2
Ambientes de Criação de Serviços

Há mais de uma década, ambientes de criação de serviços já eram apontados como a próxima fronteira a ser explorada no setor de telecomunicações (Haselton, 1992; Int'l Conference on Intelligence in Services & Networks, 1994). O termo 'ambiente de criação de serviços', cunhado a partir do surgimento das redes inteligentes (Int'l Telecommunications Union, 1992a), tem sido atualmente utilizado com um significado bem mais amplo, referindo-se às atividades de concepção e implantação de serviços sobre quaisquer tecnologias que propiciem alguma forma de adaptação de sistemas de comunicação a novos serviços. Um dos objetivos deste capítulo é estabelecer cenários possíveis de aplicação da estratégia de criação de serviços proposta neste trabalho.

O escopo de atuação de um ambiente de criação de serviços depende do grau de adaptabilidade do sistema de comunicação alvo, isto é, da possibilidade de adaptações serem efetuadas nas diferentes fases do 'ciclo de vida' dos serviços providos pelo sistema. Para melhor definir o escopo de atuação do ambiente de criação de serviços proposto nesta tese, as diversas fases que, em geral, estão presentes no ciclo de vida de um serviço são resumidamente apresentadas na Seção 2.1. Não é o objetivo deste capítulo definir com precisão um modelo de ciclo de vida de serviços. Diversos desses modelos podem ser encontrados na literatura (Berndt et al., 1994; Grasdijk et al., 1994; Mudhar, 1994). No entanto, é necessário que sejam destacadas as atividades mais importantes presentes nesses modelos e a atuação do processo de criação de serviços em cada uma delas. Posteriormente, a Seção 2.2 apresenta uma classificação dos mecanismos de adaptação presentes nas diversas tecnologias de sistemas de comunicação quanto às fases do ciclo de vida dos serviços em que eles atuam. A Seção 2.3 enumera uma série de requisitos tecnológicos desejáveis em ambientes de criação de serviços para que esses atendam o dinamismo crescente exigido pelo setor de telecomunicações. Por fim, a Seção 2.4 apresenta um modelo que permite representar sistemas de comunicação quaisquer, bem como seus mecanismos de

adaptação, por meio de um conjunto básico de abstrações. Outro objetivo deste capítulo é mostrar de que forma esse modelo serve como referência para a definição da linguagem de especificação LindaX, bem como quais extensões ao modelo foram necessárias nesse processo.

2.1. Ciclo de vida de serviços

O ciclo de vida de um serviço é um modelo que inclui todas as fases do serviço, desde sua concepção inicial, implantação e utilização até o momento em que ele é retirado do sistema. Adicionalmente, o ciclo de vida de um serviço identifica os diversos atores e suas diferentes responsabilidades em relação a cada uma das fases do ciclo. Colcher (1999) identifica como principais atores: (i) usuários e fornecedores de serviço – referindo-se àquelas entidades relacionadas à tecnologia do serviço (aplicações, pilhas de protocolos); (ii) consumidores e operadores de serviço – entidades relacionadas a aspectos organizacionais (assinantes, empresas de telecomunicações); e (iii) projetistas e programadores[1] – designações genéricas atribuídas às entidades responsáveis pela construção do serviço. A Tabela 2.1 provê uma visão geral das diferentes fases de um ciclo de vida tradicional, com base nas classificações apresentadas por Colcher (1999), Kosmas e Turner (1997), bem como as responsabilidades dos diferentes atores em cada uma dessas fases.

É importante frisar que informações acerca do sistema de comunicação a ser utilizado devem ser produzidas durante a análise e especificação do serviço. Em geral, ambientes de criação de serviços são restritos em relação aos possíveis sistemas de comunicação alvo. São os chamados 'ambientes dedicados de criação de serviços' (Ponten et al., 1994). Porém, a característica de abertura do mercado atual de telecomunicações e a conseqüente proliferação de novas tecnologias têm demandado ambientes mais genéricos de criação de serviços. Kosmas e Turner (1997) destacam, nesse sentido, a importância do reuso de especificação e implementação para o sucesso desses ambientes. Esse ponto – cuja discussão é retomada na Seção 2.3 – é um dos principais aspectos discutidos no presente trabalho.

[1] Programadores não são explicitados na classificação de Colcher. Porém, dada a sua importância no contexto de redes programáveis optou-se nesta tese por separá-lo do papel dos projetistas.

Tabela 2.1. Ciclo de vida de serviços tradicional.

