1
Introdução

A evolução contínua da tecnologia digital tem sido um fator-chave para a proliferação do uso de sistemas multimídia. Recursos como dispositivos de armazenamento, equipamentos de aquisição e *codecs*, entre outros, têm se tornado disponíveis em maior escala ao usuário final, com uma qualidade crescente e a custos mais acessíveis. Tal disponibilidade vem incentivando o desenvolvimento de aplicações multimídia continuamente mais complexas. Vários cenários têm sido vislumbrados, como serviços de distribuição de conteúdo (vídeo sob demanda, TV interativa), de teleconferência (ensino à distância, telediagnóstico) e de comércio eletrônico, e certamente novos cenários deverão surgir futuramente.[1] Esse crescimento em variedade e complexidade de recursos e aplicações é evidenciado pela infinidade de produtos disponíveis no mercado. O interesse de órgãos internacionais e consórcios de empresas em padronizar esses produtos mostra que essa tendência não é meramente sazonal (Int'l Organisation for Standardisation, 2004).

Nesse contexto, é evidente que o desenvolvimento de sistemas de comunicação que atendam a requisitos impostos por cenários como os descritos acima – e que sejam capazes também de se adequar a requisitos em cenários futuros – é fundamental. Um dos grandes problemas em voga no setor de telecomunicações está justamente na diversidade desses requisitos, que variam de acordo tanto com as mídias envolvidas quanto com a qualidade de serviço (*Quality of Service* – QoS) desejada por cada tipo de aplicação. A título de exemplo, enquanto a natureza isócrona das mídias de áudio e vídeo impõe requisitos relacionados ao retardo de transferência, a natureza de rajadas de mídias como imagens estáticas e texto impõe outros relacionados à tolerância a perdas de dados (Soares et al., 1995). Porém, tais requisitos de mídia podem variar conforme a aplicação em questão ser interativa ou baseada em armazenamento e

[1] O crescimento massivo no uso de redes e dispositivos sem fio bem como as novas aplicações surgidas em decorrência – por exemplo, os serviços unificados de mensagens – são talvez o melhor exemplo dessa tendência.

distribuição, por exemplo. Há ainda outros requisitos envolvendo diferentes mídias simultaneamente, como os de sincronização temporal, que tornam ainda mais complexos os cenários supracitados.

É relativamente simples para um sistema de comunicação oferecer a QoS requerida por uma aplicação específica quando recursos dedicados são usados, como nos sistemas de telefonia pública. Entretanto, o desenvolvimento contínuo da tecnologia digital na aquisição e processamento de mídias, o surgimento de novas aplicações e restrições de custo mostram o quão importante é para uma operadora de serviços de telecomunicações que o sistema de comunicação utilizado possua um esquema eficiente de compartilhamento de recursos entre diferentes aplicações e permita a implantação rápida e de baixo custo de novos serviços. Em particular, a flexibilidade no oferecimento de serviços tem se mostrado um fator de diferenciação entre as operadoras, determinando seu sucesso na manutenção ou expansão de sua base de clientes.

Nesse sentido, percebe-se no setor de telecomunicações duas fortes tendências. Em primeiro lugar, operadoras têm investido na implantação de redes IP de multi-serviço que proporcionam, além de uma redução potencial de custo, uma maior facilidade de integração de diferentes aplicações – como telefonia e WWW – e, em um contexto mais amplo, um modelo de convergência universal no acesso a informações multimídia. Em segundo lugar, fabricantes de equipamentos de rede têm procurado desenvolver soluções que dissociem cada vez mais o software responsável pelo controle desses equipamentos do hardware responsável pela transmissão de dados. As unidades de processamento de rede (*Network Processing Units* – NPUs) (Network Processing Forum, 2001) representam o estado-da-arte dessas soluções. NPUs permitem a construção de equipamentos extensivamente programáveis com custo relativamente baixo e desempenho próximo àqueles baseados em hardware especializado. O objetivo, nesse caso, é viabilizar a criação de sistemas de comunicação ao mesmo tempo eficientes e mais facilmente adaptáveis a novos serviços.

