Tradução de Trechos do Relatório Técnico CMU/SEI-2007-TR-005

WOOD, W. "A Practical Example of Applying Attribute-Driven Design (ADD), Version 2.0," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report CMU/SEI-2007-TR-005, 2007.

Tradução realizada por Gabriela Teixeira Vieira

2. Definição de sistema

Esta seção descreve o sistema básico cliente-servidor do nosso exemplo, o Sistema Controlador dos Trilhos do Metrô de Nova York. É um sistema cliente-servidor que possui requisitos de disponibilidade e tolerância a falhas. Nós estamos projetando sua arquitetura em termos de 3 requisitos arquiteturais: requisitos funcionais, restrições de projeto e requisitos de atributo de qualidade.

2.1 REQUISITOS FUNCIONAIS

A Figura 2 descreve uma visão geral funcional do nosso exemplo de cliente servidor:

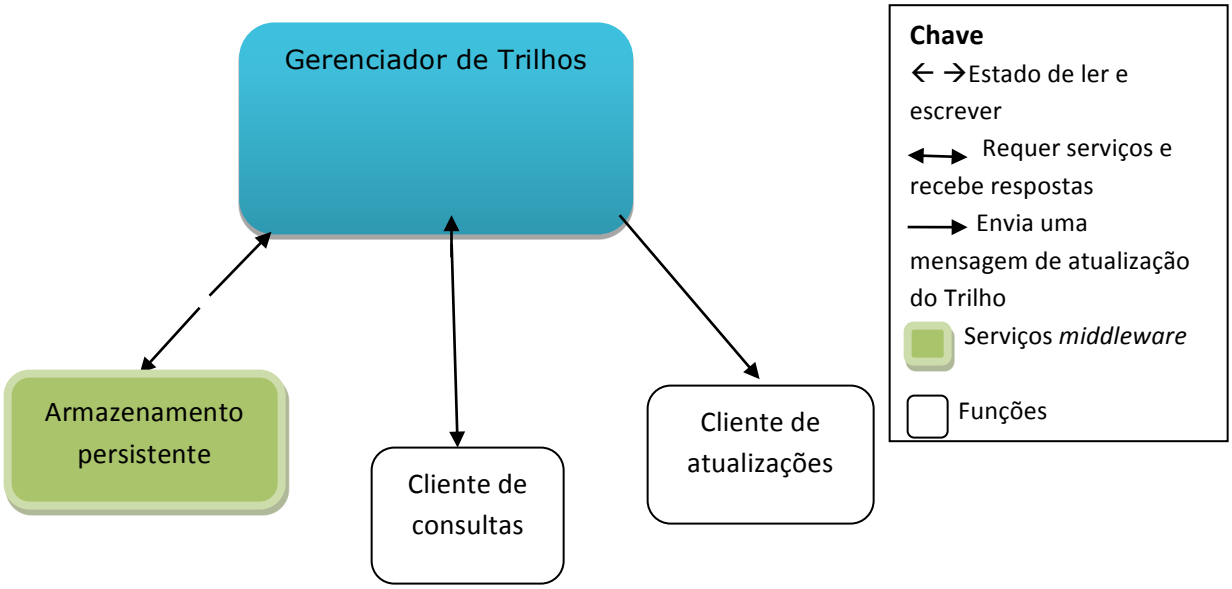

Figura 2: visão geral funcional

O servidor do Gerenciador de Trilhos provê serviços para dois tipos de clientes:

- **Clientes de atualizações:** enviam atualizações periodicamente para o Gerenciador de Trilhos. O Gerenciador de Trilhos pode tolerar algumas perdas ocasionais de atualizações, especialmente em algumas condições transitórias causadas por falhas de equipamento. Todos os clientes de atualizações fazem atualizações de 1 em 1 segundo. O Gerenciador de rastreabilidade consegue se recuperar de 2 sinais perdidos de atualizações quando ele recebe um terceiro sinal de atualização. Se mais de dois sinais forem perdidos o operador terá de auxiliar o Gerenciador de Trilhos no processo

de recuperação. Em outras palavras: se uma falha acontecer, o processamento precisa ser reinicializado antes de se passarem 2 segundos para que se evite a intervenção do operador.

- **Clientes de consulta**: esses clientes operam esporadicamente e devem receber exatamente uma única resposta para as suas consultas. Clientes de consulta podem requisitar frequentemente pequenos volumes de dados, e outros podem requisitar grandes volumes de dados ocasionalmente. O tempo de resposta máximo de consultas deve ser menor do que o dobro do tempo normal de resposta de uma consulta.

2.2. RESTRIÇÕES DE PROJETO

Existem 3 restrições de projeto para o sistema:

1- Restrições de capacidade: todos os processadores devem ter 50% de processamento e de capacidade de memórias livres para entrega e a LAN deve ter 50% de vazão de capacidade livre. Existem 100 clientes de atualização e 25 clientes de consultas. Para as estimativas de tempo pode-se assumir 100 atualizações e 5 consultas por segundo.

2- Serviço de armazenagem persistente: este serviço deve manter uma cópia do estado que é registrada pelo menos uma vez por minuto pelo Gerenciador de Trilhos. Se todas as réplicas do Gerenciador de Trilhos falharem, uma reinicialização pode começar do arquivo de verificação.

3- Duas réplicas: para satisfazer os requisitos de disponibilidade e confiabilidade, um estudo de Disponibilidade, Confiabilidade e Manutenibilidade foi realizado e o Gerenciador de Trilhos e elementos persistentes armazenados devem todos ter duas réplicas operando durante circunstâncias normais.

2.3 REQUISITOS DE ATRIBUTOS DE QUALIDADE

As partes interessadas no sistema concordam em três cenários de atributos de qualidade que descrevem as várias respostas dos sistemas a falhas. Esses cenários são descritos nas tabelas 1-3.

Tabela 1: Cenário de Atributos de Qualidade: recuperação rápida

Elemento	**Afirmação**
Estímulo	Um componente de software ou *Hard*ware do Gerenciador de Trilhos falha.
Fonte de estímulo	Uma falha ocorre em um componente de software ou *Hard*ware do Gerenciador de Trilhos.
Ambiente	Diversos clientes de software usam esse serviço. Na ocasião da falha, o componente pode estar atendendo a vários clientes concorrentemente.
Artefato	Gerenciador de Trilhos.
Resposta	Todas as consultas feitas por clientes antes e durante a falha devem ser atendidas. Pedidos de atualização podem ser ignorados por até dois segundos sem perda de precisão.
Medida da resposta	A réplica secundária deve ser promovida a primária e começar a

	processar pedidos de atualização em até dois segundos após a ocorrência da falha. Quaisquer respostas a consultas que estão a caminho (ou que foram feitas próximas ao momento da falha) devem ser respondidas em até três segundos (em média).

Tabela 2: Cenário de Atributos de Qualidade: recuperação lenta

Elemento	**Afirmação**
Estímulo	Um componente de software ou *Hard*ware do Gerenciador de Trilhos falha quando não tem serviço de backup disponível.
Fonte de estímulo	Uma falha ocorre em um componente de software ou *Hard*ware do Gerenciador de Trilhos.
Ambiente	Uma única cópia do Gerenciador de Trilhos está fornecendo serviços e falha. Um processador de reserva que está disponível não contém uma cópia desse componente. Uma cópia do componente está disponível no armazenamento persistente e pode ser transferido para o processador de reposição através da LAN.
Artefato	Gerenciador de Trilhos.
Resposta	Os clientes são informados que o serviço se tornou indisponível. Uma nova cópia do serviço é iniciada e se torna operacional. O estado do componente na reiniciação pode diferir daquele do componente falho, mas não por mais de um minuto. Os clientes são informados que o serviço está disponível para receber sinais de atualização. Para alguns Trilhos, as novas atualizações podem ser automaticamente correlacionadas aos velhos Trilhos. Para outros, um administrador ajuda nessa correlação. Novos Trilhos são começados quando necessário. Os clientes são então informados que o serviço está disponível para novas consultas.
Medida da resposta	A nova cópia estará disponível dentro de 3 minutos.

Response measure The new copy is available within three minutes.

Tabela 3: Cenário de qualidade: reinicialização

Elemento	**Afirmação**
Estímulo	Uma nova réplica é iniciada como standby.
Fonte de estímulo	O Gerenciador de recursos do sistema inicia em standby.
Ambiente	Uma única réplica está atendento a requisições sob condições normais. Nenhuma outra réplica está presente.
Artefato	Nova réplica do Gerenciados de Trilhos.
Resposta	A inicialização da nova réplica tem impacto transiente sobre as requisições de serviço que duram menos de dois segundos.
Medida da resposta	A inicialização da nova réplica tem impacto transiente sobre as requisições de serviço que

	duram menos de dois segundos.

3 APLICANDO ADD

São necessárias pelo menos 2 repetições através do processo ADD para desenvolver uma arquitetura que satisfaça os requisitos arquiteturais de um sistema proposto. Como mostra a figura 1 na página 2, a etapa 1 (descrita abaixo) é conduzida apenas para garantir que a informação que você tem sobre os requisitos é suficiente. Nós não discutimos as etapas do projeto na primeira repetição já que o nosso interesse primário é na tolerância a falhas. A equipe de arquitetura criou as visões arquiteturais mostradas na figura 3 e esboçadas na seção 3.2.
Um elemento do serviço de tolerância as fallhas está incluso nessa visão. Esse elemento é atribuído a um especialista em tolerância a falhas por projetar em paralelo com alguns outros projetos (por exemplo serviços de *start-up*), os quais nós não descrevemos aqui. Os especialistas em tolerância a falhas prosseguem com a segunda repetição do método ADD e decompõem diferentes aspectos dos serviços de tolerância a falhas.

3.1 ETAPA 1 DO MÉTODO ADD: CONFIRME QUE HÁ REQISITOS DE INFORMAÇÃO SUFICIENTES

A Seção 2 (ver pag. 5) lista os requisitos para o exemplo, o qual consiste em requisitos funcionais, restrições de projetos e requisitos de qualidade de atributos.

3.2 RESULTADOS DA PRIMEIRA REPETIÇÃO ADD

A equipe de arquitetura conduz a primeira repetição. Esta repetição usa um conjunto de guias arquiteturais que consistem nos requisitos da mais alta prioridade, cenários e as suas restrições de projeto associadas. Estes guias arquiteturais, os detalhes das ponderações de equipe e os requisitos adquiridos durante a etapa 1 não estão incluídos aqui. A arquitetura resultante é apresentada na Figura 3 e posteriormente descrita ao final desta seção. Além disso, os elementos do software estão listados na tabela 6, na página 12.