Fases	Atividades	Atores
Construção	Análise	Projetistas
	Especificação	
	Verificação	
	Implementação	Projetistas e programadores
	Validação	
	Testes	
Implantação	Instalação	Projetistas, programadores e operadores
	Ativação	Operadores
Operação	Assinatura	Consumidores e operadores
	Acesso	Usuários e fornecedores
	Interação	
	Saída	
	Cancelamento	Consumidores e operadores
Retirada	Desativação	Operadores
	Remoção	Projetistas, programadores e operadores

2.2. Classificação dos mecanismos de adaptação

As diferentes estratégias de adaptação de serviços encontradas na literatura são classificadas nesta seção segundo três parâmetros: a fase do ciclo de vida em que adaptações podem ocorrer, quais elementos do serviço são adaptáveis e quais os atores responsáveis por essas adaptações. A classificação apresentada na Tabela 2.2 é baseada em uma simplificação da classificação definida por Colcher (1999). Embora atualizada para refletir novas estratégias baseadas em computação autonômica (Kephart e Chess, 2003), a Tabela 2.2 de forma alguma abrange todas as possibilidades. Porém, ela permite identificar aquelas estratégias cobertas pelo trabalho presente.

Nota-se, a partir da Tabela 2.2, que as estratégias baseadas em redes de sinalização aberta e redes ativas se contrapõem ao ciclo de vida de serviços tradicional apresentado anteriormente, em relação aos atores atuantes na fase de operação dos serviços. Ambas as abordagens – denominadas coletivamente nesta tese de 'redes programáveis' – encaram sistemas de comunicação não só como plataformas de execução de serviços, mas também como plataformas de 'programação' desses serviços, que podem ser adaptados (reprogramados) em tempo de execução. É sobre as arquiteturas de redes programáveis que ganha mais

força a idéia introduzida por Znaty e Hubaux (1998) de uma 'engenharia de serviços de telecomunicações', em que conceitos de engenharia de software são aplicados no projeto e implementação de serviços. Nessa direção, as redes "autogeridas" – baseadas em princípios de computação autonômica – surgem como uma evolução das redes programáveis, em que a reconfiguração ou reprogramação de um serviço pode ocorrer automaticamente, em face de condições especificadas sob a forma de políticas. Redes programáveis, em suas várias formas, são o foco principal do ambiente LindaStudio, proposto nesta tese.

Tabela 2.2. Classificação das estratégias de adaptação

Estratégias	Fases de adaptação	Alvos de adaptação	Atores adaptores
Construção tradicional de serviços (ex: Rede Telefônica Pública Comutada)	Construção	Hardware e Firmware	Projetistas
Adaptação de serviços a partir de mecanismos de sinalização e gerência convencionais (ex: RSVP (Braden et al., 1997))	Operação (antes do acesso ao serviço, durante interação ou após saída do mesmo)	Configuração de software e firmware	Usuários
Redes de sinalização aberta (Campbell et al, 1999)	Operação (antes do acesso ao serviço ou após saída do mesmo)	Configuração e código do software	Projetistas e programadores
Redes ativas (Tennenhouse et al., 1997)	Operação (durante interação)	Código do software	Programadores
Redes autogeridas (Konstantinou, 2003)	Operação (antes do acesso ao serviço, durante interação ou após saída do mesmo)	Configuração e código do software	Projetistas e software

2.3. Requisitos para ambientes de criação de serviços

Crucialmente, o que se espera de um ambiente eficiente de criação de serviços é a capacidade de construção rápida de novos serviços, mesmo que considerações de gerência e mercado resultem em modificações nos requisitos de um serviço durante a construção do mesmo. Serviços não devem ser concebidos de modo isolado, mas sim tendo em vista estruturas genéricas de serviço, de modo que novos serviços possam reaproveitar essas estruturas e que modificações futuras nos mesmos possam ser incorporadas com o mínimo possível de mudanças estruturais. Além disso, a criação de serviços deve ser, o quanto possível, desacoplada de detalhes de implementação, dando a oportunidade de reuso de partes de um projeto em diferentes plataformas. Outra questão é a detecção de

inconsistências entre o comportamento esperado do serviço e o comportamento observado pelos seus consumidores. No estabelecimento de um processo de criação de serviços deve-se levar em conta, por exemplo, que a implantação de um novo serviço em uma determinada plataforma pode ocasionar interações indesejáveis com outros serviços pré-existentes (Calder et al., 2003).

Kosmas e Turner (1997) apresentam um conjunto de requisitos tecnológicos e organizacionais para ambientes de criação de serviços que, se atendidos, podem satisfazer as necessidades mencionadas acima, tornando o processo de criação de serviços mais efetivo. Dentre os requisitos tecnológicos, destacam-se quatro:

1. **Reuso**: bibliotecas de elementos básicos (componentes, objetos) – que permitam a criação de serviços por meio da composição desses elementos – devem ser estabelecidas em todas fases de construção de serviços;
2. **Arquitetura**: durante o processo de criação de serviços as características arquiteturais dos serviços devem ser explicitamente definidas pelos projetistas;
3. **Automação**: linguagens para especificação e geração de código que recebam suporte de ferramentas confiáveis são necessárias para simplificar a tarefa dos programadores e permitir a verificação tanto da especificação dos serviços quanto da conformidade da implementação.
4. **Verificação**: mecanismos que permitam a detecção e resolução de problemas na operação dos serviços, em especial aqueles decorrentes de interações indesejadas com outros serviços, devem ser providos. Nesse sentido, o uso de formalismos durante todas as fases de construção do serviço pode ser decisivo para a eficácia desses mecanismos.

