AVALIAÇÃO FUNCIONAL DE ARQUITETURAS DE SISTEMAS DE ARQUIVOS DISTRIBUÍDOS

INSTITUIÇÃO: CEFET-PR

AUTOR : Marcos Antonio Masnik Ferreira, Msc

ENDEREÇO: Rua Roberto Christensen, 133 Casa 3, Portão, CEP 80.330-270, Curitiba-Pr.

FONE : 247-3221

S U M Á R I O

Este artigo apresenta um modelo de avaliação funcional de arquiteturas de sistemas de arquivos distribuídos. Os sistemas operacionais distribuídos são constituídos por determinadas características abstratas (transparência, extensibilidade, disponibilidade, etc) que os diferenciam dos sistemas centralizados e dos sistemas operacionais de rede. O modelo proposto permite, para cada característica abstrata selecionada, (necessária ao projeto de um sistema de arquivos distribuído) determinar quais são as características concretas (mecanismos de implementação) que deverão estar presentes no sistema, a fim de que aquela característica abstrata seja atendida. Portanto, baseando-se no modelo proposto ao avaliar-se uma determinada arquitetura de sistema de arquivos distribuído, é necessário investigar as características concretas do projeto, que em conseqüência determinarão se as abstrações são ou não atendidas.

CURITIBA

1993

1. INTRODUÇÃO.

O surgimento de sistemas distribuídos trouxe aos usuários um novo ambiente onde há maior flexibilidade, potencial de trabalho e capacidade de expansão. Em contraposição aos sistemas centralizados, os sistemas distribuídos caracterizam-se pela existência de diversas unidades de processamento distribuídas, e pelo fato da comunicação entre estas unidades ser feita através da troca de mensagens.

Os sistemas operacionais que trabalham em ambientes distribuídos são classificados em **sistema operacional distribuído** (SOD) e **sistema operacional de rede** (SOR). Apesar do crescente uso de sistemas distribuídos em ambientes computacionais, muita confusão existe entre os termos: sistemas distribuídos, sistema operacional distribuído e sistema operacional de rede.

Arquivos são uma das principais abstrações fornecidas por um sistema operacional aos usuários. Portanto, exercem um papel fundamental na classificação de um sistema operacional como distribuído ou de rede.

O objetivo deste trabalho é apresentar um modelo de avaliação funcional, o qual define os mecanismos de implementação necessários em um sistema de arquivos distribuído. Portanto, o modelo auxilia a compreender uma determinada implementação de sistema de arquivos, permitindo inferir se esta implementação atende aos requisitos de um sistema operacional distribuído.

2. MODELO PROPOSTO PARA AVALIAÇÃO DA FUNCIONALIDADE DE ARQUITETURAS DE SISTEMAS DE ARQUIVOS DISTRIBUÍDOS.

Os anos de experiência no desenvolvimento de sistemas de arquivos centralizados e de sistemas de arquivos baseados em rede contribuíram para o surgimento de novas técnicas de administração de dados, algumas delas importantes no desenvolvimento de **sistemas de arquivos distribuídos** (SADs). No entanto, na maioria dos casos, as técnicas originadas nestes sistemas não podem ser diretamente aplicadas. Alguns aspectos do projeto de SADs, por exemplo, são conflitantes (concorrência no acesso e disponibilidade dos dados) e precisam ser balanceados de acordo com os objetivos esperados do sistema.

Analisar arquiteturas distintas de SADs pode ser uma tarefa bastante árdua, caso o analista não compreenda claramente quais critérios considerar e qual ponderação aplicar. Um esforço muito grande pode ser dispendido e, talvez, resultados modestos sejam alcançados.

O modelo apresentado neste trabalho determina critérios, através de dois níveis (**abstrato** e **físico**), que possibilitam a compreensão das virtudes e deficiências de uma determinada implementação de SAD. Além disso, o modelo permite estabelecer comparações objetivas entre implementações distintas.

No nível abstrato do modelo situam-se as características fundamentais no projeto de um SAD [SAT 90b], [SIL 90] e [GOS 91]. Apesar de sua fácil enumeração, estas abstrações desejáveis não são facilmente identificáveis ou comprováveis no projeto, porém são fundamentais na concepção de uma arquitetura distribuída. Por outro lado, o nível físico compõe-se dos mecanismos e técnicas de implementação que podem ser utilizados pelo projetista, a fim de que atenda a uma ou mais características do nível abstrato.