1. Nosso projeto usa um modelo cliente-servidor no qual o Gerenciador de Trilhos fornece serviços para os clientes de atualização e de consulta. Apenas os conectores primários são apresentados no diagrama; pra simplificá-lo, algumas interfaces secundárias não são apresentadas (por exemplo, todos os clients, elementos do Gerenciador de Trilhos e a maioria dos serviços teriam conectores ao seviço de nomeação).
2. O Gerenciador de Trilhos foi dividido em 2 elementos: A e B. Esta decomposição permite que duas estratégias de implementação sejam consideradas:

- Estratégia 1: os dois elementos (A e B) operam em um único processador, P1. A e B juntos consomem 50% do ciclo de trabalho do processador para realizar com 100 atualizações e 30 consultas.
- Estratégia 2: Elemento A está no processador P1 e elemento B está no processador P2. Juntos, eles conseguem lidar com 150 clientes atualizações e 5 clientes de consultas. Esta estratégia execede os requisitos de performance do Sistema.

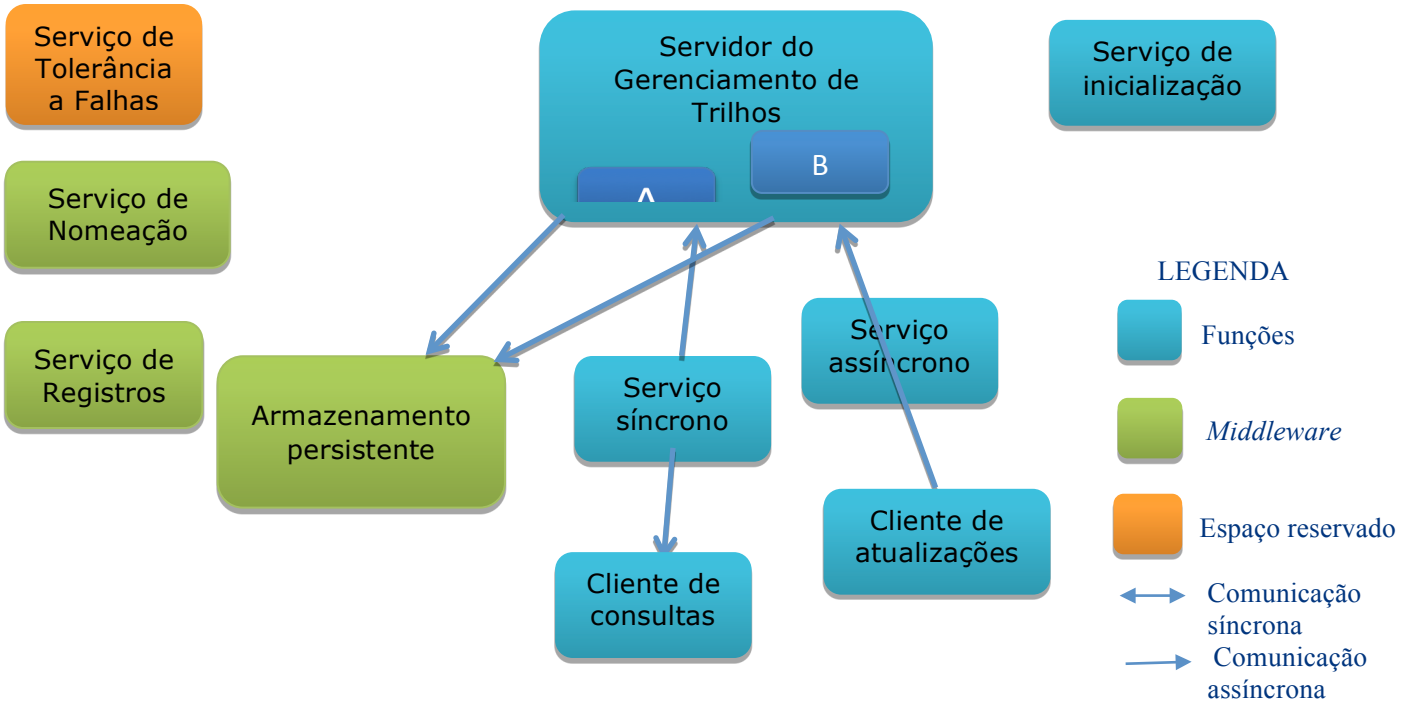

Figura 3: Elemento de software de conectividade primária

O resultado da análise destas estratégias de projetos será apresentado na tabela 6. A largura da banda do sistema de comunicação aumenta em 2% quando os componentes são colocados em processadores diferentes.

Tabela 4: Características de Implementação

	P1	**P2**	**#Atualizações**	**#Consultas**	**Carga P1**	**Carga P2**
Estratégia 1	A,B		100	30	50%	N/A
Estratégia 2	A	B	150	50	50%	30%

3. Os mecanismos de comunicação entre os clientes de atualização e os clientes de consulta e o Gerenciador de Trilhos diferem:

- Clientes de atualização usam um mecanismo de **comunicação assíncrona**. Eles enviam uma mensagem de atualização, não recebem resposta e não são suspensos enquanto a mensagem está

sendo entregue (se estivesse disponível um serviço de publicação /subscrição, seria utilizado em substituição).

- Clientes de consulta usam um mecanismo de **comunicação síncrona**. Eles fazem um requisição de serviço e ficam suspensos até receberem a resposta.

4.Tanto o elemento A como o B contém dados do estado que devem ser salvos como um *checkpoint* no armazenamento persistente. Os tempos decorridos para copiar e para recuperar o estado do armazenamento persistente são idênticos (ver Tabela 5).

Tabela 5: Tempo Decorrido no Armazenamento Persistente

#	**Componente**	**Tempo**
1	A	0.8 segundos
2	B	0.6 segundos

5.Um **serviço de nomeação** *middleware* aceita o nome de um serviço requisitado e devolve um código de acesso para o serviço.

6. Um **serviço de registro** *middleware* nega serviço a novos clientes se fornece-los irá causar um excesso ao limite de capacidade do armazenamento persistente. Para simplificar, as conexões entre clientes e esse serviço não são apresentadas na Figura 3.

7. Uma equipe separada é designada para considerar a inicialização dos elementos do Gerenciador de Trilhos. A interação entre os projetos iniciais da inicialização e dos serviços de tolerância a falha serão resolvidos depois que os dois projetos tenham sido completados.

8. Tanto A quanto B registram suas interfaces com o serviço de nomeação. Mais uma vez, devido à simplicidade, as conexões entre as funções e este serviço não são apresentadas na Figura 3.

9. O que acontece quando um serviço de A ou B é requisitado pela primeira vez depende de que tipo de cliente está fazendo o requisito:

- Quando um cliente de atualização está fazendo o requisito, a mesma vai diretamente de A ou B para o serviço de comunicação assíncrona e depois ao serviço de nomeação para obter o tratador para o serviço. (Neste ponto, os mecanismos de comunicação guardam no cache o tratador, por isso não têm de ir buscá-lo novamente na próxima vez que receba um pedido para este serviço). Então, o serviço de comunicação envia o requisito a A ou B, apropriadamente. O cliente de atulização pode continuar a operação sem esperar uma resposta.
- Quando um cliente de consulta está fazendo o requisito, a mesma vai diretamente de A ou B para o serviço de comunicação síncrona e depois para o serviço de nomeação para obter o tratador para o serviço. Neste ponto, o mecanismo de comunicação guarda no cache o tratador, então ele não tem que pegá-lo novamente da próxima vez que receber uma requisição para esse serviço. Então o serviço de comunicação envia o requisito para A ou B apropriadamente, e espera até que receba uma resposta. Quando recebe, envia a resposta para o cliente de consulta. Durante este tempo todo, o cliente de consulta está bloqueado para executar até que receba essa resposta.

10. A equipe decide ter um especialista em tolerancia a falha para refinar o espaço reservado a tolerância a falha. De fato, eles suspeitam que o espaço reservado a tolerância a falhas é um conceito especulativo que irá permear o sistema – não apenas novos modulos tem que ser adicionados, como mudanças nas funcionalidades e interfaces dos módulos também terão que ser feitas. Neste ponto, eles não sabem quais módulos serão afetados. Os módulos já definidos, os quais incluem o espaço reservado (PH1) para os serviços de tolerância a falhas, estão listados na Tabela 6.

Tabela 6 – Elementos Depois da Repetição I

#	Elemento	Classe de Tolerância a Falhas (Sim/Não)	Alocação dos Guias Arquiteturais
1	Gerenciador de Trilhos	Sim	N/A
2	Cliente de consulta	Não	N/A
3	Cliente de atualização	Não	N/A
4	Armazenamento Persistente	Sim	N/A
5	Gernciador de Trilhos A	Sim	Requisito 1, 3
6	Gernciador de Trilhos B	Sim	Requisito 1, 3
7	Comunicação síncrona	Sim	N/A
8	Comunicação assíncrona	Sim	N/A
9	Serviço de nomeação	Sim	N/A
10	Serviço de registro	Sim	N/A
PH1	Elementos do serviço de tolerância a falhas	Desconhecido	Requisito 5 Cenário 1,2,3 Repetição ADD 1: #1, #3

Ao especialista em tolerância a falhas é dito para se concentrar nos elementos do serviço de tolerância as falhas como eles se aplicam aos elementos do Gerenciador de Trilhos. Depois desta tarefa ter sido completada e aprovada pela equipe de arquitetura, as considerações de tolerância a falhas para outros elementos, tais como comunicações síncronas, podem prosseguir. Estes elementos podem ou não usar os mesmos serviços do que o Gerenciador de Trilhos. O projeto de fazer outros elementos tolerantes a falhas não é considerado aqui.

3.3 CONSIDERAÇÕES ORGANIZACIONAIS

A equipe de arquitetura decide considerar como fazer a tolerância a falhas do Gerenciador de Trilhos antes de criar uma abordagem geral para a tolerância a falha. A equipe pede a um arquiteto com experiência em

tolerância a falhas que tome este espaço reservado e desenvolva uma arquitetura tolerantes falhas usando estas cinco orientações:

- Use os requisitos, os projetos existentes do elemento crítico e os cenários, bem como os guias arquiteturais para adicionar tolerância a falhas ao Gerenciador de Trilhos.