Contudo, adaptabilidade é uma característica extremamente complicada de ser provida no setor de telecomunicações, que possui requisitos bastante estritos de disponibilidade (7 dias por semana, 24 horas por dia), confiabilidade, desempenho e segurança. A definição de sistemas de comunicação que propiciem a inserção de novos serviços ou a atualização de serviços existentes sem a interrupção ou comprometimento de outros serviços em produção é uma tarefa tão

complexa a ponto de ter justificado o surgimento de um novo termo, a 'engenharia de serviços de telecomunicações', definido por Znaty e Hubaux (1998) como

> "o conjunto de métodos, técnicas e ferramentas utilizado para especificar, projetar, implementar, verificar e validar serviços que atendam às necessidades dos usuários, assim como para implantar e explorar esses serviços em sistemas de telecomunicações existentes ou futuros".

Znaty e Hubaux ressaltam a importância de três elementos em sistemas de comunicação com algum grau de adaptabilidade:

1. **Infra-estrutura flexível de comunicação**: uma interconexão de equipamentos que oferecem interfaces bem definidas para programação de serviços;
2. **Arquitetura de rede**: uma estrutura que objetiva controlar a execução de serviços na infra-estrutura de comunicação, de modo que requisitos específicos – por exemplo, de QoS – sejam satisfeitos;
3. **Ambiente de criação de serviços**: uma plataforma, similar a ambientes de desenvolvimento de software, especializada para a especificação, projeto, implementação, verificação e validação de serviços.

Na busca por arquiteturas de rede mais adaptáveis, vários grupos têm centrado seus esforços em pesquisas na área de redes programáveis (Campbell et al., 1999). A diversidade de modelos de hardware para NPUs encontrada no mercado (Agere Systems, 2004; Intel Corporation, 2004; Int'l Business Machine Corporation, 2004) vem abrindo maior espaço para o desenvolvimento dessas pesquisas.

A complexidade inerente ao software de arquiteturas de redes programáveis – especialmente em infra-estruturas de comunicação idiossincráticas como as providas pelas NPUs – cria uma nova perspectiva na área de engenharia de serviços. A premissa desta tese é que projetistas e programadores de sistemas de comunicação adaptáveis podem se beneficiar ainda mais da aplicação de conceitos e princípios da área de engenharia de software que organizem os processos presentes em um ambiente de criação de serviços.

1.1. Posicionamento da tese

Embora já haja um corpo significativo de pesquisa em ambientes de criação de serviços, esta tese é motivada pela carência de soluções integradas de projeto e desenvolvimento de software para sistemas de comunicação adaptáveis, particularmente para redes programáveis. Nesse sentido, a tese presente lida com questões gerais relacionadas ao projeto e implementação de mecanismos de adaptação de software em sistemas de comunicação, bem como questões mais específicas, como provisão de QoS. Para posicionar melhor a tese, esta seção apresenta uma visão geral do estado-da-arte do processo de criação de serviços, juntamente com alguns comentários referentes às questões citadas acima.

1.1.1. Técnicas de descrição formal

Um dos focos principais da área de métodos formais é dar suporte à construção de software complexo de maneira correta e eficaz. Em geral, essa área advoga a utilização de técnicas de descrição formal (*Formal Description Techniques* – FDTs) para a especificação e projeto de sistemas de software. FDTs permitem representar o 'comportamento' de um sistema em um nível abstrato, independente da implementação real do mesmo. FDTs provêem assim uma base para a análise das propriedades mais importantes do sistema antes mesmo de sua construção. FDTs podem ser usadas também para verificar e validar continuamente a implementação de um sistema, o que é particularmente interessante no caso desse sistema ser adaptável. Além disso, métodos de síntese baseados em refinamentos sucessivos podem ser desenvolvidos com o intuito de ajudar projetistas a gerarem implementações reais a partir de especificações em uma FDT.

Várias FDTs foram definidas no contexto de sistemas de comunicação (Hoare, 1985; Int'l Organisation for Standardisation, 1989a, 1989b; Int'l Telecommunications Union, 1992b). No entanto, essas FDTs não levam em conta três características que hoje são fundamentais em praticamente todo sistema distribuído: (i) o uso de técnicas de desenvolvimento de software orientadas a objetos (*Object-Oriented Software Development* – OOSD) ou a componentes

(*Component-Based Software Development* – CBSD), (ii) a possibilidade de reconfigurações dinâmicas e (iii) a existência de requisitos de QoS, segurança etc. associados aos serviços providos pelo sistema.

