Reutilização de Software*

Hugo Silva m53080@alunos.uevora.pt

Universidade de Évora, Évora, Portugal

Abstract. O presente artigo aborda a reutilização de software como um pilar essencial no desenvolvimento de software. A reutilização de software consiste em reutilizar software existente para criar novos sistemas, permitindo assim, poupar tempo e recursos, aumentando a qualidade de software. É abordado os problemas associados à reutilização de software, como também as formas de minimizar os problemas que podem surgir devido à sua reutilização. O presente artigo aborda também o desenvolvimento de software baseado em componentes e em serviços, como as vantagens e benefícios associados à reutilização de software. Por fim, é realizada a conclusão sobre a reutilização de software.

Keywords: Reutilização de Software · Desenvolvimento baseado em componentes e serviços · Problemas e soluções na reutilização de software.

1 Introdução

O presente trabalho aborda a estratégia de reutilização de software, tendo como base de referência o livro. . . . . .

A reutilização de software é uma estratégia de desenvolvimento utilizada de forma generalizada que visa reutilizar software existente. Esta estratégia tem vindo a ser utilizada pois permite reduzir os custos de desenvolvimento e manutenção, aumentar a velocidade de entrega dos sistemas informáticos e aumentar a qualidade do software.

Cada vez mais as empresas têm visto o seu software como um ativo importante, adotando a estratégia de reutilização de software as empresas podem aumentar o seu retorno no investimento.

Existem várias unidades de software que podem ser utilizados, como sistemas completos, aplicações, componentes e objetos/funções.

A reutilização de software apresenta variadíssimas vantagens, como a aceleração do desenvolvimento, o uso efetivo de especialistas, maior confiança no software, custo reduzido de manutenção e desenvolvimento, redução da margem de erro nas estimativas dos projectos e um aumento da conformidade com os padrões, que aumenta a confiança no software e reduz os erros.

No entanto, a estratégia de software também apresenta alguns problemas, como possíveis aumentos de custos de manutenção, falta de suporte para as

* Universidade de Évora.

ferramentas, dificuldade em criar e manter uma biblioteca de componentes e o síndrome de “Não inventado aqui”.

É abordado o desenvolvimento baseado em componentes e serviços, descrevendo brevemente o que são e referindo os seus benefícios.

Por fim, é realizada a conclusão sobre o presente trabalho, destacando a importância da reutilização de software.

2 Reutilização de Software

A reutilização de Software (Software Engineering (Ian Sommer Ville)) é uma estratégia utilizada em engenharia de software que tem como intuito reutilizar software existente.

A estratégia baseada em reutilização de Software têm sido adotada devido à necessidade de:

- Menores custos de produção e manutenção de software;
- Maior velocidade de entrega dos sistemas informáticos;
- Aumento da qualidade do software.

As empresas cada vez mais veem o seu software como um ativo importante e têm vindo a adotar a estratégia de reutilização do software, pois podem aumentar o seu retorno sobre os investimentos no software.

As unidades de software que podem ser reutilizadas podem ser de diferentes tamanhos, sendo:

- Reutilização completa dos Sistemas;
- Reutilização da Aplicação;
- Reutilização de Componentes;
- Reutilização de Objectos e funções.

Todo o software e componentes que incluem funcionalidades genéricas podem ser potencialmente reutilizáveis, no entanto, existem certos softwares e componentes que por vezes são tão específicos que são muito custosos de modificar para a lógica de negócio pretendida.

2.1 Vantagens e benefícios associados à reutilização de software

Existem variadíssimas vantagens e benefícios associados à estratégia de reutilização de software, estes são:

- Aceleração do desenvolvimento;
- Uso efectivo de especialistas;
- Maior confiança no software;
- Baixo custos de desenvolvimento;
- Redução da margem de erro na estimativa dos projectos;
- Conformidade com os padrões resulta em maior confiabilidade e menos erros.

Aceleração do desenvolvimento. A reutilização de software pode acelerar o desenvolvimento do software, isto porque, o tempo de desenvolvimento e validação podem ser reduzidos. Trazer um sistema para o mercado o mais cedo possível é normalmente mais importante do que os custos totais de desenvolvimento.

Uso efectivo de especialistas. A reutilização de software permite que os especialistas se foquem no desenvolvimento de novo software reutilizável que encapsula os seus conhecimentos, em vez de estarem constantemente a fazer o mesmo trabalho.