Dietrich e Hubaux (2002) e Aujla et al. (1994), entre outros, fazem ressalvas quanto à aplicação de formalismos no desenvolvimento de sistemas de software, em diversos domínios. Em particular, Dietrich e Hubaux levantam a questão do dinamismo crescente do setor de telecomunicações e a conseqüente impraticabilidade de se utilizar FDTs extensivamente no processo de criação de serviços. Tanto Dietrich e Hubaux quanto Aujla et al. propõem o uso de FDTs de modo incremental aos processos de desenvolvimento tradicionalmente usados na engenharia de software. No presente trabalho, tentou-se levar em conta todos os

requisitos e considerações supracitados. O Capítulo 3 aponta de que forma esses requisitos são atendidos.

2.4. Modelo de composição de serviços

Na engenharia de software, o projeto de sistemas passa geralmente pelo uso de 'modelos' que permitam representar de maneira clara suas arquiteturas, seus comportamentos e os dados por eles manipulados. Na engenharia de serviços, modelos devem vislumbrar também, além de outros aspectos típicos de telecomunicações (QoS, segurança, gerência da comunicação, tarifação etc), a representação de estratégias de adaptação de sistemas de comunicação a novos serviços. Colcher (1999) define um 'modelo de composição de serviços' (*Service Composition Model* – SCM) cujo principal objetivo é servir de base para a análise e comparação de diferentes mecanismos de adaptação em sistemas de comunicação. Colcher deixa claro que o objetivo principal do modelo SCM não é cobrir aspectos necessários à implementação de sistemas. Porém, o modelo fornece abstrações cujo mapeamento em estruturas de implementação pode ser facilmente obtido. As entidades estruturais básicas oferecidas na linguagem de especificação proposta neste trabalho são representações dessas abstrações. Para que esse mapeamento possa ser feito de maneira precisa, este trabalho propõe também algumas extensões a essas abstrações.

O modelo SCM define duas abstrações básicas (vide Figura 2.1) que exercem o papel dos usuários e fornecedores no modelo de ciclo de vida proposto por Colcher: componentes e provedores.

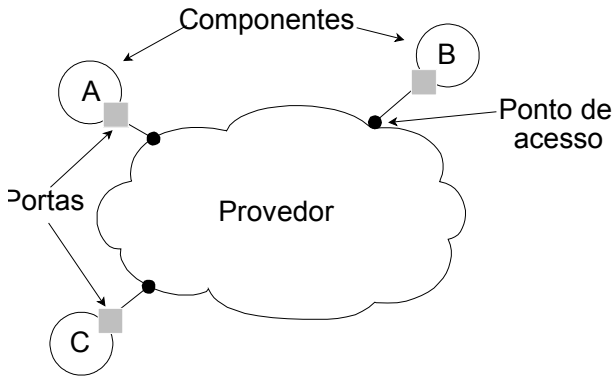

Figura 2.1. Abstrações básicas do SCM.

Componentes representam unidades de computação em sistemas de comunicação. Componentes são dotados de portas por meio das quais eles se comunicam com outros componentes. Provedores representam infra-estruturas de comunicação. Provedores oferecem pontos de acesso aos quais os componentes acoplam suas portas a fim de se comunicarem (componentes não podem ter portas diretamente ligadas). Esses 'acoplamentos' delineiam ambientes de fornecimento de serviços que podem ser tão simples quanto um processo (o provedor seria o próprio sistema operacional) ou enlace (componentes e provedores representariam, respectivamente, as interfaces de rede e o meio físico de ligação entre elas), ou tão complexos quanto um sistema de comunicação (componentes seriam, nesse caso, aplicações ou entidades de protocolo).

Componentes e provedores podem ser compostos pelo aninhamento de outras dessas entidades, segundo algumas restrições de estruturação, descritas a seguir. Primeiro, componentes e provedores internos a uma composição podem ter suas portas e pontos de acesso expostos para fora da composição. Isso permite que o constituinte interno seja acessível a partir de entidades externas à composição. Segundo, portas de componentes internos são 'mapeadas' em portas da composição. O mesmo se aplica em relação a provedores e pontos de acesso. Essas restrições implicam no tipo da entidade constituinte mapeada externamente corresponder ao tipo da entidade composta. A Figura 2.2 ilustra exemplos de composições válidas no modelo SCM (por questão de legibilidade, a notação para representação de portas e pontos de acesso não é usada na figura).

