1
Introdução

A customização em massa de linha de produtos é hoje uma estratégia de negócio bastante atrativa para as fábricas de software que necessitam que seus produtos satisfaçam os requisitos individuais de seus clientes [1]. Essa estratégia é estudada dentro da engenharia de linha de produtos de software (ELPS). A ELPS, através do seu conjunto de técnicas, provê subsídios aos projetistas na atividade de desenho de uma linha de produtos, permitindo que diversas funcionalidades possam ser inseridas ou removidas de um produto, conforme a necessidade de quem o está adquirindo. Com esse conjunto de técnicas, diversas qualidades são agregadas aos produtos finais, principalmente devido ao reúso dos componentes [2].

A engenharia de linha de produtos de software possui duas atividades principais: *engenharia de domínio* e *engenharia de aplicação* [3]. A engenharia de domínio consiste em desenvolver um conjunto de artefatos que podem ser configurados e combinados para criar diferentes produtos de software. Por outro lado, a engenharia de aplicação tem como objetivo resolver progressivamente as variabilidades providas pela engenharia de domínio, através de um *processo de configuração*. O cerne da engenharia de aplicação é a especificação de um produto que reflita os requisitos impostos, através da escolha de algumas funcionalidades ou do descarte de outras.

Uma atividade de configuração de produto bem conduzida é crucial para a customização em massa [4]. Nessa atividade podem participar diversas pessoas, grupos ou organizações que estão de alguma forma relacionados com a linha de produtos de software ou com o contexto que estará inserido o produto a ser gerado. Chamamos essas entidades de *stakeholders configuradores*. Como primeiro passo para atividade de configuração, esses *stakeholders* avaliam quais dos pontos de variabilidade da linha de produtos satisfazem seus requisitos, decidindo sobre características técnicas e não técnicas.

Uma técnica comum para representar uma linha de produtos de software, assim como suas variabilidades e pontos em comum, é o *modelo de features* [5]. Além disso, nesse modelo também é possível representar as restrições impostas sobre os estados relativos entre as *features*. Uma *feature* é uma propriedade ou

funcionalidade da linha de produtos de software que é relevante para algum *stakeholder* [6].

Portanto, o resultado da atividade de configuração de um produto de uma linha de produtos é um conjunto de *features* escolhidas que estarão presentes no produto final. Essas *features* devem satisfazer os requisitos de negócio impostos e respeitar as restrições definidas no modelo de *features* que representa essa linha de produto. Dessa maneira, com a configuração em mãos, é possível utilizar alguma ferramenta de geração de produtos para construir o sistema desejado [7]. Esse, por fim, é entregue ao cliente de maneira praticamente instantânea.

1.1 Problema e Limitação das Abordagens Atuais

No contexto de grandes projetos industriais, o processo de configuração de produto pode ser por si mesmo uma atividade complexa, consumindo vários meses e bastantes recursos [4, 8]. Isso se dá especialmente quando múltiplas e grandes linhas de produtos são consideradas [9]. Além disso, a atividade de configuração, em geral, é mais efetiva quando conduzida de forma colaborativa [6]. Em contraste à configuração realizada por apenas um stakeholder, a colaboração permite que diversos *stakeholders* com diferentes especialidades participem da atividade de configuração, o que tende a aumentar a confiabilidade do produto gerado [10].

Infelizmente, a colaboração na configuração de produtos, além dos benefícios, introduz novos assuntos e desafios, tais como a coordenação e integração correta das seleções das variabilidades, que vão além do cumprimento de restrições expressas pelo modelo de *features* [11]. Quando vários profissionais participam de um trabalho colaborativo de configuração, questões como as seguintes podem surgir:

- Como serão distribuídas as responsabilidades sobre cada *feature* do modelo entre os *stakeholders*? Mais de um *stakeholder* pode atuar sobre uma mesma feature?
- Como organizar a participação de cada *stakeholder* na atividade configuração?
- Tendo em vista que os *stakeholders* podem ter perspectivas diferentes e que decisões entre eles podem produzir estados de configurações inconsistentes, como identificar esses conflitos? Como gerenciá-los? E, principalmente, como resolvê-los?