- Se, de acordo com o especialista em tolerância a falhas, os guias arquiteturais estão forçando uma solução demasiadamente complexa, devolva para a equipe de arquitetura com propostas para flexibilizar um ou mais desses guias. A equipe irá tomar as decisões das mudanças necessárias para alcançar uma solução mais simples.
- Capte a racionalidade para a arquitetura tolerante a falhas e as alternativas que foram consideradas. Detalhes sobre cada alternativa não são necessários - só a racionalidade usada quando escolhendo entre as opções.

- Não tente direcionar as preocupações da inicialização. Outra equipe de projeto está combatendo esse problema. A inicilização e as soluções de tolerancia a falhas serão fundidas numa fase posterior.

- Importante: Lembre-se que o seu projeto é preliminar e será fundido com outros projetos que procedem de forma paralela. Não desenvolva um projeto completo. Pare quando estiver confiante que sua abordagem irá satisfazer os guias de arquitetura; por exemplo, não construa um conjunto completo de diagramas em sequencia ou outros diagramas “UML”.

4 Segunda Repetição ADD

4.1 ETAPA 1 DO ADD: CONFIRME QUE HÁ INFORMAÇÕES REQUISITADAS SUFICIENTES

Esta etapa não é necessária durante cada repetição. Foi feita uma vez, no começo do processo ADD.

4.2 ETAPA 2 DO ADD: ESCOLHA UM ELEMENTO DO SISTEMA PARA DECOMPOR

O elemento do serviço de tolerância a falhas é escolhido como o elemento do sistema a decompor. Especificamente o Gerenciador de Trilhos é o alvo, uma vez que ele é o elemento primário do sistema. Como você pode ver na Tabela 7, outros elementos no sistema também devem ser tolerantes a falhas; de qualquer modo, a equipe de projeto queria saber o impacto arquitetural em fazer o Gerenciador de Trilhos tolerante a falhas antes de considerar os outros elementos. Esta decisão, é claro, poderia levar a um retrocesso nas repetições posteriores de ADD se um esquema diferente for necessário para adicionar tolerância a falhas aos outros elementos.

4.3 ETAPA 3 DO ADD: IDENTIFICAR GUIAS DE ARQUITETURA CANDIDATAS

Dez guias e suas prioridades estão listadas abaixo, na Tabela 7. Sete guias são identificados
do conjunto inicial de requisitos de arquitetura. Três são identificados das limitações de projetos resultantes da primeira repetição do ADD.

Considere os seguintes pontos enquanto lê a Tabela 7:
• Guias nomeados (alto, alto) suportam diretamente o requisito de tempo de dois segundos de uma ponta a outra no cenário 1. Esta condição é a mais difícil de satisfazer e tem os guias de mais alta prioridade

• Guias nomeados (medio, medio) são asssociados com o tempo no qual uma única cópia do Gerenciador de Trilhos está operando e a restauração deve ocorrer dentro de dois minutos.

• O cenário de restauração é menos importante e um esforço de projeto "inicialização" separado considera seus detalhes. Sendo assim, os guias #3 não impactam o projeto e estão cruzados na tabela. Como resultado, apenas nove guias arquiteturais devem ser considerados.

Tabela 7: Prioridades dos Guias Arquiteturais

#	Guias Arquiteturais	Seção na qual foi discutida	Importância	Dificuldade
1	Cenário 1 Restauração rápida	2.3	Alta	Alta
2	Cenário 2 Restauração lenta	2.3	Média	Média
3	Cenário 3 Reinicialização	2.3	Baixa	Baixa
4	Requisito 1 Funcionalidade do Gerenciador de Trilhos	2.1	Alta	Alta
5	Restrição de projeto 1 Restrições de capacidade	2.2	Alta	Alta
6	Restrição de projeto 1 Restrições de capacidade	2.2	Média	Baixa
7	Restrição de projeto 1 Restrições de capacidade	2.2	Alta	Alta
8	Restrição de projeto 1 Restrições de capacidade	3.2	Alta	Alta
9	ADD Passo 1, #3 Mecanismos de comunicação	3.2	Alta	Baixa
10	ADD Passo 1, #3 Mecanismos de comunicação	3.2	Alta	Alta

4.4 ETAPA 4 DO ADD: ESCOLHA UM CONCEITO DE PROJETO QUE SATISFAÇA OS GUIAS ARQUITETURAIS.

Esta etapa é a primeira etapa do projeto no método ADD.

Obs: Esta seção possui uma referência cruzada com a Seção7.1 do reporte técnico do SEI intitulado *Attribute-Driven Design (ADD),* Versão 2.0 [Wojcik 2006]. Esta etapa é o ponto central desse documento; é onde a maioria das alternativas de projeto são listadas, padrões preferidos são selecionados, uma avaliação é feita para validar o projeto, e mudanças são feitas para corrigir as deficiências detectadas. Dentro da Seção 7.1, há seis parágrafos enumerados. Para simplificar a sua referência cruzada cada um desses paragrafos está referenciado nos títulos desses relatórios como ADD subetapa 1, 2 ... e assim por diante.

4.4.1 Etapa 4, subetapa 1 do ADD: Identificando as preocupações/interesses do projeto

As três preocupações do projeto associadas com serviços de tolerância a falhas são
• **preparação a falhas:** Esta preocupação consiste naquelas táticas desempenhadas cotidianamente durante a operação normal para garantir que quando uma falha ocorra, possa ser seguida de uma recuperação.
• **detecção das falhas:** Esta preocupação consiste nas táticas associadas na detecção da falha e na notificação de um elemento para lidar com a falha.
• **recuperação as falhas:** Esta preocupação aborda operações durante uma condição transitória — o período de tempo entre a ocorrência de falhas e a restauração da operação normal.

A Tabela 8 mostra estas preocupações e a sua subdivisão em questões secundárias, as seções neste relatório no qual padrões alternativos são listados e as seleções feitas.

Tabela 8:Preocupações de Projeto

Preocupações de Projeto	**Preocupações Subordinadas**	**Seções de Padrões Alternativos**	**Seção de Selecionar Padrões de Projeto**
Preparação para falhas	Reiniciar	4.4.2.1	4.4.3.1
	Implantação	4.4.2.2	4.4.3.2
	Integridade dos dados	4.4.2.3	4.4.3.3
Detecção das falhas	Monitoramento de saúde	4.4.2.4	4.4.3.4
Recuperação das falhas	Transparência aos clientes	4.4.2.5	4.4.3.5
	Começar nova réplica	4.4.2.6	4.4.3.6
	Comportamento do cliente de atualização depois de uma falha transitória	4.4.2.7	4.4.3.7
	Comportamento do cliente de atualização depois de uma falha *Hard*	4.4.2.8	4.4.3.8
	Comportamento do cliente de consulta depois de uma falha transitória	4.4.2.9	4.4.3.9
	Comportamento do cliente de atualização	4.4.2.10	4.4.3.10

	depois de uma falha *Hard*		

4.4.2 Etapa 4, Subetapa 2 do ADD: Liste Padrões Alternativos para Preocupações Secundárias

4.4.2.1 Recomeçar

Quatro alternativas de projeto para reiniciar um componente falho são apresentadas na Tabela 9 abaixo. Dois parâmetros discriminativos são relacionados a estes padrões:

• o tempo de inatividade que pode ser tolerado depois da falha (cenário 1)
• o modo pelo qual o sistema trata requisições para serviços no intervalo de tempo próximo ao tempo de falha; por exemplo, se se os honra e decai o tempo de resposta ou faz com que eles caiam (cenário 1).

Tabela 9 também lista estimativas de tempo de inatividade "razoáveis" (baseadas na experiência) desses parâmetros discriminativos.

Tabela 9: Reiniciar Padrões

#	**Nome doPadrão**	**Tipo de Réplica**	**Estimativa de tempo de inatividade**	**Perda dos serviços**
1	Reinicialização fria	Passiva	> 2 minutos	Sim
2	*Standby* quente	Passiva	> 0.3 segundos	Talvez
3	Mestre/Mestre	Ativa	> 50 millisegundos	Não
4	Compartilhamento de carga	Ativa	> 50 millisegundos	Não

4.4.2.2 Implementação

Os dois componentes podem ser implementados com (1) os dois primários em um processador e os dois secundários no segundo processador ou (2) cada primário em um processador diferente. Os primários são denotados por A e B; os secundários por A' and B'. A condição de falha de B imita a de A e não é registrada na tabela.

Os dois parâmetros discriminativos são
• o tempo de inatividade que pode ser tolerado depois da falha (cenário 1)
• o suporte de 100 clientes de atualização e 25 clientes de consulta (requisito 2)

Tabela 10: Padrões de implementação

#	Nome do Padrão	P #1	P #2	A falha	# Atualizações	# Consultas	Tempo de Recuperação do Estado
1		A, B	A', B'	A', B'	100	30	1.4
2		A, B'	A', B	A', B	150	50	0.8

4.4.2.3 Integridade dos dados

A tática de integridade dos dados garante que quando uma falha ocorra, os secundários tenham estado de informação suficiente para proceder corretamente. Os padrões estão apresentados na tabela 11.

Tabela 11: Padrões de integridade de dados

#	Nome do padrão	Carregamento da Comunicação	Carregamento do Processador de *Standby*
1	*Checkpoint* Devagar	1.2 segundos a cada minuto	Nada
2	*Checkpoint* Rápido	1.2 segundos a cada 2 segundos	Nada
3	*Checkpoint* + mudanças de *Log*	1.2 segundos por minuto + 100 mensagens por segundo	Nada
4	*Checkpoint* + mudanças de Pacote de *Log*	1.2 segundos por miniuto + 1 mensagem por X segundos	Nada
5	*Checkpoint* + Primário sincronizado e *Backup*	1.2 segundos a cada minuto + uma mensagem por X segundos	Executar para manter uma cópia de atualização do estado

4.4.2.4 Monitoramento de saúde

Uma única tática de monitoramento de saúde deve ser considerada para detecção de falhas. A Tabela 12 lista os padrões a serem considerados e seus parâmetros discriminatórios.

Tabela 12: Padrões de Falhas Detectados

#	Nome do Padrão	Carregamento de Linha de Comunicação
1	*Heartbeat*	4 mensagens (para A, A', B, B')
2	*Ping/Echo*	8 mensagens (ping e echo para A, A', B, B')
3	Cliente de Atualização Detecta Falha	0 mensagens
4	Cliente de Consulta Detecta Falha	0 mensagens

4.4.2.5 Transparência aos Clientes

Nós listamos três alternativas para tornar as falhas transparentes paras os clientes na Tabela 13 abaixo. O padrão 1 não tem transparência, mas os padrões 2 e 3 fornecem transparência.