Uma das principais características do modelo SCM que o distingue de outros modelos de referência para sistemas distribuídos – como OSI e ODP (Int'l Organisation for Standardisation, 1989, 1995) – está no fato de que a estrutura aninhada de componentes e provedores permite a especificação uniforme de diferentes perspectivas de um mesmo serviço. Por exemplo, a Figura 2.2 pode ilustrar uma arquitetura em camadas na qual componentes internos representam entidades de protocolo implementando um provedor composto de nível N e se comunicando por meio de provedores internos de nível N-1. Nessa arquitetura, o termo 'serviço' adquire um significado tal qual o usado no modelo OSI. Partindo ainda da Figura 2.2, pode-se também imaginar um serviço do ponto de vista da arquitetura topológica do sistema. Nesse caso, os provedores internos que dão acesso ao provedor composto representam, por exemplo, redes de acesso sem fio e os outros provedores internos representam uma inter-rede cabeada. Pode-se ainda

privilegiar a especificação de partes do sistema, como a arquitetura de processos (Schmidt e Suda, 1993) de uma pilha de protocolos, para focar na granularidade de paralelismo da mesma. Essa visão é exemplificada pelo componente composto V da Figura 2.2, cujo provedor interno representaria explicitamente um 'ambiente de sistema local', que é considerado como fora do escopo do modelo OSI.

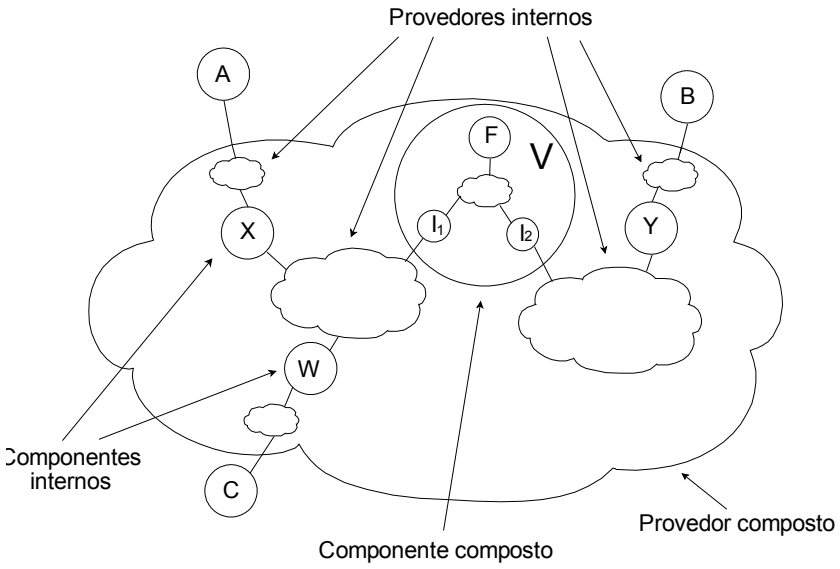

Figura 2.2. Composições arquiteturais no SCM.

O modelo SCM define também *pipes* (vide Figura 2.3). Um *pipe* captura a idéia de uma associação entre pontos de acesso de um provedor, que pode ser usada para fins de comunicação (uni- ou bidirecional) entre componentes. Conexões virtuais, invocações remotas de operações ou um simples pacote de rede são exemplos de comunicações que um *pipe* pode representar.

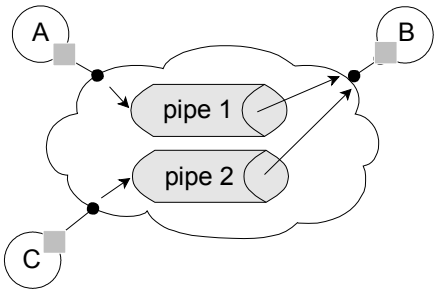

Figura 2.3. Abstração de *pipe* no SCM.

Pipes podem ser usados para representar o uso de parcelas de recursos computacionais e de comunicação internamente a um provedor. Se o provedor é composto, o *pipe* também o será, e sua constituição interna ilustrará parcelas de

recursos distribuídos pelo provedor composto sendo de alguma forma orquestrados para prover o serviço de comunicação desejado.

Por exemplo, a Figura 2.4 esboça um cenário em que os *pipes* `1` e `2` da Figura 2.3 são composições que compartilham componentes e *pipes* internos (provedores são omitidos na figura). O *pipe* `3`, por exemplo, pode representar uma conexão virtual sendo compartilhada pela multiplexação de dois fluxos de dados distintos, representados pelos *pipes* `1` e `2`. Note que o modelo SCM permite a definição de *pipes* compostos com compartilhamento de constituintes. Na tese presente, essa característica do modelo é usada também para componentes compostos.