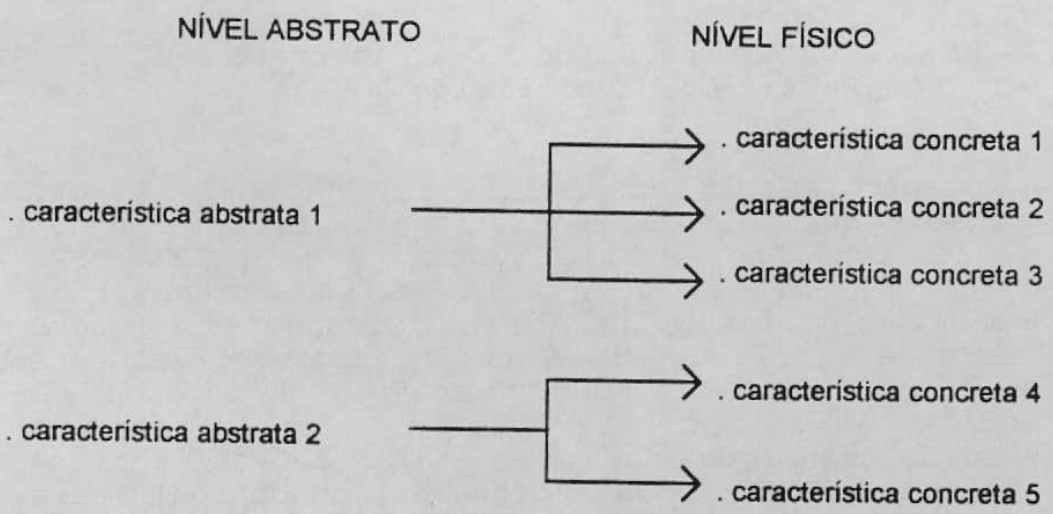

FIGURA 2.1 Determinação do Nível Físico a Partir do Nível Abstrato.

O modelo desenvolvido neste capítulo é uma relação de mapeamento entre o nível abstrato e o físico. Para cada característica selecionada no nível abstrato são determinados os mecanismos e técnicas necessárias no nível físico (figura 2.1).

O objetivo desta relação é prover o analista com subsídios concretos que validem as características abstratas da arquitetura. Com os dados concretos, facilmente identificáveis na descrição de um produto, o analista poderá proceder o caminho inverso, conforme mostra a figura 2.2.

A tarefa do analista, no escopo do modelo apresentado, é o de identificar os mecanismos utilizados no projeto analisado, a fim de estabelecer a adequação ou não às características abstratas atribuídas ao produto.

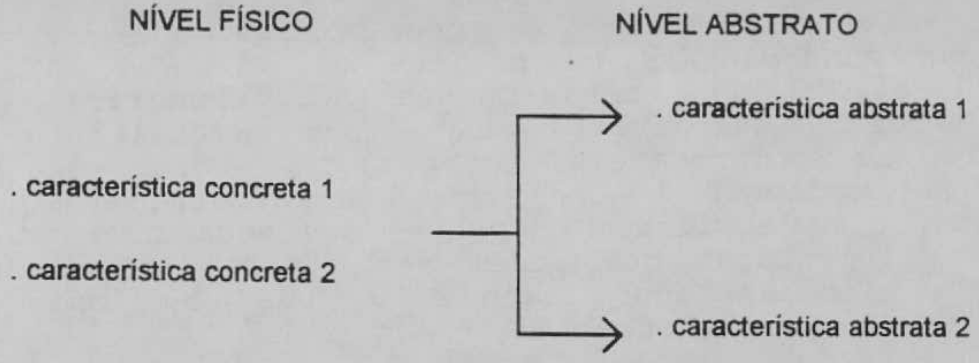

FIGURA 2.2 Determinação do Nível Abstrato a Partir do Nível Físico.

A especificação dos critérios do modelo, tanto no nível abstrato quanto físico, foi baseada fortemente nos sistemas de arquivos **Andrew File System** (AFS) e **CODA** [SAT 90a], [SAT 90b] e [SAT 91]. A escolha destes sistemas como base deveu-se ao fato de serem enquadrados como pertencentes à categoria de sistemas de arquivos realmente distribuídos. Outro aspecto que justifica a escolha do sistema Andrew é o fato de que a tecnologia de processamento de transações da empresa **TRANSARC** foi adotada recentemente pela **OSF** (**Open Software Foundation**) para prover serviços de compartilhamento de dados no **Distributed Computer Environment** (**DCE**). Esta companhia é financiada pela IBM e HP e utiliza o sistema Andrew para prover compartilhamento de dados. Este trabalho não descreve a estrutura lógica e os princípios de implementação destes sistemas de referência.