Maior confiança no software. A reutilização de software estabelece uma maior confiança no software existente porque o software reutilizável normalmente já foi experimentado e testado anteriormente em sistemas reais, o que permitiu encontrar e corrigir erros de design e de implementação.

Baixo custos de desenvolvimento. A reutilização de software pode permitir baixar os custos de desenvolvimento de software.. Isto porque, é possível reutilizar componentes e serviços já existentes, em vez de desenvolver software do zero. Reduzindo assim o tempo e recursos envolvidos no processo de desenvolvimento.

Redução da margem de erro na estimativa dos projectos. A reutilização de software ajuda na redução de margem de erro na estimativa dos projectos. Isto porque o software já passou por testes e validação em outros projetos, reduzindo assim o risco de erros ou problemas encontrados.

O custo do software existente já é conhecido, enquanto os custos de desenvolvimento estão sempre dependentes sobre o julgamento. Este é um importante factor a ter em consideração na gestão do projecto, porque reduz a margem de erro na estimativa.

Conformidade com os padrões resulta em maior confiabilidade e menos erros. Alguns padrões, como padrões de interfaces, podem ser implementados como componentes reutilizáveis. Por exemplo, se um menu é implementado utilizando um componente reutilizável, todas as aplicações apresentam o mesmo formato de menu ao utilizador. O uso de padrões melhora a confiabilidade e usabilidade, pois, os utilizadores fazem menos erros quando apresentados com uma interface familiar.

2.2 Problemas associados à reutilização de software

A estratégia de reutilização de software, também têm certos problemas associados, estes são:

- Criação, manutenção e uso de uma biblioteca de componentes;
- Encontrar, entender e adaptar componentes reutilizáveis;
- Aumento de custos de manutenção;
- Falta de suporte para ferramentas;
- Síndrome de “Não inventado aqui”.

Criação, manutenção e uso de uma biblioteca de componentes. Popular um componente reutilizável de uma biblioteca e assegurar que os programadores possam utilizar a biblioteca pode ser bastante custoso. Os processos de desenvolvimento devem-se adaptar para garantir que a biblioteca é utilizada.

Encontrar, entender e adaptar componentes reutilizáveis. Componentes de software têm de ser encontrados, entendidos e muitas das vezes adaptados para funcionarem num novo ambiente de desenvolvimento. Os engenheiros devem ser confiantes o suficiente para poderem encontrar um componente na biblioteca, antes de o incluírem no seu normal processo de desenvolvimento.

Aumento de custos de manutenção. Se, por exemplo, o código fonte de um sistema ou componente reutilizável não estiver disponível, os custos de manutenção podem ser maiores. Isto porque, os elementos reutilizáveis do sistema podem-se tornar incompatíveis com as mudanças feitas ao sistema.

Falta de suporte para ferramentas. Algumas ferramentas de software não suportam um desenvolvimento baseado em reutilização. Pode ser difícil ou impossível implementar estas ferramentas com um sistema de libraria de componentes.

No processo de desenvolvimento e design deste software, podem não ter considerado o desenvolvimento baseado em reutilização. É mais provável isto acontecer para ferramentas que suportam Embedded Systems do que sistemas Orientados a Objetos.

Síndrome de "Não inventado aqui". Alguns engenheiros de software preferem reescrever os componentes porque acreditam que os podem melhorar. Isto acontece parcialmente devido à falta de confiança e parcialmente devido ao facto que escrever software original é visto como algo mais desafiante que reutilizar o código de outros.

2.3 Formas para minimizar os problemas que podem surgir devido à reutilização de software

Existem variadíssimos factores que devem ser considerados para minimizar os problemas que podem surgir devido à reutilização de software. Alguns dos fatores a ter em consideração quando a planear a reutilização de software, são:

- O cronograma de desenvolvimento de software;
- O tempo de vida esperado do software;
- O histórico, as habilidades e a experiência da equipa de desenvolvimento;
- A importância do software e os requisitos não funcionais;
- O domínio da aplicação;
- A plataforma na qual o sistema será executado.

O cronograma de desenvolvimento de software. Se o software tem de ser desenvolvido rapidamente, deve ser tentado reutilizar sistemas completos em vez de componentes individuais. Isto pois, apesar dos requerimentos poderem ser imperfeitos, esta abordagem minimiza o tempo de desenvolvimento necessário.

O tempo de vida esperado do software. Quando se está a desenvolver um sistema de software que terá uma vida longa, deve se ter em consideração a manutenção do sistema. Não se deve pensar apenas nos benefícios imediatos da realização mas também as suas implicações futuras.