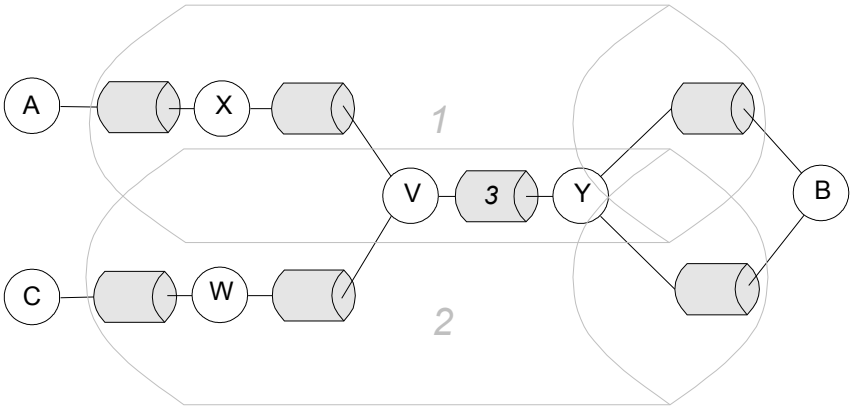

Figura 2.4. *Pipes* compostos no SCM.

A definição de um *pipe* composto é restrita à composição interna do provedor ao qual os componentes se ligam. Mais especificamente, um *pipe* composto entre dois ou mais componentes só pode ser definido dentro de um provedor composto se houver um grafo conexo de constituintes internos a esse provedor que ligue os componentes. Nesta tese, o conceito de *pipe* é utilizado, sob o ponto de vista arquitetural, de modo independente do conceito de provedor. Provedores não têm, nesse caso, função estrutural, sendo considerados como representações de escopos dos mais variados tipos – organizacionais, políticos ou administrativos.[2] Desse modo, descrições de arquitetura em LindaX constituem-se em representações de componentes e *pipes*, bem como composições entre essas entidades. Assim, restrições topológicas que um provedor incutiria em uma

[2] Note que essa noção de provedor está, a princípio, muito mais ligada a ambientes de oferecimento de serviços e sua gerência por atores "não-técnicos" do que efetivamente a infra-estruturas de comunicação. Porém, conforme será visto adiante, a noção de provedor como um "escopo" é usada nesta tese para representar também contêineres de recursos de comunicação.

arquitetura descrita segundo o modelo SCM só podem ser representadas em LindaX por meio de predicados definidos no estilo associado à arquitetura.

Um dos aspectos cruciais presentes em sistemas de comunicação que não é tratado por Colcher (1999) é a associação precisa entre entidades do modelo SCM e o consumo de recursos. O conceito de 'árvore de recursos virtuais', introduzido em (Gomes, 1999), é usado por Colcher para representar o parcelamento de recursos – distribuídos por um sistema de comunicação – entre diferentes "consumidores". Árvores de recursos permitem a definição de hierarquias de parcelamento de recursos, de modo que cada parcela (um 'recurso virtual', representado por um nó da árvore) pode ser sucessivamente reparticionada e escalonada como um recurso real. O conceito de "recurso" depende do nível de distribuição do ambiente de oferecimento de serviços, podendo se referir, por exemplo, a "processos leves" (*Lightweigth Processes* – LWPs) escalonando *threads* de usuário em um sistema operacional, ou então a uma hierarquia de multiplexação de fluxos de dados em uma rede. A Figura 2.5 ilustra o conceito. O modelo SCM não identifica, contudo, exatamente quais consumidores estão associados a quais parcelas de quais recursos.

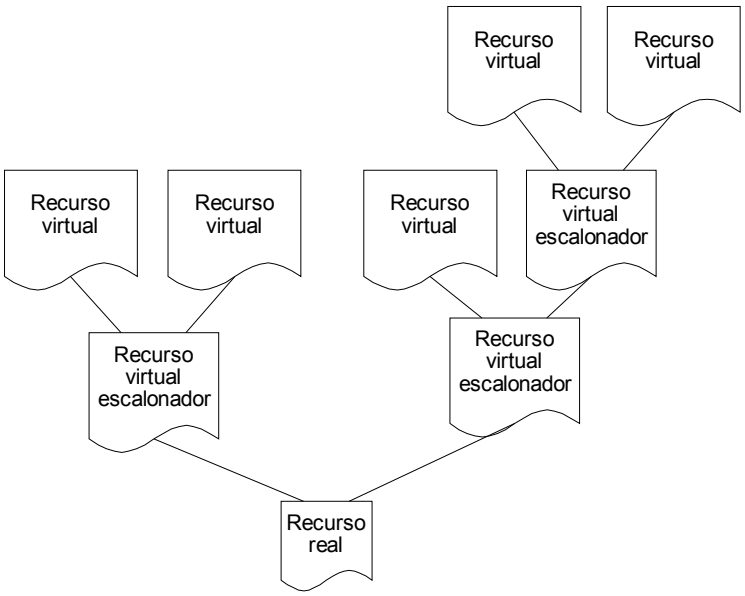

Figura 2.5. Árvore de recursos virtuais no SCM. As folhas da árvore identificam parcelas de recurso sendo utilizadas por determinados consumidores.

Pipes oferecem uma solução parcial para o problema citado acima. No contexto de sistemas operacionais, por exemplo, *pipes* não são suficientes para representar parcelas de tempo de CPU sendo alocadas a computações de

componentes. Moreno (2002) ressalta o fato de que, em sistemas operacionais, mesmo o uso de recursos de comunicação pelos *pipes* está diretamente relacionado ao uso de recursos computacionais alocados ao processamento das pilhas de protocolos.