Propostas de novas FDTs e extensões a FDTs existentes foram desenvolvidas (Robles et al., 2001; Milner, 1999; Agha et al., 2001; Sinnott e Hogrefe, 2001) com o intuito de prover as características supracitadas. Em algumas propostas, uma única FDT é usada para descrever todos os aspectos de um sistema. Se por um lado o uso de uma única FDT facilita a verificação das propriedades de um sistema e possibilita a construção de métodos de síntese mais simples, por outro lado o projetista é obrigado a embutir aspectos "não-funcionais" – QoS, comunicação de grupo, segurança etc. – nos meandros de uma especificação funcional. Isso pode, por exemplo, tornar mais complexo o projeto de um sistema adaptável a novos requisitos de QoS. Outros trabalhos (Blair e Blair, 1999) propõem o uso de FDTs distintas para representar aspectos funcionais e não-funcionais. Essa separação de aspectos permite uma maior liberdade em se escolher FDTs apropriadas para cada aspecto distinto a ser especificado, bem como dá margem a especificações mais simples. Essas características são particularmente interessantes para a especificação de sistemas sujeitos a adaptações.

De modo geral, apesar de propiciarem a verificação, validação e síntese de sistemas, as FDTs são tão genéricas que podem tornar a especificação de um sistema excessivamente complexa e custosa. Em geral, uma FDT permite no máximo o reuso de estruturas de granularidade muito pequena – componentes e objetos. Não há um arcabouço conceitual sobre o qual descrições arquiteturais reusáveis mais complexas, como *frameworks* e *design patterns* (vide Seção 1.1.3), possam ser definidos. Isso não só contribui para um baixo reuso de descrições arquiteturais por parte dos projetistas, como também pode comprometer o entendimento dos programadores no que se refere à arquitetura do sistema como um todo e aos serviços por ele providos.

1.1.2. Linguagens de descrição de arquitetura

Outra abordagem de especificação e projeto de sistemas, essa provinda da área de engenharia de software, consiste em usar notações declarativas para

descrever arquiteturas de software. Usando tais notações, em geral chamadas de 'linguagens de descrição de arquitetura' (*Architecture Description Languages* – ADLs), o projetista pode abstrair detalhes acerca do comportamento do sistema e apresentar ao programador uma visão global desse sistema de modo simples e preciso. Basicamente, ADLs permitem decompor a arquitetura de um software em entidades computacionais (componentes), interações entre essas entidades (conectores) e composições dessas entidades e interações (configurações). Em geral, a expressividade de uma ADL pode variar consideravelmente (Medvidovic e Taylor, 2000), indo desde aquelas que descrevem simplesmente os relacionamentos de interação e composição entre componentes e conectores de um sistema, até aquelas que descrevem toda a organização do mesmo, incluindo informações acerca do comportamento de seus elementos. Adicionalmente, algumas ADLs permitem a especificação de 'estilos arquiteturais' (Monroe et al., 1997) – idiomas que caracterizam conjuntos de arquiteturas compartilhando determinadas propriedades estruturais e semânticas. Em geral, esses conjuntos são representados por um vocabulário de elementos arquiteturais (componentes, conectores e configurações) e restrições nas combinações entre esses elementos.

De modo geral, as ADLs oferecem uma série de vantagens aos projetistas, dentre as quais: (i) melhorar o reuso de estruturas comuns entre sistemas, ou mesmo entre arquiteturas, especialmente quando a ADL permite a especificação de estilos arquiteturais, (ii) guiar a implementação dos sistemas, (iii) descrever a evolução dos sistemas e (iv) possibilitar uma avaliação prévia de aspectos arquiteturais dos sistemas, ou mesmo comportamentais, quando em conjunto com a notação declarativa são usadas também FDTs.

Vários trabalhos definem ADLs para a especificação de sistemas distribuídos, em geral focalizando diferentes problemas (Luckham e Vera, 1995; Magee e Kramer, 1996; Allen, 1997). Todas essas ADLs oferecem abstrações similares àquelas de OOSD ou CBSD e dão suporte à especificação de arquiteturas dinâmicas. Porém, aspectos não-funcionais presentes em sistemas de comunicação, em especial aqueles ligados à gerência de recursos e provisão de QoS, não são geralmente tratados.