Algumas soluções que são frequentemente recomendadas e aplicadas para coordenar múltiplos *stakeholders*, propostas pela comunidade que estuda esse tema, sugerem uma orquestração das atividades de decisão de cada *stakeholder* como um *workflow*. Esses trabalhos, salvo alguns pontos de diferença na forma como tratam as atividades internas, de forma geral, propõem uma configuração dividida em etapas sequenciais ou paralelas organizados em um fluxo que é definido *a piori*. Quando aplicado, eles garantem que apenas produtos válidos serão construídos. Por exemplo: verificar se todas as fases respeitam a semântica do fluxo de trabalho de configuração [12]; ou organizar as decisões dependentes, como estágios sequenciais [11]. No entanto, ainda existem alguns problemas em aberto que exigem atuações mais dinâmicas.

Negociação interativa sobre os requisitos do produto. A atividade de configuração de produto pode exigir decisões dinâmicas e interativas [13]. Por exemplo, as abordagens atuais especificam que os *stakeholders* realizem suas decisões de forma gradual. Porém, quando os requisitos do cliente são imprecisos e a negociação é um cenário comum, numa etapa posterior, um dos *stakeholders* pode desejar que um novo requisito seja respeitado. Uma implicação recorrente é que os requisitos não podem ser facilmente traduzidos para uma nova seleção de *features* devido a uma ou várias decisões realizadas em fases anteriores, que excluem a possibilidade dessa nova seleção. Descobrir como alterar decisões anteriores para tornar uma *feature* selecionável pode ser complicado e uma identificação prévia de todos os cenários possíveis de reconfiguração do fluxo de atividades pode ser impraticável.

Integração de decisões distribuídas. O desenvolvimento econômico da Web encoraja fábricas de software a distribuir suas atividades de desenvolvimento de software entre equipes geograficamente distribuídas. A complexidade da integração de decisões pode aumentar nesse contexto. É requerida a integração imediata de decisões e o cumprimento das restrições impostas sobre *features* que podem estar sendo configuradas de forma distribuída. Existe um alto risco de introdução de inconsistências quando as decisões não estão bem integradas e todas as restrições não são verificadas no momento da configuração. Como consequência, sanar inconsistências que só surgem em fases finais pode ser difícil, como mencionado anteriormente.

Definição de prioridade entre *stakeholders*. Uma das implicações da utilização de fluxo de atividades para coordenar as atividades de configuração de cada *stakeholder* é a imposição implícita de prioridade maior aos *stakeholders* iniciais. Quando existe a atuação de alguns *stakeholders* antes de outros e existem dependências entre as *features* relacionadas, os *stakeholders* que realizaram suas configurações depois podem ficar impedidos ou serem

obrigados a selecionar ou excluir algumas de suas *features*. Essa priorização de alguns *stakeholders* surge como efeito colateral do uso de *workflows* e pode demandar negociações que não são totalmente suportadas pelas abordagens atualmente propostas. Portanto, uma abordagem que permita a definição das prioridades entre *stakeholders* de uma maneira explícita – e que trata todas as implicações de seu uso – agrega valor ao processo de configuração colaborativa.

Ferramenta de suporte à configuração colaborativa de produto. Atualmente, várias abordagens [14] e ferramentas [15, 16] foram desenvolvidas para ajudar na redução da complexidade na atividade de configuração baseada em um único *stakeholder* pela automatização de partes do processo de seleção de *features*. No entanto, os desafios mencionados anteriormente necessitam de ferramentas que, adicionalmente, (i) facilitem a integração de decisões distribuídas, (ii) forneçam um modelo de coordenação que suporte o dinamismo da atividade de configuração, e (iii) provenham validação dos produtos e mecanismos de correção que levem em conta as restrições distribuídas.

