que qualquer sugestão será facilmente implementada, independentemente da fase em que o desenvolvimento se encontra. Além disso, ao verem uma versão demonstrativa do sistema funcionando, muitos usuários não compreendem por que há demora na entrega da aplicação final, desconhecendo as complexidades e etapas técnicas que ainda precisam ser cumpridas (IBM, 2002).

Figura 2. Modelo de processo de Prototipação



Fonte: LESSA; LESSA JUNIOR, [s.d.]

2.4 MODELO DE PROCESSO DE SOFTWARE EVOLUCIONÁRIO

Pesquisas indicam que, assim como outros sistemas complexos, o software tende a evoluir ao longo do tempo. Durante esse processo, os requisitos de negócio e do produto frequentemente se modificam, dificultando uma trajetória linear até a obtenção de um produto final completo (PRESSMAN, 2006). Diante desse cenário dinâmico, surgem modelos de desenvolvimento iterativos, que buscam construir versões progressivamente mais completas do software.

Um dos modelos que seguem essa lógica é o modelo espiral, proposto por Barry Boehm em 1988. Ele combina elementos positivos do modelo clássico em cascata e da prototipação, acrescentando um diferencial essencial: a análise contínua de riscos. Essa estrutura é composta por quatro quadrantes fundamentais: planejamento (definição de objetivos, alternativas e restrições), análise de riscos (identificação e resolução de problemas potenciais), engenharia (desenvolvimento do próximo nível do produto) e avaliação pelo cliente (validação dos resultados obtidos). O modelo espiral adota uma abordagem evolucionária da engenharia de software, permitindo que desenvolvedores e usuários compreendam e

enfrentem os riscos de forma sistemática em cada ciclo. A prototipação é empregada como uma estratégia de mitigação de riscos, podendo ser aplicada em qualquer etapa do desenvolvimento. Essa metodologia integra a disciplina e a estrutura sequencial do ciclo de vida clássico a uma dinâmica iterativa, mais condizente com os contextos reais de desenvolvimento. O modelo exige que os riscos técnicos sejam considerados desde o início do projeto, de modo a evitá-los antes que comprometam o produto (PRESSMAN, 2006).

Além disso, o modelo espiral reconhece que o processo de desenvolvimento de software não pode ser totalmente definido desde o início. Através de ciclos evolutivos, cada etapa do projeto contempla atividades como a definição de metas e restrições, a avaliação de alternativas frente aos objetivos estabelecidos, a elaboração técnica dos componentes do sistema e, por fim, o planejamento do próximo ciclo, sendo possível inclusive encerrar o projeto caso se constate um risco elevado.

Outro modelo relevante é o modelo incremental, que, segundo Pressman (2006), combina aspectos do modelo cascata com uma execução iterativa. Ele difere da prototipação por entregar, a cada incremento, um produto funcional. Essa característica torna o modelo especialmente útil em contextos nos quais a empresa não dispõe de todos os recursos humanos necessários para uma implementação completa e imediata, permitindo que o sistema seja desenvolvido por partes, dentro do prazo estipulado.

O desenvolvimento baseado em componentes, também chamado de *component-based development* (CBD) ou *component-based software engineering* (CBSE), fundamenta-se na reutilização de módulos de software previamente criados. Embora Pressman (2006) não forneça uma definição exata de componente, ele associa esse modelo ao paradigma da orientação a objetos, no qual as classes são vistas como elementos reutilizáveis que encapsulam dados e algoritmos. Esse modelo integra a lógica iterativa do modelo espiral à prática da construção de bibliotecas de classes, que podem ser reutilizadas ao longo de diferentes projetos. A cada ciclo da espiral, avalia-se se as classes existentes na biblioteca são suficientes ou se é necessário desenvolver novas para uso futuro.

Diante da complexidade e abrangência da Engenharia de Software, o IEEE formou um comitê responsável pela criação do SWEBOK (*Software Engineering Body of Knowledge*), um guia que organiza essa área em dez grandes áreas de conhecimento: Requisitos de Software, Projeto (Design) de Software, Construção de Software, Teste de Software, Manutenção de Software, Gerência de Configuração de Software, Gerência de Engenharia de Software, Processos de Engenharia de Software, Ferramentas e Métodos de Engenharia de Software, e Qualidade de Software. Cada área é subdividida em tópicos específicos, que compõem o escopo dos conhecimentos fundamentais que os engenheiros devem dominar. Entretanto, o próprio guia reconhece que o conhecimento exigido na área vai além do que está ali formalizado, incluindo disciplinas associadas como engenharia da computação, ciência da computação, matemática, gerenciamento de projetos e da qualidade, ergonomia e engenharia de sistemas.