Outras linguagens têm sido definidas levando em consideração aspectos não-funcionais (Issarny e Bidan, 1996; Bellissard et al., 1998; Durán-Limón, 2001; de Souza e Cunha, 2003; Rosa et al., 2004). Todavia,

limitações dessas ADLs – algumas delas levantadas em (Gomes et al., 2001b) – podem ser destacadas. Por exemplo, Durán-Limón (2001) define uma arquitetura muito simples e estática de controle de QoS, que não permite ao projetista representar cenários de oferecimento de serviços envolvendo a orquestração de recursos em múltiplos subsistemas – sistema operacional, subsistema de rede etc. Além disso, em vários dos trabalhos mencionados o projetista não tem como comunicar explicitamente ao programador do sistema quais pontos são sujeitos a adaptações e quais pontos devem ser mantidos inflexíveis.

1.1.3. *Frameworks*, *patterns* e linguagens de domínio específico

Outro conceito provindo da área de engenharia de software que tem sido extensivamente aplicado no contexto de sistemas de comunicação é o de *frameworks* de software. *Frameworks* de software podem ser vistos como coleções de componentes, objetos ou fragmentos de especificações que capturam decisões de projeto comuns a sistemas de software de um domínio particular. Em geral, várias partes de um *framework* não podem ser previstas, sendo deixadas incompletas. Essas partes, chamadas por Pree (1995) de '*hot spots*', permitem a definição de sistemas ou especificações adaptáveis a situações distintas (plataformas, estratégias etc).

Diversos trabalhos aplicam variações do conceito de *frameworks* na área de sistemas de comunicação. Em (Gomes, 1999; Gomes et al., 2001a; Rodrigues, 1999), *frameworks* de software que permitem o projeto de arquiteturas adaptáveis de gerência de QoS e comunicação de grupo são modelados utilizando a notação UML (Rational Software Corporation, 1997). Outros trabalhos (Kolberg et al., 1999; Logean, 1999; Dietrich, 1999) propõem a combinação do uso de *frameworks* com especificações formais, de modo que sistemas de comunicação e serviços criados a partir dos *frameworks* possam ser verificados por meio das especificações formais correspondentes a eles. Szyperski (2002) define outra linha, não restrita a sistemas de comunicação, em que o conceito de *framework* é atrelado à noção de entidades de software que controlam, com base em regras específicas de domínio, possíveis adaptações em tempo de execução de sistemas baseados em componentes. Essa linha tem sido seguida em algumas

propostas ligadas a plataformas de *middleware* para sistemas distribuídos (Blair et al., 2001).

Frameworks de software são geralmente definidos tendo em vista domínios de aplicação específicos e, portanto, não possuem a mesma abrangência das FDTs e ADLs. A criação ou mesmo adaptação de um *framework* para um novo domínio pode demandar do projetista um tempo inaceitável em diversas situações – por exemplo, no contexto da engenharia de serviços. O conceito de *design patterns* (Gamma et al., 1995) surgiu como uma forma de ajudar projetistas a reduzir o tempo de modelagem de *frameworks* de software, aplicando estruturas de projeto já utilizadas em outros *frameworks*. *Design patterns* são especialmente úteis para modelar *hot spots*, porém seu uso está muito mais voltado ao modo como os *hot spots* são implementados em um *framework* do que com a questão mais conceitual de apontar esses *hot spots* e sua função dentro da modelagem. Nesse sentido, Fontoura (1999) propõe extensões à notação UML que permitem a identificação explícita de *hot spots* dentro de *frameworks*. Essas extensões permitem o desenvolvimento de ferramentas que automatizam o processo de instanciação de *frameworks* modelados com UML, por meio do "preenchimento" adequado de *hot spots*.

Outra forma de se representar *frameworks* e seus *hot spots* é por meio do uso de linguagens de domínio específico (*Domain-Specific Languages* – DSLs). DSLs oferecem, por meio de notações e abstrações apropriadas, uma expressividade focada em (e usualmente restrita a) um domínio de problema particular. DSLs têm sido extensivamente usadas no contexto de sistemas de comunicação (Basu et al., 1997; Bonachea et al, 1999; Ladd e Ramming, 1994; Klarlund e Schwartzback, 1999). Van Deursen et al. (2000) apontam, no entanto, algumas desvantagens no uso de DSLs, em comparação com linguagens e notações de propósito geral. Dentre elas, pode-se citar como mais pertinentes: (i) o custo de se projetar, implementar e manter cada DSL e (ii) a curva de aprendizado dos usuários para cada nova DSL projetada.

