9
Revisitando os Trabalhos Relacionados

Neste capítulo, os trabalhos relacionados, discutidos no Capítulo 2, são revisitados para que uma análise comparativa entre esses trabalhos e as soluções propostas neste documento seja realizada.

9.1.
Gerência de Recursos e Evolução Dinâmica

Os resultados experimentais apresentados nesta tese reforçam as conclusões obtidas em outros domínios de aplicação, apresentados nos trabalhos citados na Seção 2.1, sobre as vantagens da adoção de uma arquitetura baseada em componentes. Uma contribuição desta tese é a apresentação de uma arquitetura específica baseada em componentes para o domínio de sistemas de middleware declarativo para TVD interativa. Entretanto, a implementação baseada em componentes de software do Ginga-NCL foi realizada com base diretamente nos serviços providos pelo sistema operacional, sem utilizar uma infra-estrutura de software com suporte a construção de sistemas baseados em componentes, como as utilizadas nos trabalhos citados anteriormente. Essa característica pode contribuir para a redução dos requisitos de memória e processamento da solução apresentada. Uma representação comparativa entre os trabalhos discutidos na Seção 2.1 e a solução apresentada no Capítulo 3 é apresentada na Tabela 13.

	Evolução Dinâmica	Gerencia Recursos	Elimina overhead do uso de Infra-Estrutura de Software	Foco no Sincronismo de Mídias
OpenCom	✓	✓		
Fractal	✓	✓		
FlexCM	✓	✓		
Ginga-NCL	✓	✓	✓	✓

Tabela 13 – Representação Comparativa entre Trabalhos para Gerência de Recursos e Evolução Dinâmica

Como apresentado na Tabela 13, entre os trabalhos discutidos, a solução apresentada nesta tese é a única que possui como requisito a manutenção das relações de sincronismo especificadas pelo autor da aplicação. Portanto, a solução apresentada é a única que leva em conta os problemas que podem ser causados pelo retardo na carga dos componentes no sincronismo da execução de um programa, definindo uma estratégia para que se tenha o mínimo de componentes carregados na memória sem que o retardo inserido ao carregar esses componentes na memória comprometa os relacionamentos espaço-temporais especificados pelo autor de aplicações NCL.

9.2. Plano de Recuperação

Os mecanismos de recuperação de falhas apresentadas nos diversos trabalhos existentes são baseados em técnicas de recuperação proativa e recuperação reativa. A Tabela 14 apresenta uma comparação entre a solução discutida no Capítulo 4 e os trabalhos relacionados discutidos na Seção 2.2. Assim como as soluções propostas por Sousa et al. (Souza, 2007) e Fugini e Mussi (Fugini, 2006), a solução apresentada é fundamentada na combinação de técnicas de recuperação proativa com técnicas de recuperação reativa, definindo tanto uma metodologia para o rejuvenescimento de algumas partes do sistema, como mecanismos para a recuperação reativa de falhas. As técnicas de recuperação reativa são importantes, principalmente, pela possibilidade de falhas ocorridas no sistema comprometerem a apresentação de aplicações NCL.

	Rec. Proativa	Rec. Reativa	Aplicações Resilientes	Sincronismo de Mídias
Huang et al.	✓			
Castro e Liskov	✓			
Sousa et al.	✓	✓		
Fugini e Mussi	✓	✓		
Ginga-NCL	✓	✓	✓	✓

Tabela 14 – Representação Comparativa entre Trabalhos de Recuperação de Falhas

Para prover resiliência à apresentação de aplicações interativas o plano de recuperação proposto deve ser capaz de monitorar também a consistência de cada apresentação realizada, de forma similar à proposta de Fugini e Mussi (Fugini, 2006), que monitora a consistência dos dados de entrada das aplicações. No entanto, na proposta de Fugini e Mussi (Fugini, 2006), esse monitoramento objetiva a resiliência nos serviços que as aplicações usam, diferentemente do plano proposto nesta tese, que deve reagir às inconsistências na apresentação das aplicações de forma a torná-las resilientes.