A seqüência de apresentação das características abstratas do modelo é feita unicamente por critérios de clareza e compreensão, portanto, a ordem de apresentação não está vinculada à importância de cada elemento do modelo. As características abstratas selecionadas, dentro

do contexto de **funcionalidade** proporcionada aos seus clientes, foram: **transparência do sistema de arquivos, mobilidade de usuários, extensibilidade, segurança de dados** e **disponibilidade de dados.**

A estrutura de apresentação de cada característica é desenvolvida, quando possível, descrevendo na seqüência o significado da abstração, a sua importância dentro do modelo, as formas de implementação e a relação, se existente, com as demais abstrações.

2.1 Transparência do Sistema de Arquivos.

Genericamente, transparência a nível do sistema de arquivos significa que a localização física distribuída dos arquivos é tornada oculta aos usuários [GOS 91].

O fato das arquiteturas de SADs apresentarem graus distintos de transparência aos usuários, torna esta abstração de vital importância como critério de diferenciação entre elas. Transparência em ambientes distribuídos é importante para os usuários pois torna as aplicações mais simples e baratas. Além disso, a administração do ambiente torna-se mais simplificada. O grau de transparência a nível dos arquivos influi diretamente na transparência do sistema operacional, atuando como elemento diferenciador da classificação do sistema como distribuído ou de rede.

A adição de transparência no sistema operacional UNIX pode ser feita tanto a nível do núcleo do sistema operacional, quanto a nível da biblioteca do sistema. A adição a nível de núcleo (na API) permite portabilidade das aplicações, ao passo que, a adição de transparência na camada de biblioteca do sistema exige que os programas sejam, no mínimo, ligados (*linked*) novamente.

Quando se discute transparência do sistema de arquivos, qualquer que seja o nível de implementação, quatro aspectos podem ser considerados [WIL 90]: **nome** (*name transparency*), **localidade** (*location transparency*), **consistência semântica** ou **semântica de compartilhamento** (*semantic consistency*) e **desempenho.**

Transparência de nome significa acesso aos arquivos com o mesmo nome em todas as máquinas do sistema distribuído. Isto implica que o nome do arquivo, utilizado nos comandos e funções de acesso aos arquivos, será o mesmo, independente do nodo onde for utilizado. Com isto, programas podem ser transportados de uma máquina a outra sem necessidade de alteração.

Compreende-se que transparência de localidade está presente no sistema se o nome de um dado não contém referência à localidade onde o dado está armazenado. Os dados podem ser transferidos de um nodo a outro, sem a necessidade de alteração dos programas que os acessam. A transparência de localidade, quando aliada à transparência de nome, reduz sensivelmente a complexidade no desenvolvimento de aplicações distribuídas.

Os dois tipos de transparência não são mutuamente exclusivos, porém complementares. Os sistemas Andrew e CODA apresentam tanto transparência de nome quanto de localidade. Já o sistema Newcastle Connection [BRO 82] apresenta transparência de nome, mas não provê transparência de localidade. O contrário também é possível, ou seja, um sistema pode apresentar transparência de localidade e não prover transparência de nome.

Consistência semântica é a propriedade que permite que a execução de serviços em um SAD produza os mesmos resultados, independente do local (nodo) onde forem executados.

Os sistemas Andrew e CODA apresentam transparência de nome, de localidade e consistência semântica dentro do seu espaço compartilhado. Em relação ao sistema de arquivos local, no entanto, apresentam uma aproximação quanto à consistência semântica. Por exemplo, no espaço local do sistema UNIX, quando o conteúdo de um arquivo é modificado, imediatamente todos os processos que compartilham o arquivo visualizam a alteração. O mesmo não ocorre no espaço compartilhado dos sistemas Andrew e CODA, em função da abordagem otimista de atualização das caches de dados do sistema de arquivos.

Outro aspecto que influencia a questão de transparência é o desempenho do SAD. Um acesso remoto só é transparente se o seu tempo de resposta estiver bastante próximo de um acesso local. Caso contrário, a existência da distribuição física não será transparente ao usuário. No modelo de avaliação proposto entende-se desempenho como uma sub-abstração da característica abstrata transparência.