1.1.4. Comentários gerais

Além dos comentários específicos feitos ao final das seções anteriores, deve-se ressaltar também, dessa vez em um contexto mais amplo, a carência de

soluções ‘fim-a-fim’ que integrem o projeto e desenvolvimento de serviços, bem como seus diferentes aspectos, em sistemas de comunicação. Percebe-se claramente que, no estado atual da arte, projetistas e programadores podem ser freqüentemente expostos a uma sobrecarga cognitiva em razão do “abismo conceitual” presente entre os diferentes níveis de abstração de projeto arquitetural (representado pelas ADLs), de descrição semântica (escopo das FDTs) e de implementação propriamente dita (alvo típico dos *frameworks* de software, *design patterns* e técnicas de OOSD e CBSD). Esse abismo se torna ainda mais visível quando são levados em conta as arquiteturas de redes programáveis, os mecanismos de adaptação de software necessários nessas arquiteturas e o tratamento de aspectos não-funcionais.

Um cenário que ilustra bem as questões acima é o de um roteador programável com suporte a QoS. Idealmente, esse roteador deve ser adaptável tanto em relação às entidades de software atuando no “plano de dados” – responsáveis pelas funções de classificação, marcação e escalonamento de pacotes, entre outras – quanto em relação às entidades atuando nos planos de sinalização e gerência. Adaptações de entidades em um dos planos (por exemplo, o tratamento de novos parâmetros de QoS no plano de sinalização) estão, em geral, intrinsecamente ligadas a adaptações nos outros planos (por exemplo, o uso de uma nova política de escalonamento de pacotes no plano de dados). De que forma esse relacionamento entre adaptações pode ser expresso tanto por meio de formalismos (garantindo de antemão a integridade das operações de adaptação) quanto na implementação real do roteador (propiciando uma melhor organização da arquitetura do software envolvido) é um ponto ainda em aberto na literatura.

1.2. Objetivos da tese

O objetivo central desta tese é apresentar uma abordagem de criação de serviços para sistemas de comunicação que: (i) expresse a adaptabilidade de múltiplos aspectos nesses sistemas, (ii) exprima interdependências na adaptação desses aspectos e (iii) reduza a sobrecarga cognitiva em projetistas e programadores.

Para alcançar o objetivo acima, é proposta a aplicação de conceitos advindos das ADLs e de técnicas de CBSD de modo uniforme, abrangendo desde

especificações de alto nível de serviços até a "componentização" de processadores de pacotes atuando nesses serviços. Espera-se que a integração proposta entre esses conceitos traga os seguintes benefícios para projetistas e programadores de sistemas de comunicação adaptáveis: (i) o uso de abstrações de alto nível durante todas as fases de projeto e implementação, (ii) um suporte, potencialmente formalizado, à configuração e adaptação de software e (iii) o encorajamento no reuso de software e de estruturas de projeto.

O trabalho presente é centrado em uma linguagem de especificação de arquiteturas de sistemas de comunicação chamada 'LindaX'. O desenvolvimento dessa linguagem constitui-se na principal contribuição apresentada nesta tese. Para lidar especificamente com a questão da adaptabilidade em sistemas de comunicação, LindaX oferece construções para a descrição tanto da arquitetura desses sistemas quanto das restrições de adaptação nos mesmos.

LindaX é baseada em esquemas XML (World Wide Web Consortium, 2001) e tem como uma de suas principais características a extensibilidade. Em particular, os 'esquemas-base' de LindaX oferecem elementos essencialmente estruturais, sem descrições de comportamento associado ou de aspectos de implementação. Porém, novos 'esquemas de extensão' podem ser definidos a partir desses esquemas-base, acrescentando às descrições em LindaX informações que permitem: (i) o refinamento para diferentes FDTs e ADLs, e (ii) a síntese de configurações de sistemas e de mecanismos de controle de adaptações desses sistemas para diferentes plataformas de programação. Ferramentas desenvolvidas como parte da abordagem proposta são responsáveis por automatizar os processos de refinamento e síntese a partir de descrições em LindaX. Essas ferramentas constituem o ambiente de criação de serviços proposto na tese, chamado 'LindaStudio'.

É importante destacar que não é objetivo desta tese propor uma notação formal em LindaX. No que se refere a formalismos, a intenção desta tese é propiciar, por meio de processos de refinamento, a integração de descrições em LindaX com formalismos já existentes. Como exemplo, esta tese apresenta uma ferramenta de refinamento[2] de descrições em LindaX para descrições em Wright (Allen, 1997), uma ADL com suporte à especificação formal do comportamento de componentes e de protocolos de interação entre eles. A

[2] Essa ferramenta foi desenvolvida a partir da reengenharia de ferramentas propostas por Soares-Neto (2003a).

representação formal de LindaX e a validação de descrições diretamente nessa linguagem foram colocadas como objetivos futuros.