Tabela 13: Padrões de Transparência

#	Nome do Padrão	Protocolo Requisitado	Localização do tempo esgotado
1	***Client Handles Failure***	***Unicast***	Cliente
2	***Handles Failure Proxy***	***Unicast***	*Proxy*

3	***Infrastructure Handles Failure***	***Multicast***	Dentro da infraestrutura

4.4.2.6 Começar Nova Réplica

Esta etapa é adiada já que está fortemente relacionada ao mecanismo de inicialização que está sendo explorado por outra equipe.

4.4.2.7 Comportamento do Cliente de Atualização Depois de Falha Transitória

A operação do serviço de *proxy* quando ocorre uma falha transitória já foi definida: O monitor de saúde informa ao serviço de *proxy* a falha. Depois esse serviço envia um novo código de acesso secundário para cada mecanismo de comunicação a assíncrono. Este código de acesso será utilizado para o próximo requisito de atualização. Essencialmente este mecanismo promove o secundário ao primário.

4.4.2.8 Comportamento do Cliente de Atualização Depois de uma ***Hard*** Falha

Quando uma falha primária e não uma secundaria está disponível, um dos padrões na Tabela poderia ser usado.

Tabela *14: Padrões de Comportamento para Clientesde Atualização depois de uma Hard Falha*

#	Nome do Padrão	Impacto
1	Continua a enviar atualizações	Dados inutilizáveis são enviados.
2	Para de enviar atualizações	A linha de comunicação carregando durante o tempo de inatividade é salva.
3	Salva atualizações em um arquivo	Mensagens maiores são carregadas na inicialização

4.4.2.9 Comportamento do Cliente de Consulta Depois de uma Falha Transitória

A operação do serviço de *proxy* quando tal falha ocorre já foi definida: o monitor de saúde informa o serviço de *proxy* sobre a falha. Depois, este serviço envia um novo código de acesso secundário para cada mecanismo de comunicação síncrono. Se nenhum serviço proeminente estiver em andamento, o mecanismo irá usar este acesso na próxima requisição. Se um serviço requisitado estiver em andamento, uma nova requisição será emitida para o novo código de acesso. É possível que a comunicação síncrona receba múltiplas respostas (uma atrasada do primário e uma do promovido a secundário). Ele deve ser capaz de descartar a segunda resposta.

4.4.2.10 Comportamento de Cliente de Consulta Depois de uma Falha ***Hard***

Quando um primário falha e não tem um segundo disponível, os clientes de consulta serão informados e poderão ajustar seu comportamento apropriadamente. Neste caso, um dos padrões do projeto na Tabela 15 poderia ser usado.

Tabela 15: Padrões de Comportamento de um Cliente de Consulta depois de uma Falha Hard

#	Nome do Padrão	Impacto
1	Continue a enviar consultas	Dados inutilizáveis são enviados.
2	Pare de enviar consultas	A linha de comunicação carregando durante o tempo de inatividade é salva.
3	Salve as consultas em um arquivo	Mensagens maiores são carregadas na inicialização

4.4.3 Etapa 4, Subetapa 3 do ADD: Selecionar Padrões da Lista

Esta atividade envolve selecionar um padrão da lista para cada conjunto de padrões alternativos. Quando estiver fazendo a sua seleção, deverá ponderar sobre qual alternativa é mais adequada. No nosso exemplo, as seleções foram feitas independentemente. Em alguns casos, valores ponderados foram escolhidos como parâmetros de projetos, tais como *heartbeats* e freqüências de *checkpoint*. No resto dessa seção nós consideramos reiniciar, implementação, integridade de dados, detecção de falhas, transparência ao cliente, começar nova réplica e comportamento do cliente depois de falhas transitórias e *Hard*. Para cada item, nós gravamos nosso raciocínio, decisão, e as implicações dessa decisão.

O método ADD requer a implementação de uma matriz que mostre a inter-relação entre padrões e seus prós e contras em cada guia arquitetural. Presume-se que haverá um número razoável de padrões possíveis e que uma tabela é um bom modo de apresentar as alternativas. Infelizmente, a inclusão de todas tolerâncias a falhas como uma única etapa - 4 de decisão de projeto cria um total de 23 padrões — muitos para representar em uma única tabela. Sendo assim, cada alternativa (reiniciar, implantação, etc.) é considerada separadamente. Os prós e os contras na tabela são considerados nas seções separadas abaixo. Cada seção tem três partes: (1) um parágrafo descrevendo o raciocínio dos prós e contras para cada padrão, (2) uma afirmação de decisão enfatizando o padrão escolhido, e (3) uma afirmação de implicação, mostrando o impacto desta decisão, incluindo quaisquer restrições óbvias sobre as escolhas que ainda não foram feitas.

4.4.3.1 Reinicialização

Raciocínio

Tanto o cenário 1 como requisito 1 indicam que o tempo para reiniciar deve ser menor que dois segundos; Sendo assim, o padrão de Reinicialização Fria é inapropriado (ver Tabela 9 na página 18). O padrão de Espera Quente aparenta satisfazer facilmente o requisito de tempo descrito no cenário 1. Dessa forma é escolhido, uma vez que é mais simples para implementar do que o Mestre/Mestre ou que os padrões de Carga Compartilhada.

Decisão

Use o padrão de Espera Quente.

Implicações

1. Um Gerenciador de Trilhos primário para cada componente recebe todas as requisições e responde a elas.
2. Um Gerenciador de Trilhos secundario (*standby*) para cada componente (A' e B') é carregado em outro processador e ocupa memória.

4.4.3.2 Implantação

Raciocínio

O arquiteto é acostumado em ter um único esquema de recuperação às falhas pra recuperar de uma falha de software ou de *Hard*ware. Sendo assim, ele escolhe o primeiro padrão Junto (ver Tabela 10 na página 18), embora ele tenha um tempo de recuperação lenta, já que os estados tanto de A como de B devem ser lidos do armazenamento persistente ao invés do que de apenas A. Este padrão vai de encontro aos requisitos de processamento, embora ele possa executar menos processamento. Note que a granularidade da recuperação difere da granularidade da falha, na qual A e B devem ser recuperados quando qualquer um falhar.

Decisão

Use o padrão Junto com os dois componentes primários que compartilham um processador. Claramente, esta opção não é excelente, já que sua oferta reduz a capacidade e aumenta o tempo de recuperação. De qualquer modo, foi escolhida por razões de familiaridade.

Implicações

1. Os componentes primários (A e B) compartilham um processador, assim como fazem os componentes secundários (A'e B').
2. O sistema nunca será operacional com os componentes primários em processadores diferentes.

4.4.3.3 Integridade dos Dados

Referente a Tabela 11, "Padrões de Integridade de Dados," na página 19.

Raciocinando

1. Claramente um *checkpoint* do estado a cada minuto é necessário para satisfazer o cenário 2. De qualquer forma, um estado que aconteceu a um minuto não pode satisfazer o cenário 1, já que o valor de um minuto de atualizações será ignorado se apenas o *checkpoint* for usado na reinicialização. Padrão 1 é rejeitado.

2. Padrão 2 satisfazeria os requirimentos de atualização dos cenários 1 e 2; de qualquer modo, ele coloca uma carga inaceitável no sistema de comunicação. Padrão 2 é rejeitado.

3. Padrão 3 satisfaria os cenários 1 e 2, mas—assim como o padrão 2— coloca uma carga significante no sistema de comunicação. Padrão 3 é rejeitado.

4. Padrão 4 satisfaz cenários 1 e 2 se *x* for menor que dois segundos. Também põe uma carga mais razoável no sistema de comunicação. Ter um pacote atualizado periodicamente em dois segundos parece ser

satisfatório, embora uma checagem mais detalhada possa ser feita posteriormente (veja a Seção 5). Padrão 4 é, em última instância selecionado.

5. Padrão 5 também satisfaz os cenários, mas é mais complexo, uma vez que o secundário deverá executar a cada *x* segundos para atualizar sua copia do estado. A recuperação será mais rápida, entretanto, desde que não seja necessário ler um *checkpoint* do estado. Padrão 5 é rejeitado devido a sua complexidade.

Decisão

Use o *Checkpoint* + os padrões de Mudança do Pacote *Log*. Os arquivos *log* serão usados como as bases para promover os novos primários.

Implicações

1. A réplica primaria grava o estado a um *checkpoint file* persistente a cada minuto.

2. O primário mantém um pacote de arquivos de todas as mudanças de estado por dois minutos. O primário envia-o como um *LogFile* a cada 2 segundos.

3. O primário promovido lê no *checkpoint file* depois que ele é promovido. Depois ele lê o *LogFile* e atualiza cada mudança de estado na medida em que é lido.

4. A seguir, o secundário promovido escreve os estados atualizados recentemente no armazenamento persistente.

5. O secundário promovido pode agora começar a processar as atualizações e consultas sem esperar até a atualização do estado de persistência ter sido completada.

4.4.3.4 Detecção de Falhas

Raciocínio

Uma abordagem na qual os clientes não detectam as falhas é preferível, já que implica que os desenvolvedores de aplicacação devem entender os requisitos de tempo a tolerância a falhas. Comparando as duas abordagens (ver Tabela 12 na págna 9), a detecção de falhas *ping/echo* é mais complexa do que a detecção dos *heartbeats* e requer duas vezes a largura da banda.

Decisão

Use o padrão *Heartbeats*. Nós estabelecemos os *heartbeats* em 0.25 segundos, o que produz quatro mensagens de comunicação por segundo.

Implicações

1. O *heartbeat* deve ser rápido o suficiente para permitir os secundários a serem inicializados e começarem processando dentro de dois segundos depois que ocorrer uma falha. Inicializar os dois

checkpoint leva 1.2 segundos. O *heartbeat* adiciona 0.25 segundos adicionais, deixando 0.55 segundos extras, o que parece razoável.

2. Um elemento de monitoramento de saúde checa os *heartbeats* a cada 0.25 segundos. Quando um *heartbeat* não é detectado, o monitor de saúde informa todos os elementos necessários.

3. Se um componente primário do Gerenciador de Trilhos detecta uma falha interna, o mecanismo para comunicar a falha é não emitir o *heartbeat*.

4.4.3.5 Transparência ao Cliente

Raciocínio

É indesejável ter clientes que lidam com as falhas, já que esta abordagem requer que o programador escreva ao cliente para que esse entenda o mecanismo de prevenção a falhas. O mesmo pode ser mal interpretado facilmente e tornar se menos robusto.