1.2 Solução Proposta

Este trabalho propõe uma abordagem colaborativa para configuração de produto dada uma linha de produtos de software. Uma ferramenta denominada *Libertas* também é apresentada e tem como objetivo implementar os conceitos propostos pela abordagem para auxiliar na avaliação e validação. Técnicas do campo de Engenharia de Software orientada a Agentes [17], tais como *agentes autônomos*, são utilizadas principalmente para coordenação do processo de configuração, comunicação e integração das decisões entre *stakeholders*, para validação da configuração e sugestão de ações de correção, no caso em que a configuração esteja em um estado inválido.

A abordagem colaborativa e dinâmica para configuração, em contraste ao fluxo de atividades orquestrado de forma pré-definida, propõe uma metáfora de gerenciamento das decisões de maneira interativa e dinâmica. Assim, *stakeholders* podem engajar numa atividade de configuração colaborativa onde os esforços repetitivos requeridos para garantir a satisfatibilidade dos requisitos do cliente, sem violar as restrições de configuração, são realizados por agentes autônomos. Portanto, propomos uma solução baseada no conceito de *assistente pessoal* [18], que é um agente autônomo trabalhando com o *stakeholder* no mesmo ambiente de configuração. O assistente pessoal não é provido como um meio de resolver a configuração, mas para assistir o *stakeholder* enquanto ele está realizando seu trabalho.

O trabalho apresentado nesta dissertação engloba uma arquitetura de

configuração base que suporta a comunicação entre várias instâncias da ferramenta de configuração e duas estratégias: (i) uma para ajudar os *stakeholders* no raciocínio local e distribuído das restrições do modelo de *features*; (ii) outra para coordenar a integração das decisões interativas. Portanto, as principais contribuições deste trabalho são duas. Primeiro, abordamos os problemas em abertos de configuração colaborativa de produtos, provendo uma abordagem de orientação assistida do usuário fundada em dois conceitos principais: *Planos de Ações de Recuperação* e *Estado de Configuração Global*. Segundo, é apresentada a ferramenta *Libertas*, que valida nossa abordagem mostrando que o uso de agentes pró-ativos, além de acelerar o projeto e implementação da abordagem, tornam possível a configuração colaborativa de uma maneira dinâmica e interativa, auxiliando, de fato, os *stakeholders* a produzirem uma configuração válida.

1.3 Objetivos

Este trabalho tem como objetivo principal propor e desenvolver uma abordagem que apresente uma solução para os problemas em aberto de configuração colaborativa de linha de produtos de software em cenários distribuídos. Além disso, o trabalho tem os seguintes objetivos específicos:

- Propor, projetar e implementar uma ferramenta que ponha em prática os conceitos apresentados pela a abordagem proposta.
- Desenvolver cenários de configuração como forma de verificação da abordagem e da ferramenta proposta.

1.4 Organização do Texto

Essa dissertação está organizada em 6 capítulos.

O Capítulo 2 apresenta uma revisão da literatura sobre os conceitos relacionados ao tema abordado nessa dissertação. Uma breve apresentação sobre linhas de produtos de software é dada inicialmente, seguida por um resumo e discussão dos principais trabalhos relacionados que abordam o tema de processo de configuração colaborativo de produtos: (i) *Configuração em Estágios*; (ii) *Fluxo de atividades para Configuração de* features; e (iii) *Coordenação em Configuração Colaborativa de Produtos*. Por fim, são brevemente apresentados os conceitos de sistemas multiagentes aplicados no trabalho.

O Capítulo 3 descreve a abordagem de configuração colaborativa e dinâmica de produtos de software. Neste capítulo são detalhados os conceitos

criados e utilizados para abordar o problema. Exemplos para ilustrar a abordagem e facilitar o entendimento também são apresentados.

O Capítulo 4 descreve a ferramenta que foi construída para auxiliar na avaliação e validação da abordagem. São também detalhadas a arquitetura, as tecnologias utilizadas e a forma de implementação.

O Capítulo 5 descreve os dois cenários ilustrativos que são utilizados para avaliar a abordagem: (i) um cenário que uma empresa oferece uma linha de produtos de sistemas operacionais baseado em Linux; e (ii) um cenário com uma linha de produtos de web sites.

Por fim, o capítulo 6 conclui a dissertação sumarizando as contribuições, as lições aprendidas e sugerindo direções para futuros trabalhos.