No que tange os processos de síntese, o foco desta tese está no desenvolvimento de ferramentas que permitam, a partir de descrições em LindaX, a geração de código para plataformas de CBSD. O objetivo aqui não é gerar código para componentes de software em si (estes devem ser implementados manualmente), mas sim para os mecanismos de composição e de controle de adaptações desses componentes. Para ilustrar essa idéia, foi desenvolvida como parte desta tese uma ferramenta de síntese para o OpenCOM (Coulson et al., 2004), uma plataforma de componentes que oferece facilidades de inspeção e adaptação de composições de componentes de modo genérico e organizado.

Para tornar mais homogênea a implementação de mecanismos de composição e de controle de adaptações em diferentes plataformas e, por conseqüência, simplificar o desenvolvimento das ferramentas de síntese, foi definido um *framework* para gerência de adaptações que modela esses mecanismos. Esse *framework* constitui-se em outra contribuição desta tese. O *framework* para gerência de adaptações é genérico em relação à linguagem de especificação em uso (LindaX é apenas uma das possibilidades), à plataforma de CBSD subjacente e às regras de adaptação que ele impõe a composições de componentes. Para isso, adotou-se uma estratégia em que a configuração de sistemas e a verificação de regras de adaptação são definidas por *scripts* interpretáveis pelos componentes do *framework*. Esses *scripts* podem ser atualizados em tempo de execução, conferindo uma alta manutenibilidade aos mecanismos modelados. Para ilustrar a aplicabilidade desse *framework*, o mesmo foi instanciado na plataforma OpenCOM. Dessa forma, a ferramenta de síntese para essa plataforma gera, na verdade, *scripts* de configuração e de verificação de regras de adaptação a partir de descrições em LindaX.

No estabelecimento dos processos de refinamento e síntese supracitados, diversas questões relacionadas ao projeto e implementação de mecanismos de adaptação de software em sistemas de comunicação devem ser tratadas em detalhe. Dessa forma, é interessante que se estabeleça um modelo de referência para a descrição desses mecanismos. Para isso, esta tese parte de um modelo pré-existente, o ‘modelo de composição de serviços’ (*Service Composition Model*) (Colcher, 1999). O modelo SCM fornece abstrações comuns para a representação

das diferentes estratégias de adaptação introduzidas pelos principais trabalhos relacionados a plataformas de *middleware*, redes ativas, redes inteligentes etc. O modelo é, portanto, um ponto de partida adequado aos propósitos desta tese.

Os esquemas-base de LindaX oferecem estruturas que são representações das abstrações do modelo SCM. Contudo, o modelo em si, conforme descrito em (Colcher, 1999), não cobre diversos aspectos necessários a uma implementação. Em particular, o modelo carece de abstrações que permitam representar de modo apropriado a associação de recursos (tempo de CPU, *buffers*, largura de banda) a atividades de computação e comunicação. Para que o mapeamento de descrições de arquitetura em LindaX para estruturas de implementação possa ser feito levando em conta o consumo de recursos por essas atividades, esta tese propõe também, como contribuição secundária, extensões às abstrações do modelo.

1.2.1. Estrutura e uso de LindaX

O principal diferencial de LindaX em relação às ADLs convencionais está no fato de que descrições de arquitetura em LindaX não são baseadas em artefatos de propósito geral. Um dos esquemas-base de LindaX oferece um arcabouço sintático para DSLs que permitem a descrição de diferentes aspectos (funcionais e não-funcionais) de sistemas de comunicação de modo mais simples, conciso e, por conseqüência, menos sujeito a erros que as ADLs. O fato das DSLs LindaX compartilharem uma mesma estrutura sintática propicia uma redução na curva de aprendizado de seus usuários, que precisam reconhecer somente a semântica específica de cada DSL criada.

O processo de criação e uso de uma DSL construída a partir do arcabouço sintático de LindaX é sumariado na Figura 1.1. Dois atores são centrais nesse processo: os ‘projetistas de DSLs’ e os ‘projetistas de arquiteturas’.