Nesta tese, é proposta a introdução de uma nova forma de composição de componentes no modelo SCM, chamada de 'tarefa'. Tarefas permitem relacionar nós das árvores de recursos virtuais à execução de atividades de computação por parte dos componentes.

Uma única tarefa pode englobar vários componentes. Por exemplo, na Figura 2.6 – relacionada à arquitetura ilustrada na Figura 2.3 – as tarefas `AC′` e `AC″` podem representar diferentes execuções de uma atividade de computação envolvendo os componentes `A` e `C`, cada uma delas associada a um recurso próprio. Note que a característica de compartilhamento de constituintes é largamente explorada com tarefas. Conforme será visto no Capítulo 3, LindaX explora essa característica também na descrição de componentes e *pipes*.

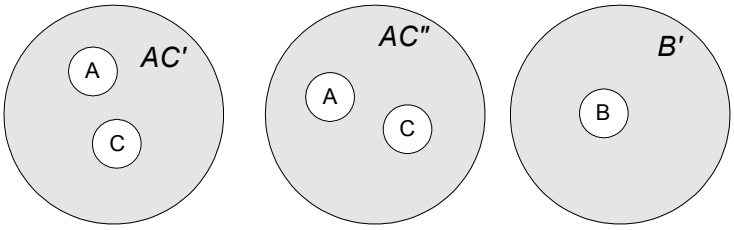

Figura 2.6. Tarefas no SCM.

Provedores são utilizados nesta tese de modo complementar a tarefas, para permitir a associação de recursos a *pipes*. Ao se utilizar os conceitos de tarefa e provedor para definir contêineres de recursos, compõe-se uma "visão de recursos" no modelo SCM, relacionada à "visão arquitetural" oferecida pelas abstrações de componentes e *pipes*. Contudo, é importante reiterar que o conceito de provedor, ao contrário do de tarefa, representa mais que simplesmente um contêiner de recursos no modelo SCM, conforme ressaltado por Colcher (1999), embora esse aspecto não seja explorado nesta tese.

Com o intuito de ajudar o entendimento dos relacionamentos entre as várias abstrações do modelo SCM, conforme expressos em LindaX, a Figura 2.7 ilustra esses relacionamentos na notação UML (*Unified Modelling Language*) (Object Management Group, 2003). Note pela figura que, em LindaX, tarefas e

provedores não podem ser recursivamente compostos. O uso de composições na visão de recursos do modelo SCM foi deixada como trabalho futuro.

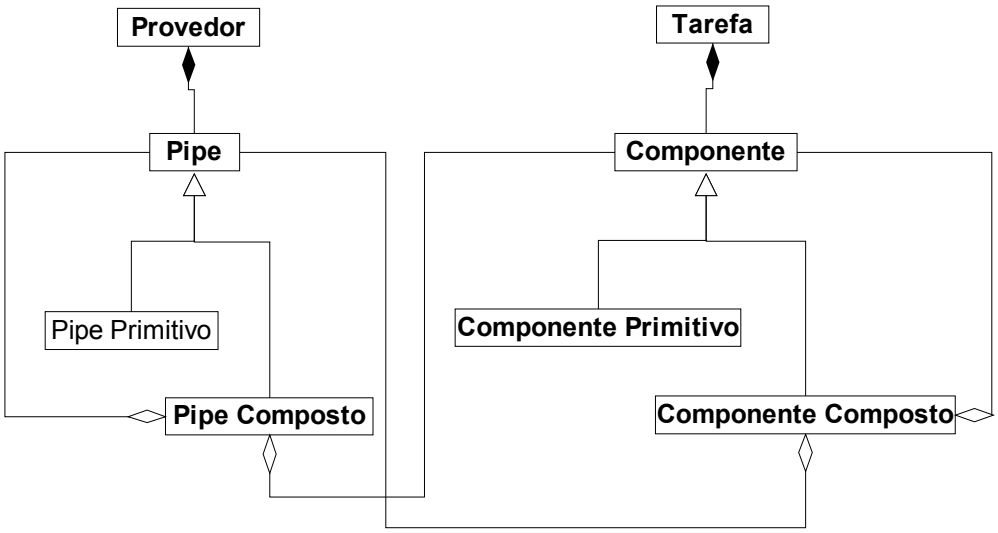

Figura 2.7. Relacionamento entre abstrações do modelo SCM.

2.4.1. Metasserviços

Componentes podem ser dotados de dois tipos de portas: de 'nível base' (*Base Level* – BL) ou 'nível meta' (*Meta Level* – ML).[3] Portas BL são portas usadas na troca usual de informações entre componentes. Portas ML provêem uma abstração genérica para mecanismos de adaptação – baseada no conceito de 'implementações abertas' (Kiczales et al., 1991) – que permite representar a possibilidade de alteração do conteúdo dos componentes (em termos de estrutura interna, comportamento etc.) que oferecem esse tipo de porta. Componentes que oferecem portas ML são denominados 'alvos' de adaptação.