O SWEBOK passou por três versões, representadas metaforicamente por estágios de robustez:

"homem de palha", "homem de pedra" (1998–2001) e, por fim, "homem de ferro" (2003), considerada a versão final. Contudo, por mais abrangente que seja, o guia não consegue acompanhar com total precisão as constantes transformações que ocorrem no campo da engenharia de software, especialmente com o surgimento de novas tecnologias e práticas, o que pode torná-lo parcialmente defasado com o tempo.

3 CONCLUSÃO

A conclusão deste trabalho reafirma a importância dos modelos de processos de engenharia de software como ferramentas fundamentais para a organização, o controle e a qualidade no desenvolvimento de sistemas computacionais. Ao longo da pesquisa, foi possível observar que, embora existam diversos modelos disponíveis, cada um apresenta características, vantagens e limitações que os tornam mais adequados a determinados contextos e necessidades. O modelo cascata, por exemplo, se destaca pela estrutura sequencial e previsível, mas apresenta dificuldades em se adaptar a mudanças de requisitos ao longo do projeto. Já a prototipação oferece maior flexibilidade e interação com o usuário, embora possa gerar expectativas irreais e problemas de definição de escopo. O modelo espiral, por sua vez, introduz de forma eficaz a análise de riscos como elemento central, oferecendo uma estrutura iterativa que contempla tanto planejamento quanto avaliação contínua, sendo mais apropriado para projetos complexos e sujeitos a incertezas. O modelo incremental propõe uma entrega progressiva de funcionalidades, possibilitando melhorias contínuas e validações parciais durante o desenvolvimento, enquanto o desenvolvimento baseado em componentes enfatiza o reuso de soluções e a modularização, contribuindo para a eficiência e a padronização de sistemas maiores. Além disso, o estudo do SWEBOK demonstrou que a engenharia de software é uma área dinâmica e multidisciplinar, que requer conhecimento técnico atualizado, domínio de processos e compreensão das relações entre a engenharia e outras áreas do conhecimento, como gestão, ergonomia, matemática e ciência da computação. A análise dos modelos reforça a hipótese de que a escolha adequada de um processo de desenvolvimento está diretamente associada à complexidade do projeto, ao perfil da equipe, ao grau de maturidade da organização e às demandas específicas dos usuários. Também foi possível perceber que as abordagens iterativas, como o espiral e o incremental, têm se mostrado mais eficazes frente à volatilidade dos requisitos e à necessidade de maior interação com os usuários finais. Por fim, conclui-se que, para garantir o sucesso de um projeto de software, é essencial que os profissionais da área estejam atentos não apenas às ferramentas e técnicas disponíveis, mas também às transformações constantes do setor tecnológico, à busca por qualidade contínua e ao compromisso com as boas práticas da engenharia de software. O aprimoramento contínuo dos métodos de desenvolvimento e a adoção consciente dos modelos de processo são, portanto, caminhos indispensáveis para a construção de soluções eficazes, sustentáveis e alinhadas às exigências do mercado atual.

REFERÊNCIAS

BROWN, ALAN W., On Components and *Objects: The Fundation of Component- Based Development,* Assessment of Software Tools and Tecnology, Procedings Fifth International Symposium on Proceedings - IEEE, 1997.

IBM; Practicing Object-Oriented Analysis and Design- ERC2.2.; IBM Education andTraining; 2002L.

LESSA, Rafael Orivaldo; LESSA JUNIOR, Edson Orivaldo**.** *Princípios da engenharia de software*. Palhoça, SC: Universidade do Sul de Santa Catarina – UNISUL, [s.d.].

PRESSMAN, ROGER S., *Engenharia de Software- (3ª edição)*, São Paulo, Ed. MakronBooks, 1995.

PRESSMAN, ROGER S., *Engenharia de Software- (6ª edição)*, São Paulo, Ed. McGrawHill, 2006.

PETERS, JAMES F., *Engenharia de Software: Teoria e Prática*, Rio de Janeiro, Editora Campus, 2001.

SOMMERVILLE, I. Software Engineering (International Computer Science Series). 5a Edição. Reading: Addison-Wesley, 1995.

SWEBOK 2004, Guide for the Software Engineering Body of Knowledge, 2004 version, IEEE Computer Society, California, EUA.