São duas as principais questões no projeto dos SADs que mais influenciam o seu desempenho [HOW 88]: utilização de cache de dados e o mecanismo de controle de acesso.

O mecanismo de cache de dados pode ser utilizado tanto nos servidores quanto nos clientes. Do ponto de vista de desempenho, o que deve ser decidido é a unidade (registros, páginas, arquivos) que será mantida na cache. Unidades pequenas (registros) implicam em constante tráfego com os servidores. Unidades grandes (arquivos) acarretam tráfego intenso apenas no início e término de uma transação.

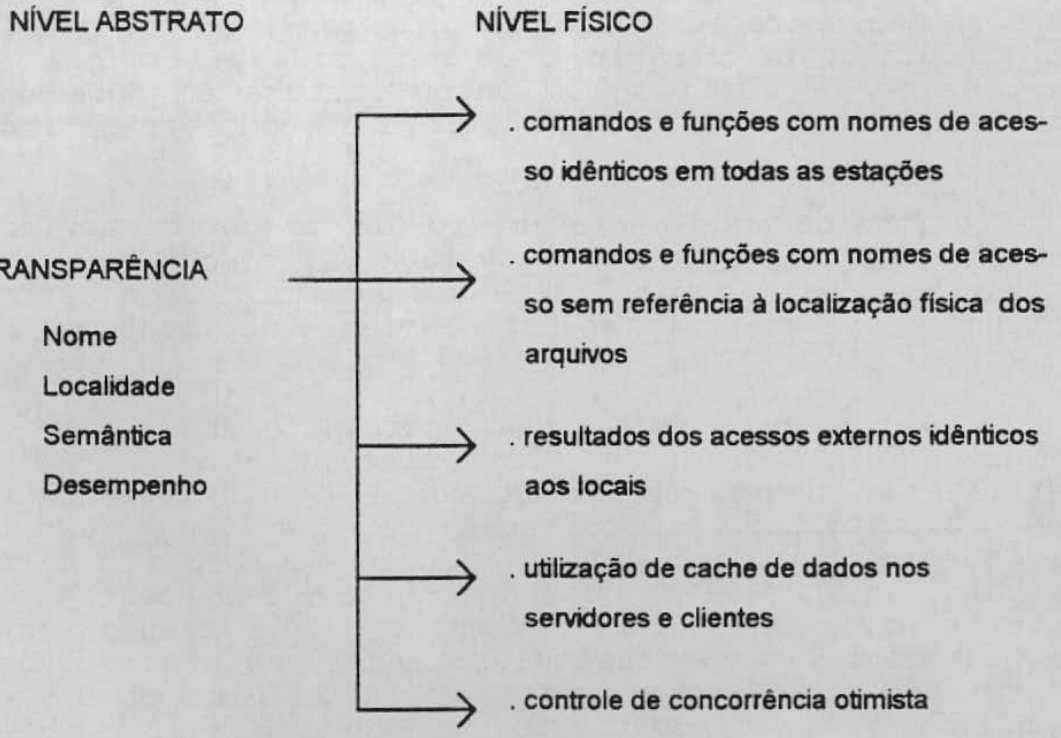

FIGURA 2.3 Mapeamento entre a Característica Abstrata *Transparência* e as Características Físicas no Projeto.

O controle de acesso possui a finalidade de garantir que atualizações simultâneas em arquivos do sistema produzam o mesmo resultado caso fossem executadas seqüencialmente [TAN 87]. A segunda versão do AFS adotou o mecanismo de controle de acesso otimista por razões de desempenho, em contraposição ao mecanismo pessimista adotado na sua primeira versão. O sistema CODA manteve o mecanismo otimista com algumas pequenas mudanças visando proporcionar uma maior disponibilidade de dados [SAT 90a].

A figura 2.3 resume o mapeamento entre a característica abstrata transparência e as características físicas necessárias no projeto.

2.2 Mobilidade dos Usuários.

A capacidade de um usuário poder deslocar-se de um máquina a outra, não permanecendo vinculado a um nodo específico, é denominada mobilidade dos usuários [SAT 90b].

O conceito de máquina virtual proporcionado pelos sistemas operacionais multi-usuários [PET 85] permite a não vinculação do usuário a um determinado terminal. Da mesma maneira, a visão de um único sistema em SODs deve permitir a não vinculação do usuário a um nodo específico.