Se um componente-alvo tiver uma de suas portas ML acopladas a um ponto de acesso de um provedor, ele estará oferecendo a outros componentes – chamados de 'metacomponentes' – a oportunidade de adaptá-lo (vide Figura 2.8). Metacomponentes podem adaptar componentes-alvo em qualquer nível de aninhamento. Isso permite a representação de planos de 'metasserviços' independentes do plano principal de serviços de usuário – o "plano de dados".

[3] Colcher (1999) aplica a idéia de pontos de interação ML também a pontos de acesso de provedores. Contudo, essa característica não é explorada nesta tese.

Essa dissociação entre o metasserviço e o aninhamento de componentes e provedores no nível base torna mais simples a representação dos mecanismos responsáveis pelo tratamento de aspectos não-funcionais de um sistema de comunicação, em especial aqueles relacionados à provisão de QoS fim-a-fim.

Metacomponentes podem se comunicar por meio de provedores através de portas BL, definindo 'metassistemas'. Metassistemas seguem as mesmas restrições básicas de composição dos sistemas-alvo. Exemplos típicos de metassistemas no modelo SCM são sistemas de roteamento, sinalização e gerência, como OSPF (Moy, 1998), SS7 (Int'l Telecommunications Union, 1993) e RSVP (Braden et al., 1997). Um provedor usado na comunicação entre metacomponentes pode ser totalmente independente dos provedores usados pelos componentes-alvo, como ilustrado na Figura 2.8 (caso do sistema SS7), ou os componentes do metassistema e do sistema-alvo podem compartilhar um mesmo provedor (caso dos protocolos OSPF e RSVP).

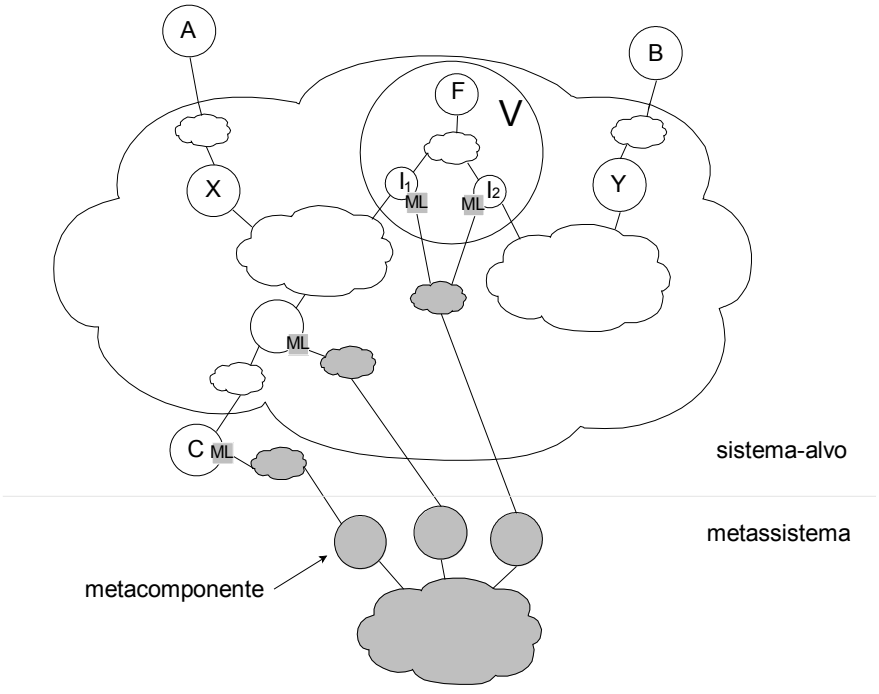

Figura 2.8. Metasserviços no SCM.

Metassistemas também podem ser alvos de adaptações. As redes de sinalização aberta são um exemplo típico, uma vez que os protocolos de sinalização dessas redes podem ser reprogramados (via protocolos de metassinalização) para lidar com novos requisitos de um aspecto, como novos

parâmetros de QoS. Colcher define o conceito de 'torres de metasserviços'[4] no modelo SCM – cuja representação é ilustrada na Figura 2.9 – para identificar esse tipo de relacionamento entre sistemas e metassistemas.

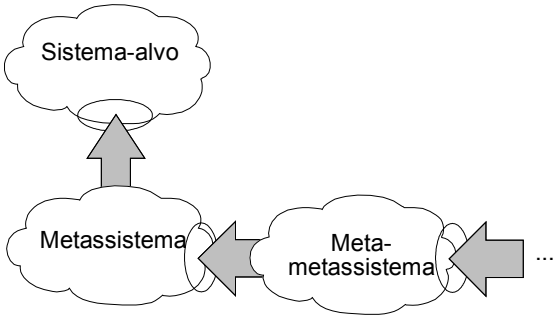

Figura 2.9. Torre de metasserviços no SCM.