Durante a vida do software será necessário adaptar o sistema a novos requisitos. Se não se têm acesso ao código fonte dos componentes reutilizáveis, pode ser preferivel evitar componentes e sistemas de fornecedores externos. Isto pois, estes fornecedores externos podem deixar de continuar a suportar o software reutilizável. Pode ser mais seguro e útil utilizar componentes e sistemas reutilizáveis de software open-source, porque, utilizando software open-source podemos aceder e criar cópias do código fonte.

O histórico, as habilidades e a experiência da equipa de desenvolvimento. É necessário bastante tempo para compreender e conseguir utilizar efetivamente software reutilizáveis, isto pois, este tipo de software costuma ser bastante complexo. Por isso, deve-se concentrar o esforço de reutilização em áreas onde a equipa tenha experiência.

A importância do software e os requisitos não funcionais. Para um sistema crítico que é necessário ser certificado por um regulador externo, pode ser necessário criar um mecanismo de proteção ou segurança para o sistema. Isto é difícil se não se têm acesso ao código fonte do software. Se o sistema tem requisitos de desempenho rigorosos pode ser impossível utilizar certas estratégias, isto pois, estas estratégias normalmente geram código ineficiente.

O domínio da aplicação. Em muitos domínios de aplicação, como sistemas de fabricação e sistemas de informação de medicina, existem produtos genéricos que podem ser reutilizados configurando-os para uma situação local. Esta é uma das formas mais eficientes de utilizar reutilização, é também quase sempre mais barato comprar do que construir um novo sistema.

A plataforma na qual o sistema será executado. Alguns modelos de componente, como por exemplo, .NET são específicos da Microsoft.

O software reutilizável pode ser específico para certas plataformas, a sua utilização só é possível nessas plataformas. É necessário ter em consideração este elemento para minimizar problemas na reutilização do software.

2.4 Desenvolvimento de software baseado em componentes

O desenvolvimento de software baseado em componentes surgiu devido a necessidade de se ter um software mais confiável e seguro, que pode ser entregue e lançado mais rapidamente.

Desenvolvimento de Software baseado em componentes (referência) é o processo de definir, implementar, e integrar ou compor os componentes soltos independentes em sistemas.

Um componente é um pedaço de software independente que pode ser combinado com outros componentes para criar um sistema de software.

Os essenciais do desenvolvimento de software baseado em componentes são:

1. Componentes independentes que são completamente especificados pelas suas interfaces.
2. Padrões de componentes que definem interfaces e facilitam a integração de componentes.
3. Middleware que fornece suporte de software para a integração de componentes.
4. Um processo de desenvolvimento que é direcionado para a engenharia de software baseada em componentes.

2.5 Desenvolvimento de software baseado em serviços

Sistemas de software baseado em serviços são sistemas que são implementados utilizando componentes de serviços reutilizáveis e que são acedidos por outros programas, em vez de diretamente pelos utilizadores. No entanto, não é necessário implementar o software desta maneira para ser oferecido como um serviço.

Um serviço web é um componente solto reutilizável que encapsula funcionalidades discretas, que podem ser distribuídas e acessíveis programaticamente. Um serviço web é um serviço que é acedido utilizando a internet padrão e protocolos XML.

Os serviços são independentes da plataforma e da linguagem de programação utilizada.

O desenvolvimento de software baseado em serviços oferece bastantes benefícios, estes são:

1. Serviços podem ser oferecidos por qualquer fornecedor de serviços, seja de dentro ou de fora da organização.
2. O fornecedor do serviço faz com que a informação do serviço seja pública para que qualquer utilizador autorizado possa utilizar o serviço.
3. As aplicações podem atrasar a ligação dos serviços até que sejam implementados ou até que sejam executados.
4. A criação oportunista de novos serviços é possível.
5. Utilizadores de serviços podem pagar por serviços de acordo com a sua utilização, em vez da sua disposição.

6. As aplicações podem ser feitas mais pequenas, o que é particularmente importante para utilizadores de smartphones onde a capacidade de processamento e de memória são limitados.

É de extrema importância num sistema baseado em serviços, garantir que todos os serviços que compõem o sistema, são compatíveis entre si e que respeitam os requisitos pré definidos, isto para garantir que nenhum dos serviços do sistema comprometa o bom funcionamento do sistema.

3 Conclusão

A reutilização de software é no meu ponto de vista um pilar essencial na engenharia do software. Isto pois, ajuda efetivamente na redução de custos, acelerando assim o desenvolvimento e aumentando a qualidade do software desenvolvido.