Considerado o provimento de um único ambiente operacional (*single system image*) [SUM 89] como o principal objetivo dos SODs , a mobilidade dos usuários é de extrema importância por materializar este ambiente unificado.

Para que se possa permitir uma completa mobilidade dos usuários no sistema, os arquivos compartilhados devem ser armazenados exclusivamente no espaço compartilhado do sistema de arquivos. Caso os arquivos estejam armazenados no espaço local das estações de trabalho, tornar-se-ia bastante difícil implementar esta característica de forma completa e transparente. Conseqüentemente, o eventual disco local das estações deve ser utilizado exclusivamente como cache de dados do sistema de arquivos e como repositório de arquivos temporários e bibliotecas (/tmp e /lib, por exemplo). Como os sistemas Andrew e CODA provêem um espaço compartilhado para o armazenamento permanente dos arquivos, através dos servidores, eles atendem à característica mobilidade aos usuários.

Outra necessidade para que o sistema de arquivos possa prover mobilidade dos usuários é que a forma de acesso aos arquivos seja a mesma em todas as estações (transparência de nome). Caso contrário, tornar-se-ia difícil a mudança de estações, pois, envolveria converter programas existentes.

O armazenamento dos arquivos compartilhados no espaço compartilhado do sistema de arquivos favorece também a **extensibilidade** (item 2.3) do sistema. A inexistência de arquivos pertencentes ao espaço compartilhado nos discos locais das estações facilita a adição e remoção de estações de trabalho.

A figura 2.4 mostra o mapeamento entre a característica abstrata mobilidade dos usuários e as características físicas necessárias no projeto.

FIGURA 2.4 Mapeamento entre a Característica Abstrata *Mobilidade dos Usuários* e as Características Físicas no Projeto.

2.3 Extensibilidade.

Grau de extensibilidade, no contexto do sistema de arquivos, representa a capacidade que o sistema possui de ampliar seu potencial de armazenamento, o número de suas estações de trabalho e de usuários, sem queda no desempenho e sem aumento da complexidade de administração do sistema [SAT 90b].

Extensibilidade a nível de SOD é fundamental devido à natureza dinâmica do ambiente. Após a implantação do sistema, novos nodos podem ser incluídos no sistema, até mesmo servidores de arquivos, e, portanto, o sistema de arquivos deve prover suporte adequado para sua expansão.

Conforme [SAT 90b], extensibilidade é uma das características que não estão presentes nos sistemas de arquivos por acidente ou pela adição de camadas suplementares. Ao contrário, é um objetivo de projeto do sistema. A adição de camadas aos sistemas operacionais existentes, com o objetivo de prover transparência, apenas produz sistemas operacionais de rede.

Extensibilidade foi a grande preocupação dos sistemas Andrew e CODA. Ambos orientam-se, no tocante ao aspecto de extensibilidade, fundamentalmente pelas seguintes diretrizes de projeto:

a) executar ações, preferencialmente, nas estações de trabalho;

b) empregar cache de dados sempre que possível;

c) minimizar informações globais do sistema.

O princípio de executar ações preferencialmente nas estações de trabalho, ao invés de executá-las nos servidores, têm por objetivo não sobrecarregar estes últimos, o que degradaria o desempenho do sistema com a adição de novos usuários. Andrew e CODA executam nos servidores de arquivos apenas ações críticas para segurança, integridade e localização de dados.

A utilização de cache de dados, além de facilitar a extensibilidade do sistema, permite mobilidade dos usuários e autonomia das estações em caso de falhas da rede, aumentando a disponibilidade dos dados (como no sistema CODA). A vantagem da utilização de cache reside no fato de liberar os servidores de freqüentes requisições de acesso. Os servidores não são acessados por procedimentos de leitura e escrita, mas somente durante as operações de abertura e fechamento de arquivos. Neste caso, todo o arquivo é transferido para a cache da estação cliente no instante da sua abertura.

A adoção do princípio da minimização da utilização e modificações de informações globais está baseado no fato de que à medida que o sistema cresce torna-se mais difícil estar ciente da situação (estado) global do SOD e modificá-la consistentemente. Um exemplo que demonstra a adoção deste princípio é o fato de que as estações de trabalho apenas monitoram o status dos

servidores dos quais elas possuem dados armazenados na cache.