Como apresentado na Tabela 14, entre os trabalhos discutidos, o trabalho proposto nesta tese é o único capaz de prover resiliência tanto no sistema quanto nas aplicações que utilizam o sistema. Para tornar resilientes as aplicações NCL, a compreensão do sincronismo de mídias deve ser definida como um dos requisitos do plano de recuperação. Isso porque uma aplicação interativa de TVD pode conter objetos de mídia de diferentes tipos, como texto, imagem, vídeo, áudio etc. e, em comum, esses objetos devem obedecer às relações de sincronismo especificadas pelo autor da aplicação, tanto em relação ao tempo de exibição, quanto ao espaço de exibição nos dispositivos utilizados para apresentação. Mais uma vez, o requisito da manutenção do sincronismo de mídias é um diferencial da solução proposta.

9.3. Comandos de Edição

Nos trabalhos relacionados, discutidos na Seção 2.3, para as aplicações de TVD desenvolvidas na linguagem Java, certo grau de edições ao vivo pode ser alcançado criando observadores para monitorar a chegada dos eventos DSM-CC.

Nas aplicações baseadas em XHTML soluções semelhantes são empregadas para as tarefas de edição ao vivo usando funções de scripts imperativos, normalmente na linguagem ECMAScript. O recebimento dos eventos de sincronismo pelas aplicações XHTML só é possível através de mapeamento de eventos DSM-CC em eventos DOM (ETSI, 2010). A partir de um evento DOM, funções *ECMAScript* devem ser utilizadas para realizar as operações do comando de edição. Assim, em ambos os casos, procedimentos imperativos são necessários, exigindo programadores especialistas, conhecedores de bibliotecas e de técnicas de programação específicas, sendo, portanto, mais propensos a erros de programação, uma vez que todo o controle é passado ao programador. Nenhum suporte declarativo é oferecido para a edição de aplicações ao vivo, incluindo o seu início a partir de um instante qualquer de um serviço, ou seu início por diferentes interfaces (partes da aplicação).

O trabalho desenvolvido por Costa et al. (Costa, 2006b) é um avanço no sentido de oferecer suporte declarativo a edições ao vivo. No entanto, Costa et al. (Costa, 2006b) preocupam-se apenas na definição de comandos que alterem o escalonamento de objetos em uma aplicação hipermídia. Naquele trabalho, comandos são definidos para manipulação do grafo hipermídia temporal (HTG – Hypermedia Temporal Graph) pro meio do qual todos os planos de controle para transmissão, recepção e apresentação de objetos de mídia de uma aplicação hipermídia são derivados. Nenhum suporte, no entanto, é dado para a identificação dos recursos a serem apresentados, e nenhuma de estrutura de dados é definida para os comandos que os permitam serem enviados pelos mais diferentes sistemas de transporte. Os comandos de edição definidos nesta tese estendem os trabalhos de Costa et al. (Costa, 2006b), visando não apenas o controle de escalonamento, mas todas as demais tarefas exigidas no controle do ciclo de vida de uma aplicação (Morris, 2005).

Ao contrário dos trabalhos discutidos, a solução apresentada no Capítulo 5 faz com que o mapeamento dos eventos de sincronismo em comandos para edições das aplicações de TVD seja abstraído pela máquina de apresentação. Além disso, é oferecido ao autor das aplicações um suporte declarativo para edição das aplicações. Ou seja, o autor pode especificar entidades declarativas para adições ou alterações no documento; ou comandos simples para remoção de

entidades NCL, além da flexibilidade no controle das bases privadas e do ciclo de vida das aplicações.

9.4. Identificação de Recursos

Os mecanismos para identificação de recursos de uma aplicação nos dispositivos receptores, apresentados no Capítulo 6, são capazes de livrar os autores das aplicações de conhecer como as estruturas de dados serão transmitidas e armazenadas no cliente receptor, onde as aplicações serão exibidas, oferecendo uma abstração da localização desses recursos.