A infraestrutura não tem embutida a capacidade de enviar dados através de uma rede de computador para vários usuários ao mesmo tempo e adicionar este aspecto seria caro. Você pode simular uma capacidade de enviar dados através de uma rede de computador para vários usuários ao mesmo tempo com vários ***unicasts***, mas esta abordagem dobra o uso do sistema de comunicação e é, portanto, indesejável (para rever as opções de padrões, ver Tabela 13 na página 20).

Decisões

Use o padrão de Controle de Falhas *Proxy*

Implicações

1. O serviço de *proxy* registra os métodos de serviço (por exemplo, A.a, A.b, B.c, B.d) com o servidor de nomeação.

2. O serviço de *proxy* inicia os primeiros componentes, registrando-os sob diferentes nomes (AA.a, AA.b, BB.c, e BB.d) e faz do mesmo modo para os componentes secundários (AA'.a, AA'.b, BB'.c, e BB'.d).

3. O cliente requisita um serviço (A.a). Este requisito faz com que o servidor de nomes a seja chamado e retorne o código de acesso para A.a, designado como acesso (A.a). Em seguida o cliente chama o acesso (A.a).

4. O serviço de *proxy* (A.a) determina que AA é a réplica primária e retorna o acesso (AA.a) ao cliente como uma "requisito encaminhada para".

5. O cliente chama o access (AA.a) e continua a fazê-lo até que AA falhe

6. Quando o monitor de saúde detecta uma falha de *heartbeat* no AA, informa ao serviço de *proxy*.

7. O *proxy* informa os elementos síncronos e assíncronos da falha. Estes elementos enviam seus requisitos de consulta e de atualização ao primário recém-promovido.

4.4.3.6 Começar uma nova réplica

Esta etapa é adiada, já que, nesse exemplo, é parte do mecanismo de iniciação ser explorado por outra equipe.

4.4.3.7 Comportamento do Cliente de Atualização Depois de Falha Transitória

Uma falha transitória ocorre quando o primário falha e uma cópia de segurança é agendada para assumir o controle. No nosso caso, o monitor de saúde detecta a falha e informa o serviço de *proxy*. O mesmo envia uma requisição de código de acesso ao Serviço de Comunicação Síncrona (SCS). Se não há solicitações a caminho, o SCS simplesmente usa o novo código de acesso para todas as requisições futuras. Se um requisito está a caminho, o SCS executa um requisito de encaminhamento com o novo código de acesso ao novo Gerenciador de Trilhos. É possível receber duas respostas: uma resposta do componente falho do Gerenciador de Trilhos, o qual foi inexplicavelmente atrasado além da notificação da falha e um do novo Gerenciador de Trilhos. Se duas respostas forem recebidas a segunda é descartada.

4.4.3.8 Comportamento do Cliente de Atualizações Depois de uma Falha "*Hard*"

Raciocínio

O cenário 2 coloca as bases para esta escolha (ver Tabela 14 na página 20). Nós estamos dispostos a aceitar comportamentos degradantes e reiniciar; no entanto, o padrão 3 é desnecessário e complicado. Não há um objetivo em continuar a enviar atualizações sem ter um Gerenciamento de Trilhos disponível para recebê-los.

Decisão

Nós escolhemos o padrão de Parar de Enviar Atualizações, o qual, quando não tem Gerenciador de Trilhos, para de enviar atualizações até que um novo Gerenciador de Trilhos se torne disponível.

Implicações

Os clientes devem ser capazes de fazer duas coisas: (1) aceitar uma entrada informando-os que o Gerenciador de Trilhos falhou e (2) parar de mandar atualizações.

4.4.3.9 Comportamento do Cliente de Consulta Depois de uma Falha Transitória

Nós escolhemos o mesmo padrão que o cliente de atualização (ver Seção 4.4.3.7) para simplificar.

4.4.3.10 Comportamento do Cliente de Consulta Depois de uma Falha *Hard*

Nós escolhemos o mesmo padrão que o cliente de atualização (ver Seção 4.4.3.8) para simplificar.

4.4.4 Etapa 4, Subetapa 4 do ADD: Determinar a Relação Entre Padrões e Guias

Um resumo dos padrões selecionados está apresentado abaixo, na Tabela 16. No cabeçalho da tabela:

• SC# se refere ao número do cenário que contribui para a decisão da seleção.

• DC# se refere ao projeto prévio # (da repetição 1) que contribuiu para a seleção.

Tabela 16: Mapeamento do Padrão/Guia

#	Tipos de Padrão	Padrão Selecionado	Guia Arquitetural
0	# de Réplicas	Duas Réplicas	Duas Réplicas (DC#3)
1	Reinicialização	*Standby* Quente	Duas Réplicas (DC#3) Recuperação Rápida (SC#1)
2	Implementação	Distribuído	Restrição de Capacidade (DC#1)
3	Integridade de Dados	*Checkpoint* + Mudanças de Pacote de *Log*	Serviço de Armazenamento Persistente (DC#2) Restrição de Capacidade (DC#1) Recuperação Rápida (SC#1) Recuperação L
4	Detecção de Falhas	*Heartbeat*	Restrição de Capacidade (DC#1) Recuperação Rápida (SC#1) Outra – ver nota abaixo
5	Transparência aos Clientes	Falha de Controle de *Proxy*	Restrição de Capacidade (DC#1) Outra – ver nota abaixo
6	Nova Réplica	N/A	N/A
7	Comportamento do Cliente de Atualização - Transitório	Falha de Controle de *Proxy*	N/A
8	Comportamento do Cliente de Atualização - *Hard*	Para de Enviar Atualizações	Restrição de Capacidade (DC#1)
9	Comportamento do Cliente de Consulta - Transitório	Falha de Controle de *Proxy*	Falha de Controle de *Proxy*
10	Comportamento do Cliente de Consulta - *Hard*	Para de Enviar Consultas	Para de Enviar Atualizações

Nota: Há vários exemplos de decisões sendo tomadas baseadas na experiência e na preferência do arquiteto, ao invés de em um guia arquitetural específico. Por exemplo, na seleção de detecção das falhas na Seção 4.4.3.4, o arquiteto considerou inapropriado aos clientes detectarem falhas.

4.4.5 Etapa 4, Subetapa 5 do ADD: Capturar Visões Arquiteturais Preliminares

Nesta seção, nós apresentamos visões arquiteturais preliminares incluindo

• uma tabela de elementos do sistema e a repetição do ADD na qual é desenvolvida

• uma visão funcional da arquitetura

• um diagrama de sequência para o acesso dos clientes de consulta aos dados

4.4.5.1 Lista de Elementos

Tabela 17 lista os elementos do sistema e a repetição ADD na qual eles são desenvolvidos.

Tabela 17: Elementos do Sistema e a Repetição ADD na qual Eles são Desenvolvidos

#	**Este elemento**	**É Desenvolvido Nesta Repetição de ADD**
1	Gerenciador de Trilhos	Requisito
2	Clientes de Consulta	Requisito
3	Clientes de Atualização	Requisito
4	Armazenamento Persistente	Requisito
5	Gerenciador de Trilhos A	1
6	Gerenciador de Trilhos B	1
7	Comunicação Síncrona	1
8	Comunicação Assíncrona	1
9	Serviço de Nomeação	1
10	Serviço de Registro	1
11	Monitor de Saúde	2
12	Servidor de *Proxy*	2
13	*Checkpointfile* A	2
14	*Checkpointfile* B	2
15	*Logfile* A	2
16	*Logfile* B	2

4.4.5.2 A Visão da Arquitetura de um Elemento de Software

Figura 4 mostra uma visão funcional dos elementos do software na arquitetura e suas relações.

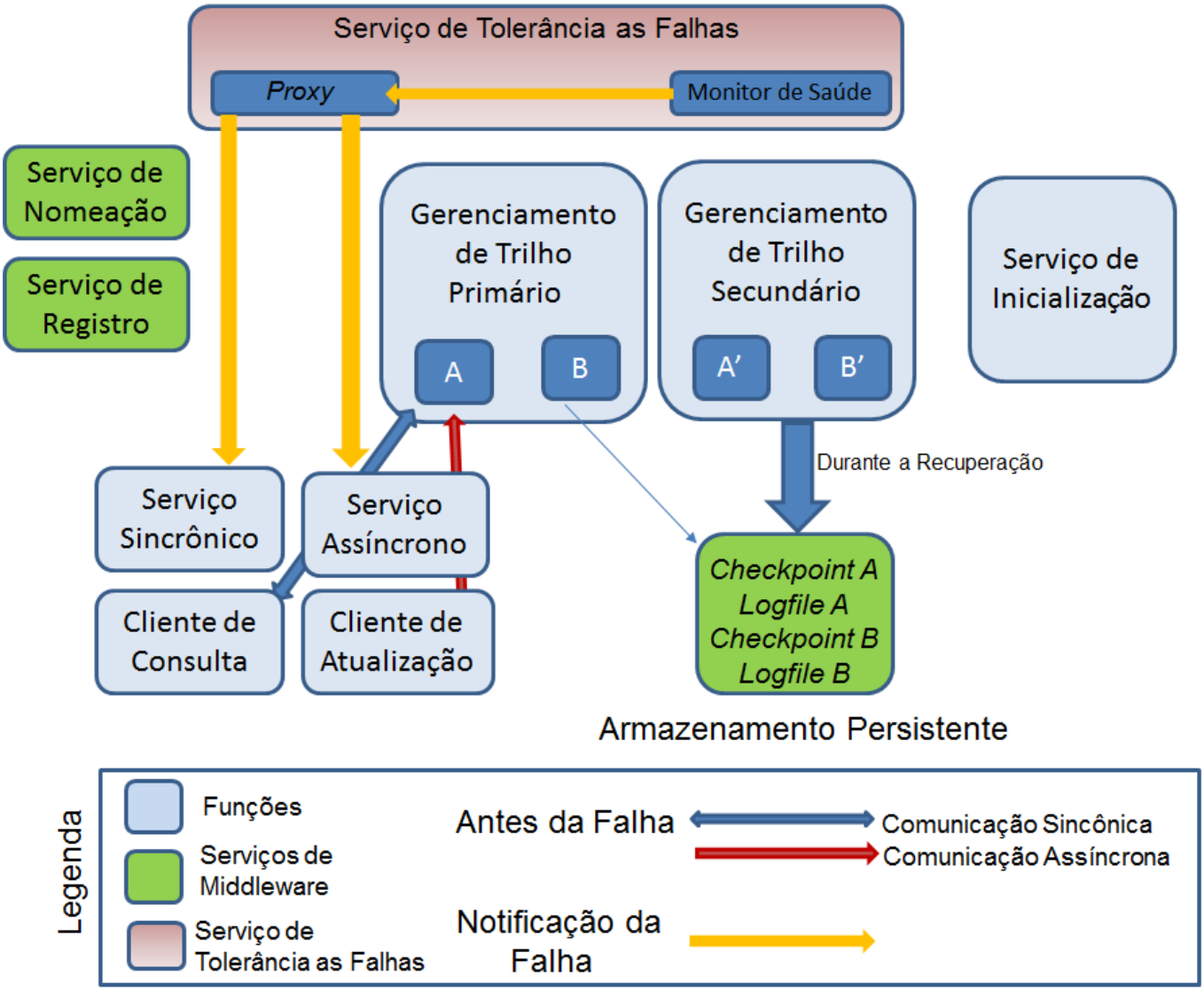

Figura 4: A Visão da Arquitetura de um Elemento de Software

4.4.5.3 Diagrama Sequencial

O diagrama sequencial para o acesso de dados de um cliente de consulta A está apresentado na Figura 5. A Figura ilustra duas sequencias:

1. Para o primeiro requisito, a comunicação síncrona envia o requisito de serviço para o *proxy*. O mesmo retorna uma mensagem de "requisito encaminhado para A". O serviço de comunicação síncrona armazena em cache o requisito encaminhado para A e o usa para todos os futuros requisitos.
2. Se A falha para emitir uma *heartbeat* para o monitor de saúde, o último informa ao *proxy* que A falhou. O *proxy* envia uma mensagem de "encaminhamento do requisito para A" ao serviço de comunicação síncrona. O serviço então encaminha o requisito para A', armazena em cache o pedido, e continua a enviar mensagens para A'.

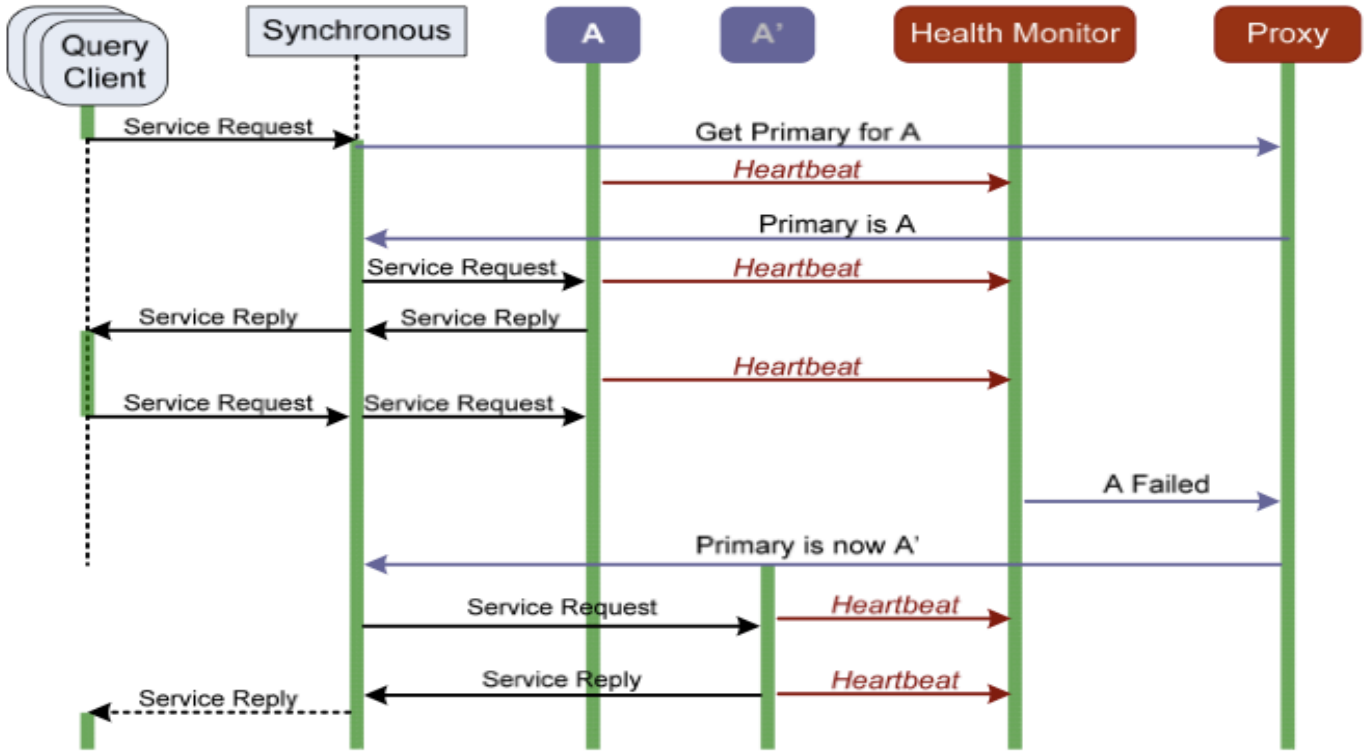

Figura 5: Um Diagrama de Sequência de Falhas do A para o A'

4.4.6 Etapa 4.6 do ADD: Avaliar e Resolver Inconsistências

Numa avaliação de arquitetura, o arquiteto constrói modelos para descrever o comportamento do sistema. O arquiteto então analisa estes modelos para garantir que eles satisfazem os guias arquiteturais. No nosso exemplo, nós desenvolvemos uma linha do tempo mostrando a operação próxima do momento da falha.

A Figura 6 esboça a operação do sistema ao longo de um período de tempo que inclui uma falha.

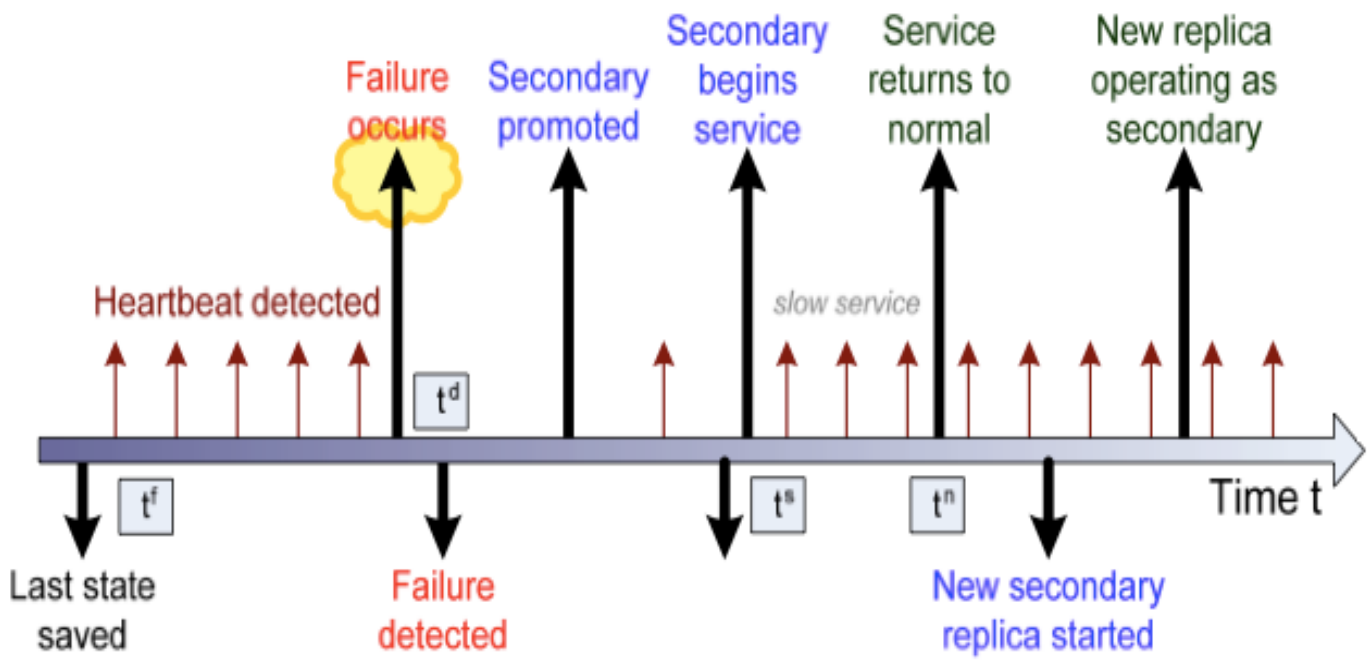

Figura 6: Modelo de Temporização

Os nove eventos que se seguem, os quais ocorrem nesta ordem, estão ilustrados na Figura 6.

1. As atualizações de estado são salvas no *LogFile* persistente.
2. Um *heartbeat* é detectado várias vezes depois do estado salvo.
3. Uma falha acidental ocorre no Gerenciador de Trilhos.
4. O monitor de saúde detecta a falha quando o tempo se esgota antes do *heartbeat*.
5. O Gerenciador de Trilhos secundário é promovido a primário.
6. O serviço secundário começa a responder as requisições dos clientes, trabalhando fora do *backlog* das solicitações e dando tempos de resposta mais lentos.
7. O serviço retorna ao normal quando o período transitório de respostas lentas termina.
8. Uma nova réplica completa a inicialização e está pronta para sincronizar com o atual primário e se tornar o secundário.
9. A nova réplica completou qualquer atualização de estado necessária e o processo de restaurar o serviço está completo.

Seis importantes aspectos de temporização do sistema são apresentados na Figura 6:

• **Tps:** periodicidade do estado do *LogFile* salvo (2 segundos, ver Seção 4.4.3.3)
• **Th:** periodicidade dos *heartbeats* (0.25 segundos, ver Seção 4.4.3.4)
• **TrA:** o intervalo de tempo gasto para recuperar o estado de A do armazenamento persistente (0.8 segundos, ver Tabela 5 na página 11)
• **TrB:** o intervalo de tempo gasto para recuperar o estado de B do armazenamento persistente (0.6 segundos, ver Tabela 5 na página 11)
• **TrL:** o intervalo de tempo gasto para recuperar o *LogFile* do armazenamento persistente (estimado em 0.2 segundos)
• **Tus:** o intervalo de tempo estimado para atualizar o estado de A and B do *LogFile* (estimado em 0.1 segundo)

O pior caso de tempo total (T1) até a recuperação do Gerenciamento de Trilhos ocorre quando a falha é logo depois de um *heartbeat* e logo antes da próxima escrita das atualizações para o *LogFile*. Nesse caso o tempo seria

T1 = Tps + Th + TrA + TrB + TrL + Tus

T1 = 2 + 0.25 + 0.8 + 0.6 + 0.2 + 0.1 = 3.95

O resultado é um tempo inaceitável de 3.95 segundos.