2.4.2. *Frameworks* de serviços

As entidades do modelo SCM apresentadas ao longo deste capítulo formam uma espécie de "linguagem" básica para a especificação de serviços. O caráter recursivo do modelo dá às especificações baseadas nessa "linguagem" um forte potencial de reuso de projeto e de implementação. Assim, estruturas pré-montadas podem ser definidas com base no modelo para descrever aspectos comuns em sistemas de comunicação, como funções de encaminhamento, mecanismos de provisão de QoS, entre outros.

O conceito de *frameworks* é aplicado no modelo SCM para representar as estruturas pré-montadas citadas acima. Nesse caso, os *hot spots* dos *frameworks*, segundo a definição estabelecida por Pree (1995), correspondem às partes de um sistema que são adaptáveis a novos serviços. O momento dentro do ciclo de vida de um serviço em que esses *hot spots* são completados determina o grau de adaptabilidade de cada parte do sistema. Alguns *hot spots* estão relacionados à plataforma de execução em que o sistema se insere, sendo, portanto, tipicamente "preenchidos" durante a fase de construção. Outros *hot spots* podem ser preenchidos em tempo de operação, sem interrupções nos serviços em uso, como parte de um procedimento de atualização de um módulo de software no sistema, por exemplo.

[4] Conceito baseado nas "torres de reflexão" de sistemas reflexivos (Maes, 1987).

Em (Gomes, 1999; Gomes et al., 2001a) um conjunto de *frameworks* permite modelar mecanismos de provisão de QoS atuantes em qualquer tipo de provedor. Uma vez que os ambientes de sistema local de um sistema de comunicação (usando a terminologia do modelo OSI) também são representados por provedores, o modelo SCM oferece uma maneira natural de se especificar mecanismos de provisão de QoS verdadeiramente fim-a-fim, que orquestram o uso de todos os recursos computacionais e de comunicação envolvidos em um serviço (vide Figura 2.10).

Quando instanciados, os *frameworks* definidos em (Gomes, 1999; Gomes et al., 2001a) propiciam também a definição de interfaces de controle e gerência de QoS que podem ser reusadas em diferentes partes de um sistema. Por exemplo, Mota (2001) e Moreno (2002) definem instanciações desses *frameworks* em partes distintas de uma rede IP – para as arquiteturas IntServ e DiffServ e para sistemas operacionais, respectivamente. Essas implementações de mecanismos de QoS compartilham um mesmo conjunto de interfaces. Essa homogeneidade permite simplificar consideravelmente a implementação de algoritmos de orquestração entre esses mecanismos.

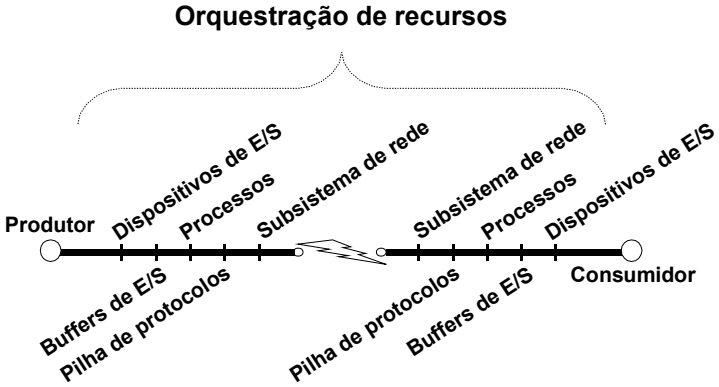

Figura 2.10. Provisão de QoS fim-a-fim (extraída de (Aurrecoechea et al., 1998)).

Colcher (1999) utiliza extensões para a notação UML propostas por Fontoura (1999) para descrever os *hot spots* dos *frameworks* ligados ao modelo SCM. Um processo desenvolvido por Fontoura permite a síntese automática de código a partir das descrições na notação UML estendida. Todavia, esse processo de síntese enfoca primordialmente o reuso de código e projeto, englobando somente adaptações ocorridas durante a fase de construção no ciclo de vida dos serviços. Apesar de permitir a representação de *hot spots* mais "dinâmicos", a abordagem proposta por Fontoura não prevê como esses *hot spots* podem ser

implementados. O ambiente LindaStudio propõe um processo de síntese que leva em conta esse tipo de *hot spot*.

2.5. Sumário

Este capítulo estabeleceu uma terminologia de engenharia de serviços, baseada no modelo SCM, a partir da qual os cenários possíveis de aplicação deste trabalho são delineados. A carência, no modelo SCM, de abstrações para a representação da alocação de recursos a atividades de computação e comunicação demandou extensões ao mesmo, bem como uma "releitura" de algumas de suas abstrações. Mais especificamente, a noção de 'tarefa' como contêiner de recursos computacionais e o tratamento do conceito pré-existente de 'provedor' como contêiner de recursos de comunicação são contribuições desta tese ao modelo.