9.4.1. Identificação de Recursos do Carrossel de Objetos

Diferente das soluções apresentadas na Seção 2.4.1, não é necessário o conhecimento de estruturas complexas do carrossel de objetos, ou a necessidade da geração de uma nova versão da aplicação de TVD no ambiente do servidor, com identificadores de recursos que só funcionam no ambiente de exibição. Uma solução simples e eficiente é o envio de informações adicionais (metadados) para eliminar a necessidade de criação de novas versões da aplicação. A solução traz uma mudança de paradigma: a resolução de referências não é mais realizada no lado do servidor, mas no lado do cliente.

As informações enviadas nos sistemas citados na Seção 2.4.1 apresentam uma importante restrição. Ao definir nos descritores um diretório base para caminhos relativos e especificar que seu valor deve ser "/", caso seja o diretório raiz, as especificações estão impossibilitando o uso de referências absolutas para objetos transmitidos no carrossel. Sem alterações nos identificadores de recursos, o mecanismo não contempla a transmissão de aplicações armazenadas no ambiente de autoria que referenciem conteúdos dispostos em dispositivos de armazenamento compartilhados da rede; ou recursos dispostos em discos ou partições diferentes de uma mesma máquina, mesmo que esses recursos estejam sendo transmitidos em outro carrossel de objetos. Por exemplo, com essa restrição, é impossível desenvolver em um ambiente de autoria com o sistema

operacional Windows, uma aplicação que referencie de forma absoluta recursos em discos diferentes.

Ao contrário dos mecanismos especificados nos outros sistemas de TVD, independentemente se a aplicação interativa possui URL absoluta ou relativa, ou mesmo se o carrossel de objetos utiliza recursos de outros carrosséis, através do mecanismo proposto, a identificação de recursos é realizada nos receptores exatamente da mesma forma que no ambiente de autoria. O mapeamento para a localização física do recurso pode ser automaticamente realizado pelo middleware.

É importante ressaltar que o mecanismo descrito no Capítulo 6 é completamente compatível com os padrões utilizados nos sistemas de TVD citados na Seção 2.4.1. No entanto, atualizações na implementação dos respectivos middlewares seriam necessárias, uma vez que é fundamental a existência do módulo responsável por manter a estrutura de metadados que relaciona as informações para identificação de recursos.

9.4.2. Identificação de Fluxos Elementares

Para permitir a uma aplicação referenciar fluxos elementares, os sistemas de TVD precisam especificar URLs complexas, como discutido na Seção 2.4.2, que consistem em identificadores de fluxos elementares que fazem parte de serviços de um fluxo de transporte originado de determinada rede de TVD.

Com o objetivo desobrigar o autor das aplicações interativas de conhecer os detalhes de representação do sistema de transporte, os sistemas especificam as palavras reservadas apresentadas na Seção 2.4.2.

No entanto, as palavras oferecidas ao autor possuem importantes restrições como não poder referenciar fluxos elementares de texto e nem mesmo poder referenciar outros serviços ao especificar URLs com as palavras reservadas.

Uma solução para essas limitações foi apresentada na Seção 6.2, através das palavras reservadas definidas para aplicações NCL.

9.4.3. Identificação de Bases Temporais

Nos sistemas de TVD discutidos na Seção 2.4.3, o acesso a bases temporais para especificar o sincronismo pode ser feito de duas formas: monitorando constantemente os valores NPT recebidos, por meio de procedimentos imperativos, ou registrando os eventos de sincronismo a serem monitorados e criando observadores para monitorar a chegada desses eventos.

A primeira abordagem exige que na fase de autoria os autores conheçam os detalhes de acesso às bases temporais e toda a complexidade da manipulação de descritores NPT (ISO, 1998). Já a segunda abordagem exige que os autores se preocupem com o envio de eventos de sincronismo para alterar o comportamento da aplicação, como discutido na Seção 2.4.3.