Um projetista de DSLs cria uma nova DSL implementando uma ferramenta ‘extensora’ que detém conhecimento acerca da semântica dessa DSL (passo (1) da Figura 1.1). Uma descrição de arquitetura em LindaX é associada a uma DSL específica (ou seja, a uma ferramenta extensora) por meio de uma referência, na descrição, a um ‘estilo arquitetural’ especificado em LindaX. Um estilo LindaX define os artefatos arquiteturais que podem ser usados em descrições que seguem

a DSL (o seu 'vocabulário') e as restrições impostas a essas descrições. A especificação de um estilo também é de responsabilidade do projetista de DSLs, sendo feita a partir de outro esquema-base de LindaX (passo (1')).

Um projetista de arquiteturas é responsável por descrever arquiteturas usando o arcabouço sintático provido por LindaX (passo (2) da figura). Na descrição, esse projetista deve obrigatoriamente fazer referência a um estilo, a partir do qual será possível identificar exatamente qual DSL está sendo usada pelo projetista e, conseqüentemente, qual a ferramenta extensora que provê semântica a essa descrição. Note que, com LindaX, pode-se atribuir semânticas distintas a uma mesma descrição de arquitetura se ferramentas extensoras distintas forem usadas. Em outras palavras, uma mesma descrição pode ser válida em duas DSLs diferentes. Isso depende, contudo, do vocabulário usado na descrição estar definido em ambas as DSLs.

Ferramentas de refinamento e síntese analisam as descrições feitas pelos projetistas de arquiteturas. Na retaguarda dessas ferramentas atuam as ferramentas extensoras. Assim, uma requisição de refinamento de uma descrição em LindaX para uma ADL (passo (3) da figura) passa obrigatoriamente por uma dessas ferramentas extensoras (passo (5)). A determinação da ferramenta extensora a ser usada é feita a partir da consulta a um repositório de estilos (passo (4)). A ferramenta extensora escolhida apresenta então à ferramenta de refinamento correspondente a semântica completa da descrição. Com base nessa semântica a ferramenta de refinamento é capaz de mapear a descrição em LindaX para a ADL alvo (passo (6)). O mesmo processo ocorre em relação às ferramentas de síntese de configurações e de mecanismos de controle de adaptação para uma plataforma de programação. Note que, para ser possível a comunicação entre as ferramentas de refinamento e síntese e as ferramentas extensoras, estas últimas devem oferecer uma mesma interface às primeiras. O ambiente LindaStudio define essa e outras interfaces entre as diferentes ferramentas.

Restrições descritas em estilos LindaX servem como base para a geração dos *scripts* de verificação de regras dos mecanismos modelados pelo *framework* para gerência de adaptações (passo (6') na Figura 1.1). A geração desses *scripts* a partir de restrições de estilos LindaX é demonstrada na tese através da integração entre as ferramentas de síntese e a instanciação do *framework* para a plataforma OpenCOM.