4.4.6.1 Resolvendo as Inconsistências da Temporização

Nós podemos melhorar nossos modelos de vários modos, e nós devemos fazer compensações entre estas melhorias propostas. Nosso principal objetivo é reduzir o tempo de reinicialização de 3.9 para menos de 2 segundos, assegurando ao mesmo tempo que a carga de comunicação continue a ser razoável. Nós podemos modificar importantes aspectos temporais das seguintes maneiras:

• Reduzir a periodicidade do *LogFile* salvo no armazenamento persistente. Sincronizar o *LogFile* salvo e o *heartbeat* de tal modo que ocorram apenas depois que um salvamento já tiver sido iniciado; eles não precisam ter a mesma periodicidade.

• Ter o *LogFile* salvo no armazenamento persistente serve como o equivalente ao *heartbeat*. Envia o *log* a cada 0.5 segundos. Estender o elemento de armazenamento persistente para que reconheça que uma falha no recebimento da atualização do *LogFile* dispara um requisito para informar aos outros elementos necessários sobre uma falha (ex: *proxy*, standby, clientes).

• Fazer os três acessos de armazenamentos persistentes concorrentes ao invés de sequenciais.

• Mudar a decisão de implantação para o segundo padrão, no qual os primários de A e B estão em diferentes processadores; consequentemente, a falha do processador com o componente A será o pior caso (já que gasta mais tempo para recuperar seu estado).

• Mudar o estilo da atualização do estado para a opção 5 na Tabela 11 (na página 19), no qual o secundário mantem um modelo do estado através da sincronização com o primário durante a inicialização. Também recebe um pacote de atualizações do estado periodicamente, tornando óbvia assim a necessidade de ler do armazenamento permanente.

• Reduzir o tamanho do estado a ser salvo pelos componentes A e B recomputando algum dado de estado na reinicialização.

Raciocinando

1. O designer de tolerância a falhas é relutante para escolher as alternativas 2, 5, or 6, uma vez que eles implicam em requisitar mudanças no projeto anterior. O especialista irá apenas fazer isso se não tiver outra forma razoável para reduzir o tempo dentro do qual esteja sob o controle direto dele ou dela. O especialista poderia propor tais mudanças apenas depois de rever outras alternativas.

2. O salvamento do *LogFile* em um armazenamento persistente pode ocorrer a cada segundo e ser sincronizado para ocorrer logo antes de cada quarto *heartbeat*. Este esquema reduz os termos (Tps + Th) de 2.25 segundos para 1 segundo. Esta medida irá garantir um ganho de 1.25 segundos, o qual iria reduzir a resposta para 2.7 segundos, o que ainda não é bom o suficiente. Esta medida poderia ser melhorada posteriormente pela redução da periodicidade para 0.5 segundos, mas esta opção é rejeitada. Causaria muita carga adicional ao mecanismo de comunicação.

3. O acesso ao armazenamento persistente para todos os três arquivos leva (TrA + TrB + TrL) ou 1.6 segundos, se feito sequencialmente. De qualquer modo, se os acessos são concorrentes usando comunicações assíncronas, na teoria eles irão gastar apenas 0.8 segundos, que é o tempo requerido para obter o estado persistente para o componente A. De qualquer forma, uma análise detalhada do armazenamento persistente mostra que as três solicitações concorrentes irão compartilhar alguns recursos e gastar 1.0 segundos. Esta redução é ainda 0.6 segundos, o que nos deixa com uma resposta de 2.1 segundo — ainda não boa o suficiente.

4. A decisão de implantação é modificada para a segunda opção de ter cada A e B em um processador separado. Assim, o pior caso de acesso ao armazenamento persistente ocorre para o componente A e é 0.8 segundos. Se isso continua sendo feito concorrentemente com o acesso do *LogFile*, o tempo total para os dois será 0.85 segundos. As economias em etapa anterior estão agora invalidadas, e os 1.6 segundos agora levam 0.85 segundos, o que produz uma resposta de 1.95 segundos. Esta resposta não fornece uma margem suficiente.

5. O único modo no qual o arquiteto possui controle para resolver posteriormente este problema é selecionar a alternativa D e mudar o estilo de integridade dos dados. Tomar esta abordagem presume que os estados primário e secundário A e B não irão divergir de nenhum modo, o que está fora do controle do arquiteto. O arquiteto então aborda aquelas responsibilidades para o projeto anterior e explica o problema e as opções. A equipe do projetista concorda em reduzir o tempo de atualização do componente A para 0.6 segundos e o acesso concorrente com o *LogFile* para 0.65 segundos. Esta mudança representa reduções posteriores de 0.2 do resultado anterior e cria um tempo de resposta de 1.75 segundos, o que está dentro de uma margem razoável.

4.4.6.2 Resumo das decisões de temporização

A Tabela 18 resume as decisões de temporização.

Tabela 18: Resumo das decisões de temporização

#	Descrição	Intervalo de Tempo Inicial	Intervalo de Tempo Final
Tps	Salvar o *Checkpoint* do arquivo	2.0	1.0 (ver Nota 1)
Th	*Heartbeat*	0.25	0.25
TrB	Recuperar *Checkpoint* para B	0.6	0.0 (ver Nota 2)
TrA	Recuperar *Checkpoint* para A	0.8	0.65 (ver Nota 3)
TrL	Recuperar *LogFile*	0.2	0.2
Tus	Atualizar estado do *LogFile*	0.1	0.1
T	Tempo de recuperação	3.95	1.75
	Estado do *Checkpoint*	60	60

Notas:

1. O *heartbeat* e o *checkpoint* salvos são sincronizados juntos (ponto do raciocínio 2 na Seção 4.4.6.1).

2. Desde que A e B estejam em processos separados, nós apenas temos que recuperar o estado de um deles para uma única falha (ponto de raciocínio 4 na Seção 4.4.6.1).

3. A recuperação do estado e do *checkpoint* são realizadas concorrentemente (ponto de raciocínio 4 e 5 na Seção 4.4.6.1).

4.5 ETAPA 5 DO ADD: INSTANCIAR ELEMENTOS ARQUITETURAIS E DISTRIBUIR RESPONSABILIDADES

4.5.1 A e B primários

O primário e os elementos da cópia de segurança de A e B juntos têm o mesmo comportamento. O comportamento de A sozinho é descrito aqui.

• O elemento A recebe mensagens tanto do cliente de consulta, quanto do cliente de atualização. Ele atualiza o seu estado baseado nas mensagens e respostas às consultas dos clientes de consulta.

• O elemento A normalmente é implantado no mesmo processador que a cópia de segurança B’ do elemento B. Logo depois que uma falha ocorre em B, B’ é promovido, e tanto A quanto B ocupam o mesmo processador até que uma nova versão de B se inicie. O processo de troca do B primário pelo o elemento B que acabou de começar não é definido.

• Elemento A envia um *heartbeat* para o monitor de saúde a cada 0.25 segundos.

• Elemento A copia o seu estado para o arquivo do *checkpoint* A a cada minuto.

• Elemento A acumula as mudanças de estado feitas devido as mensagens do cliente de atualização e as escreve no *LogFile*A a cada 1.0 segundo. Esta escrita é sincronizada com o envio do *checkpoint*.

• A inicialização de A e A' não foi endereçada, já que tem outra equipe abordando esta questão.

• O elemento de *proxy* irá receber uma requisição que se as duas cópias do elemento A falharem, irá parar de mandar atualizações e irá notificar os atores necessários.

4.5.2 Armazenamento Persistente

Existem quatro arquivos de armazenamento persistente: *Checkpoint* A, *Checkpoint* B, *LogFile*A, e *LogFile*B.
Todos os novos valores destes arquivos substituem os valores antigos.

4.5.3 Monitor de Saúde

O monitor de saúde usa um temporizador para checar se ele recebeu um *heartbeat* de A, B, A', e B'. Se falhar em receber um *heartbeat* antes do temporizador expirar, ele notifica o *proxy*.

4.5.4 Comunicação Assíncrona

O mecanismo de comunicação assíncrona recebe um requisito do cliente de atualização para um método (por exemplo, A.a), e direciona o requisito ao elemento apropriado.

1. O mecanismo envia ao servidor de nomes o método A.a e recebe o código de acesso ao elemento de *proxy* A.a.

2. O mecanismo envia a mensagem de atualização para o elemento de *proxy* A.a.

3. Quando o mecanismo recebe o requisito encaminhado por A.a para enviar a mensagem para AA.a, ele envia o requisito para AA.a e armazena em cache o tratador para AA.a.

4. Quaisquer solicitações subsequentes são feitas diretamente ao tratador de AA.a.

5. Quando ocorre uma falha, o mecanismo recebe o requisito encaminhado para AA'.a e usa este controlador para futuras solicitações.

6. Se AA.a falha e não tem nenhum *standby,* o mecanismo informa para o cliente de atualização para parar de enviar atualizações.

4.5.5 Comunicação síncrona

O elemento da comunicação síncrona recebe solicitações dos clientes de consulta e tem quase o mesmo comportamento que o elemento de comunicação assíncrona. A única diferença é que ele bloqueia o cliente de consulta até ele receber a resposta para a consulta, a qual depois envia para o cliente de consulta.

4.5.6 *Proxy*

O elemento de *proxy* faz a maior parte do trabalho para causar uma transição suave para a cópia de segurança quando o primário falhar. Ele faz o seguinte:

1. O serviço de *proxy* registra todos os métodos associados com A e B através do serviço de nomeação.

2. O serviço de *proxy* começa AA, AA', BB e BB' e registra todos os seus métodos com o serviço de nomeação. Ele cria uma cache através do mapeamento dos nomes usados pelos clientes (ex., A.a) e os nomes criados pelos elementos (ex., AA.a e AA'.a). Ele determina quais elementos são primários e quais são secundários.

3. O serviço de *proxy* é chamado tanto pelo elemento de comunicação síncrona ou assíncrona quando um cliente solicita um serviço; por exemplo, A.a. Ele responde com um "requisito encaminhado" para AA.a se AA é o primário.

4. Quando o monitor de saúde assinala para o *proxy* que o primário (ex. AA) falhou, ele envia um requisito encaminhado tanto para os elementos de comunicação síncronos quanto os assíncronos para acessar os métodos de standby (ex. AA'.a), promovendo assim o AA' a ser primário.