Para evitar mecanismos complexos de monitoração de eventos a serem implementados de forma imperativa pelos autores de aplicações, a especificação do sincronismo em linguagens baseadas em XHTML e Java, geralmente divide a aplicação, em vários pequenas aplicações executadas ao longo do tempo, iniciados por meio de eventos de sincronismo disparados pelo autor. Essa abordagem, no entanto, pode ser utilizada apenas para aplicações simples e mesmo nesse caso, a semântica da aplicação original é completamente perdida, visto que em seu lugar tem-se um conjunto de exibições isoladas disparadas no tempo. Em outras palavras, a semântica da apresentação como um todo fica na cabeça do autor que passa a ser o controlador da aplicação.

A solução apresentada na Seção 6.3 explora a flexibilidade e a expressividade da linguagem NCL. NCL é uma linguagem com o escopo declarativo muito mais expressivo que XHTML, permitindo a definição do sincronismo de mídias em sua forma mais ampla, sem a necessidade do uso de scripts ou programas imperativos. Com a solução discutida nesta tese, a identificação das bases temporais e sua associação aos conteúdos a serem apresentados é realizada pelo middleware Ginga-NCL de forma totalmente opaca ao autor da aplicação, permitindo que o sincronismo espaço-temporal entre conteúdos seja realizado sem que os autores precisem se preocupar com artefatos como o NPT ou eventos de sincronismos, entidades abstraídas pela máquina de apresentação.

9.5. Ciclo de Vida das Aplicações NCL

Nos sistemas de DTV terrestre citados na Seção 2.5, o controle do ciclo de vida das aplicações é realizado através de comandos enviados na estrutura denominada AIT (ETSI, 2010). Aplicações podem ser iniciadas, paradas ou abortadas, imediatamente ao recebimento dos comandos. Naqueles sistemas não há uma forma de agendar o início de uma aplicação para um certo instante do tempo a partir da sintonia de um serviço. No entanto, uma aplicação pode começar e esperar por um evento DSM-CC para sua continuação, mas todo esse controle é realizado de forma imperativa, é tarefa do autor da aplicação fazer sua especificação e, portanto, é sujeito a erros, a ineficiência, além de exigir um autor especialista em linguagens imperativas.

Ressalta-se também que uma aplicação não pode ser pausada por um comando da AIT. Novamente, caso se queira essa funcionalidade, mecanismos complexos desenvolvidos de forma imperativa devem ser programados pelo autor da aplicação, disparados por um evento de sincronismo DSM-CC.

O controle do ciclo de vida de aplicações NCL consiste em uma abordagem mais flexível, como apresentado no Capítulo 7, com alternativas para que as facilidades dos comandos de edição NCL sejam exploradas. Através das edições de entidades NCL (veja Capítulo 5), além de resolver as limitações impostas pelo controle do ciclo de vida por comandos AIT, a solução oferece ainda um controle mais granular das aplicações.

Outro ponto que deve ser ressaltado na solução apresentada no Capítulo 7, é que no Ginga-NCL, através dos parâmetros *nptBaseId* e *nptTrigger* do comando *startDocument*, independente do momento em que um telespectador inicia a recepção da transmissão, e das mudanças de sintonização realizadas durante essa recepção, a qualidade do sincronismo é sempre preservada.

9.6. Sistemas de Transporte

As estruturas de dados necessárias para o transporte de aplicações e de comandos para controlar o comportamento dessas aplicações usadas nos diversos sistemas de TVD terrestre possuem as limitações discutidas na Seção 2.6.

Estruturas de dados mais leves para transporte de comandos de edição, de arquivos de dados e de metadados para suporte à identificação de recursos são apresentadas no Capítulo 8. Essas estruturas, ao contrário das outras propostas, não estão amarradas a nenhum sistema de transporte específico e, portanto, não carregam as limitações dos sistemas de transporte discutidos na Seção 2.6. A forma como as estruturas propostas podem ser mapeadas nos sistemas de transporte dos diversos sistemas de TVD, tanto terrestre quanto IPTV, também são apresentadas no Capítulo 8.