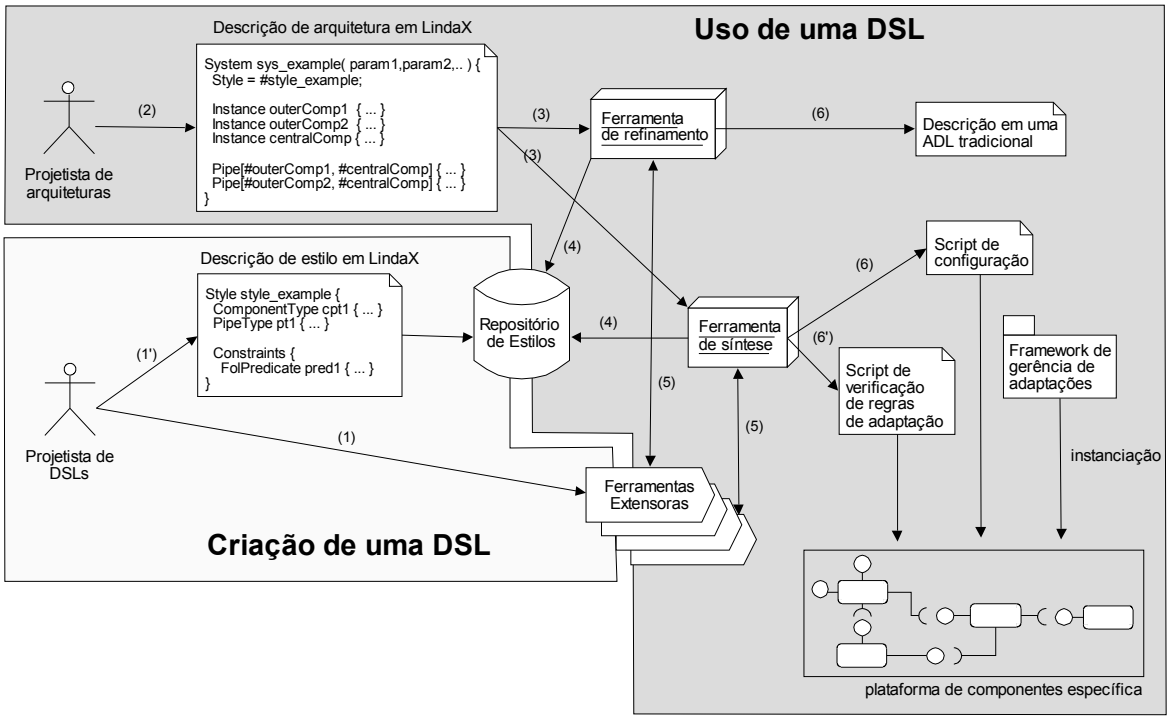

Figura 1.1. Processo de criação e uso de DSLs em LindaX.

Com o objetivo de demonstrar a expressividade da linguagem, esta tese apresenta duas DSLs LindaX que tratam de funcionalidades presentes em planos distintos de um sistema de comunicação: 'LindaRouter' e 'LindaQoS'. LindaRouter permite a especificação da arquitetura do plano de dados de roteadores programáveis (englobando, principalmente, roteadores baseados em PCs e NPUs). Os estilos LindaX associados a LindaRouter são descrições formais do *framework* de roteador informalmente apresentado em (Coulson et al., 2003). LindaQoS[3] permite a especificação de arquiteturas hierárquicas de provisão de QoS. Os estilos LindaX associados à semântica de LindaQoS são representações dos *frameworks* de provisão de QoS apresentados em (Gomes, 1999; Gomes et al., 2001a). Grande parte dos exemplos de refinamento e síntese apresentados ao longo da tese são feitos com base nessas duas DSLs e seus estilos correspondentes.

É importante reiterar que, embora tratem de funções em planos distintos, LindaRouter e LindaQoS estão intrinsecamente ligadas. Idealmente, os processos de refinamento e síntese envolvendo essas DSLs deveriam poder também ser feitos de modo integrado, pelo fato, mencionado anteriormente, de adaptações nos

[3] LindaQoS foi proposta inicialmente por Soares-Neto (2003a), tendo sido reconcebida como uma especialização do arcabouço sintático para DSLs provido por LindaX.

aspectos por elas tratados serem interdependentes. Apesar de ambas as DSLs e seus estilos correspondentes preverem esse tratamento, a integração em si das ferramentas não foi implementado, tendo sido colocada como objetivo futuro.

1.3. Organização da tese

O restante do texto está estruturado da seguinte forma. O Capítulo 2 estabelece os cenários principais de aplicação da abordagem de criação de serviços proposta nesta tese. O modelo SCM, a partir do qual a linguagem LindaX é definida, é introduzido também nesse capítulo. A estrutura e principais características de LindaX são apresentadas no Capítulo 3, tomando por base os esquemas XML que a definem. Em especial, esse capítulo identifica a importância do relacionamento entre semântica de descrições de arquitetura, estilos e ferramentas extensoras na linguagem. O Capítulo 4 tem como objetivo exercitar esse relacionamento, demonstrando a aplicação de LindaX na descrição arquitetural das funções ortogonais de encaminhamento de pacotes e de provisão de QoS. No Capítulo 5, é apresentado o ambiente LindaStudio. Dois conjuntos principais de ferramentas desse ambiente são tratados mais detalhadamente: as ferramentas de refinamento e síntese. O *framework* para gerência de adaptações, sobre o qual as ferramentas de síntese apóiam seus processos de geração de código, é introduzido também nesse capítulo. Além disso, esse capítulo apresenta estudos de caso das ferramentas de refinamento e síntese para a ADL Wright e para a plataforma de componentes OpenCOM. A seguir, o Capítulo 6 apresenta e analisa os principais trabalhos relacionados ao tema desta tese. Por fim, o Capítulo 7 tece algumas considerações finais, salientando as contribuições decorrentes deste trabalho e apresentando possíveis trabalhos futuros.