4.5.7 Clientes de atualização

A falha de um componente primário (ex., A) e a substituição para A' são transparentes para o cliente de atualização. Quaisquer atualizações enviadas durante a abertura entre a falha e a restauração são perdidas. Mas este tempo de abertura foi analisado como sendo pequeno suficiente para que o Gerenciador de Trilhos continue trabalhando mesmo com estas mensagens perdidas. Quando o componente primário falha e não tem uma cópia de segurança, o cliente de atualização será notificado e irá parar de enviar atualizações até que o serviço seja reiniciado.

4.5.8 Cliente de Consulta

A falha do componente primário quando tem uma cópia de seguraança é mais uma vez transparente para os clientes de atualização. Eles terão que esperar um pouco mais por uma resposta às suas consultas, mas este tempo foi avaliado como aceitável. Quando o componente primário falha e não tem cópia de segurança, o cliente de consulta será notificado e irá parar de solicitar consultas.

As interfaces foram definidas através da etapa 4 na Seção 4 mas são capturadas aqui por consistência e conveniencia. Note que algumas interfaces que foram definidas na primeira repetição (ver Seção 3.2) não são repetidas na Tabela 19.

Tabela 19: Resumo das Interfaces

Do Elemento	Para o Elemento	Interface	Condições Temporais	Seções Descritivas
Primário A	*CheckpointA*	Atualização do estado	60 segundos	4.5.1
Primário A	*LogFileA*	Mudanças no *Log*	1 segundo	4.5.1
Primário A	Monitor de Saúde	*Heartbeat*	0.25 segundos	4.5.1
Primário B	*CheckpointB*	Atualização do estado	60 segundos	4.5.1
Primário B	*LogFileB*	Mudanças no *Log*	1 segundo	4.5.1
Primário B	Monitor de Saúde	*Heartbeat*	-	4.5.1
CheckpointA	Primário A	Atualização do estado	Durante a recuperação	4.5.1
LogFileA	Primário A	Mudanças no *Log*	Durante a recuperação	4.5.1
CheckpointB	Primário B	Atualização do estado	Durante a recuperação	4.5.1
LogFileB	Primário B	Mudanças no *Log*	Durante a recuperação	4.5.1
Monitor de Saúde	*Proxy*	Falha Primária	Dentro de 1 segundo de detecção	4.5.3
Cliente de Consulta	Comunicação Síncrona	Requisito para Serviço	5 por segundo	4.5.8
Proxy	Nomeação	Registro dos Serviços de A, B, A', B'	Durante a inicialização	4.5.6
Proxy	Comunicação Síncrona	Falha do Primário (A ou B	Durante a recuperação	4.5.6
Proxy	Comunicação Assíncrona	Falha do Primário (A ou B	Durante a recuperação	4.5.6

4.6 ETAPA 7 DO ADD: VERIFICAR E REFINAR REQUISITOS E TORNÁ-LOS RESTRIÇÕES

Na etapa 7, nós verificamos que a decomposição dos serviços de tolerância a falhas que suportam os elementos do Gerenciador de Trilhos satisfaz as exigências funcionais, requisitos de qualidade do atributo, e restrições de projeto, e nós mostramos como esses requisitos e restrições também limitam os elementos instanciados.

Os guias arquiteturais são apresentados mais uma vez na Tabela 20 e foram usados em uma ou mais seleções de padrões (exceto para o cenário 3). O cenário reiniciado não foi usado explicitamente, uma vez que o projeto de reiniciação foi feito em paralelo e uma fusão posterior foi antecipada.

Tabela 20: Guias Arquiteturais

#	Guias Arquiteturais	Definidos na Seção	Se Aplica a Escolha de Padrões
1	Cenário 1 Recuperação Rápida	2.3	Reinicialização, Implementação, Integridade de Dados, Detecção de Falhas
2	Cenário 2 Recuperação Lenta	2.3	Integridade das Informações
3	~~Inicilização do Cenário 3~~	~~2.3~~	~~Não usado~~
4	Requisito 1 Gerenciador de Trilhos Funcionalidade	2.1	Reiniciar
5	Requisito 2 *Checkpoint* para o Armazenamento Persistente	2.1	Implementação
6	Restrição de Projeto 1 Capacidade Livre	2.2	Detecção de Falhas
7	Restrição de Projeto 2 Duas Réplicas	2.2	Reinicialização, Implementação
8	Etapa 1 do ADD, #1 Características da Implantação	3.2	Reinicialização, Implementação, Integridade de Dados
9	Etapa 1 do ADD, #2 Mecanismos de Comunicação	3.2	Comportamento do Cliente de Atualização Comportamento do Cliente de Consulta
10	Etapa 1 do ADD, #3 Temporizador do *Chechpoint*	3.2	Integridade de Dados

Notas:

1. A repartição da alocação de requisitos de tempo derivada do cenário 1 está apresentada na Tabela 18 na página 31.

2. As capacidades adicionais requisitadas pelos elementos definidos prioritariamente para esta etapa (Gerenciador de Trilhos, clientes de consulta, clientes de atualização, armazenamento persistente, comunicação síncrona e assíncrona) são todos definidos na Seção 4.5. Os serviços de nomeação e registro não requerem extensões.

3. A responsabilidade dos dois novos elementos (serviço de *proxy*) e (monitor de saúde) são totalmente descritos na Seção 4.5.

4.7 ETAPA 8 DO ADD: REPETE DA ETAPA 2 ATÉ A 7 PARA O PRÓXIMO ELEMENTO DO SISTEMA QUE VOCÊ QUISER DECOMPOR

Agora que completamos as Etapas de 1 até a 7, temos uma decomposição do serviço de tolerância as falhas (especificamente, o elemento do sistema de Gerenciamento de Trilhos). Nós geramos uma coleção de resposnsabilidades, cada uma tendo uma descrição de interface, requisitos funcionais, requisitos de qualidade de atributo e restrições de projetos. Você pode voltar ao processo de decomposição na Etapa 2 onde você seleciona o próximo elemento para decompor. No nosso caso, nós não temos elementos filho para decomposições posteriores.

5 Resumo da Arquitetura

A arquitetura desenvolvida até este ponto é descrita nessa seção. Também estão inclusos lembretes sobre os projetos paralelos a caminho com os quais esta arquitetura deve ser resolvida e com as questões que ainda devem ser abordadas.

5.1 RESUMOS DA ARQUITETURA

Um resumo do projeto está dado abaixo.

• O Gerenciador de Trilhos tem dois componentes (A and B), implantados em duas plataformas de *Hard*ware.
Cada plataforma de *Hard*ware contém o primário de um componente e o secundário do outro componente. A falha de A e B tem atividades equivalentes e a falha de Z=A ou Z=B está discutida abaixo.

• Os componentes primários e secundários (A, A' e B, B') irão fornecer um *heartbeat* a cada 0.25
Segundos para um monitor de saúde.

• Um mecanismo de armazenamento persistente é requisitado para salvar o estado dos componentes a cada minuto. Cada componente irá escrever seu estado separadamente no armazenamento persistente como *CheckpointA* e *CheckpointB*. Estas escritas são sincronizadas para ocorrer com o *heartbeat*.

• Cada componente primário irá acumular suas mudanças de atualização por um segundo e depois irá enviá-las para um *LogFile* no armazenamento persistente. Cada componente irá sincronizar o envio do seu *LogFile* com o *heartbeat*. Haverá um *LogFile*A e um *LogFile*B.

• Quando o monitor de saúde determinar que houve uma falha em Z, ele informa o serviço de *proxy*
e o componente de *standby* Z' sobre a falha. O seviço de *proxy* também envia um "requisito de encaminhamento" tanto para os componentes de comunicação assíncronos como os síncronos, indicando que Z' é o novo primário.

• Quando o componente de *standby* Z' recebe o sinal de que o seu primário falhou, ele lê assincronamente o *CheckpointZ e LogFileZ* do armazenamento persistente. Ele computa o novo estado de Z, o qual salva no armazenamento persistente. Agora está pronto para responder as solicitações dos clientes.

• Quando um componente da comunicação síncrona recebe o sinal dos "requisitos encaminhados" do *proxy*, ele envia a próxima atualização para o primário. Não sabe (ou se preocupa) se atualizações anteriores foram recebidas.

• Quando um componente de comunicação assíncrona recebe o sinal do *proxy*, uma das seguintes situações ocorrem:

− Não tem solicitações ativas, então envia a próxima atualização para o novo primário.
− Tem uma atualização ativa a caminho, então envia a requisição atual para o novo primário.

• Quando uma nova réplica de *standby* é iniciada, a primária já está em vigor.

• Quando uma nova réplica é iniciada e promovida a primária, lê o *CheckpointZ e o LogFileZ* e atualiza o estado de A. Então procede para lidar com os requisitos de atualização, mas ignora as requisições de consulta. Muitas atualizações são indeterminadas dedivo ao tempo de inatividade. Alguma ajuda do operador é necessária para trazer o sistema de volta para um estado razoável, no qual consultas de tempo serão retomadas. Este projeto tem muitas sobreposições com o projeto de inicialização inicial, mas os datalhes do mesmo não são parte deste empenho de projeto.

5.2 QUESTÕES DE PROJETO SENDO RESOLVIDAS EM OUTROS LUGARES

Alguns projetos estão sendo resovidos em outros lugares:

• o mecanismo do armazenamento persistente

• os procedimentos de iniciação de A e B e a sua coordenação

5.3 QUESTÕES DE PROJETO REMANESCENTES

Este relatório fornece uma visão global de como fazer a tolerância a falhas do Gerenciador de Trilhos e satisfazer os guias arquiteturais, especialmente o requisito de tempo de uma ponta a outra. Algumas visões do software foram captadas, mas algumas estão faltando, tais como diagramas de classe, visões em camada, visões de receptáculo, casos de uso e muitos diagramas de sequencia detalhando a informação incorporada nas várias seções da Seção 4.4.3. “Raciocínio” e “Implicações”. Em particular, um diagrama de um estado de transição que mostra os detalhes de como os elementos do software respondem juntos para fornecer o comportamento desejado está faltando. Este modelo é o único modo de garantir que não há “aberturas temporais” no comportamento.

Em acréscimo, quatro questões de projeto precisavam ser abordadas:

• Como o monitor de saúde e os elementos do *proxy* se recuperam das falhas no *software* ou *Hard*ware, e como elas são distribuídas?

• O serviço de *proxy* não sabe quais clientes tem um requisito de serviço a caminho e quais não têm. Como um cliente reage a um “requisito encaminhado” quando não há requisições a caminho?

• Como o monitor de saúde sabe do *proxy*?

• Como o sistema responde a uma falha em um componente secundário?