Com o aparecimento da internet, a reutilização de software ganhou outro relevo, isto pois, programadores podem contribuir para projetos open-source, podem criar o seu próprio software reutilizável e vender como um serviço através da internet, entre outros. Disponibilizando assim software reutilizável para milhares de milhões de pessoas.

Esta transformação permitiu o aceleramento da digitalização e evolução da sociedade como um todo, permitindo assim conectar e aumentar a produtividade do mundo.

É necessário no entanto, ter alguns fatores em consideração para minimizar os problemas que podem surgir devido à reutilização de software, isto para garantir a integridade e o bom funcionamento do sistema.

Concluiu então, que a reutilização de software é uma das abordagem de desenvolvimentos de software mais importantes nos dias de hoje, isto pois, permite acelerar o desenvolvimento, reduzir os custos e aumentar a qualidade do software.

References

1. Author, F.: Article title. Journal **2**(5), 99–110 (2016)
2. Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016). https://doi.org/10.10007/1234567890
3. Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999)
4. Author, A.-B.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010)
5. LNCS Homepage, http://www.springer.com/lncs. Last accessed 4 Oct 2017

4 First Section

4.1 A Subsection Sample

Please note that the first paragraph of a section or subsection is not indented. The first paragraph that follows a table, figure, equation etc. does not need an indent, either.

Subsequent paragraphs, however, are indented.

Sample Heading (Third Level) Only two levels of headings should be numbered. Lower level headings remain unnumbered; they are formatted as run-in headings.

Sample Heading (Fourth Level) The contribution should contain no more than four levels of headings. Table 1 gives a summary of all heading levels.

Table 1. Table captions should be placed above the tables.

Heading level	Example	Font size and style
Title (centered)	**Lecture Notes**	14 point, bold
1st-level heading	**1 Introduction**	12 point, bold
2nd-level heading	**2.1 Printing Area**	10 point, bold
3rd-level heading	**Run-in Heading in Bold.** Text follows	10 point, bold
4th-level heading	*Lowest Level Heading.* Text follows	10 point, italic

Displayed equations are centered and set on a separate line.

$$x + y = z \tag{1}$$

Please try to avoid rasterized images for line-art diagrams and schemas. Whenever possible, use vector graphics instead (see Fig. 1).

Theorem 1. *This is a sample theorem. The run-in heading is set in bold, while the following text appears in italics. Definitions, lemmas, propositions, and corollaries are styled the same way.*

Proof. Proofs, examples, and remarks have the initial word in italics, while the following text appears in normal font.

For citations of references, we prefer the use of square brackets and consecutive numbers. Citations using labels or the author/year convention are also acceptable. The following bibliography provides a sample reference list with entries for journal articles [1], an LNCS chapter [2], a book [3], proceedings without editors [4], and a homepage [5]. Multiple citations are grouped [1–3], [1, 3–5].

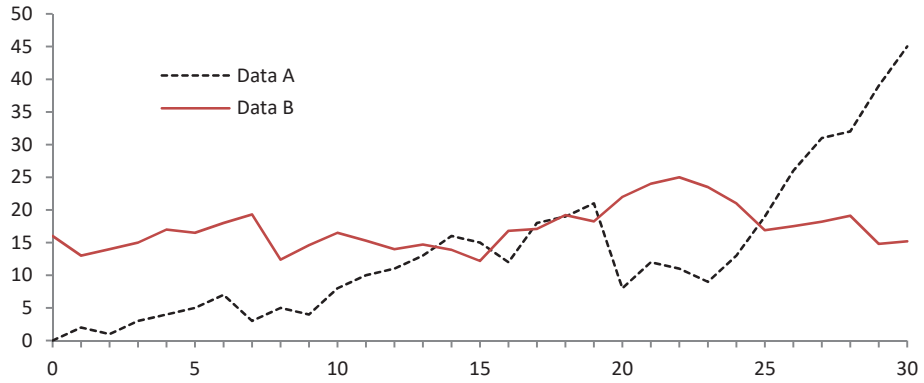

Fig. 1. A figure caption is always placed below the illustration. Please note that short captions are centered, while long ones are justified by the macro package automatically.

References

1. Author, F.: Article title. Journal **2**(5), 99–110 (2016)
2. Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016). https://doi.org/10.10007/1234567890
3. Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999)
4. Author, A.-B.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010)
5. LNCS Homepage, http://www.springer.com/lncs. Last accessed 4 Oct 2017