O mapeamento entre a característica extensibilidade e suas características físicas necessárias ao projeto é mostrado na figura 2.5.

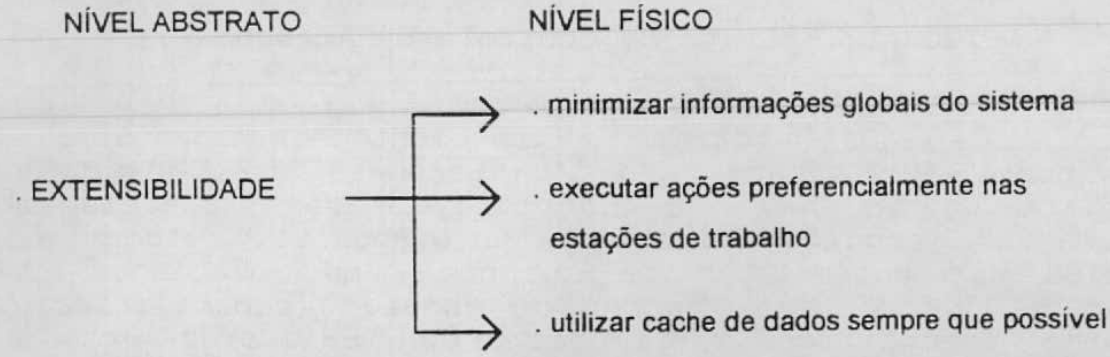

FIGURA 2.5 Mapeamento entre a Característica Abstrata *Extensibilidade* e as Características Físicas no Projeto.

2.4 Segurança de Dados.

Segurança de dados em um sistema possui a função de garantir que os dados armazenados só serão acessados por pessoas autorizadas a fazê-lo. Isto implica em proteger os arquivos, garantir o transporte seguro dos dados na rede e validar um usuário no momento do acesso a fim de certificar-se de sua identidade [SAT 89].

Segurança de dados é importante tanto em ambiente centralizado quanto distribuído. Entretanto, em ambientes distribuídos, problemas adicionais surgem, como é o caso da rede de comunicação que pode ser monitorada de forma deliberada por um usuário mal-intencionado. Um usuário somente vai utilizar um sistema quando sentir confiança na preservação de sua privacidade.

Três componentes provêem segurança aos dados em um sistema de arquivos distribuído [GOS 91]: autenticação do usuário (*user autentication*), segurança de comunicação e autorização do usuário (*user autorization*) ou controle de acesso (*access control*).

SIMBOLOGIA : arquivo: (usuário, grupo do usuário, direitos de acesso)

ARQUIVO1: (Marcos, *, RWX), (Maria, *, RWX)

ARQUIVO2: (*, Funcionário, --X)

ARQUIVO3: (Pedro, Estudante, --X)

ARQUIVO4: (*, Estudante, R--)

FIGURA 2.6 Lista de Controle de Acesso.

Os SADs herdaram as implementações de controle de acesso existentes nos sistemas de arquivos centralizados. As listas de controle de acesso associam a cada objeto (arquivo, diretório ou dispositivo) todos os usuários que podem modificá-lo, bem como relacionam as operações habilitadas sobre o objeto (*capabilities*), conforme exemplifica a figura 2.6. De acordo com a terminologia UNIX, o usuário Pedro possui direito de execução para o arquivo3.

A desvantagem apresentada pelas listas de controle de acesso em ambientes distribuídos é que os servidores de arquivos precisam autenticar os usuários (*user autentication*) a fim de certificarem-se de que em uma requisição de acesso trata-se efetivamente do usuário correto. O sistema Andrew e CODA utilizam listas de controle de acesso e chave de sessão para criptografar a comunicação entre as estações Venus e os servidores VICE [SAT 89]. A chave de sessão é obtida durante a fase de autenticação, no instante em que a estação se conecta ao VICE.

A implementação de segurança através de *capability* consiste no fornecimento, pelo usuário, da *capabality* do objeto que garante o direito de acesso ao arquivo, no instante do seu acesso. Deste modo, o usuário não necessita autenticar-se, bastando fornecer a sua *capability*. Entretanto, o sistema deve garantir que ou a *capability* fornecida não seja modificada ao deixar o servidor, ou que esta modificação possa ser detectada quando do acesso ao arquivo.

A figura 2.7 mostra o mapeamento entre a característica segurança de dados e suas características físicas necessárias no projeto.

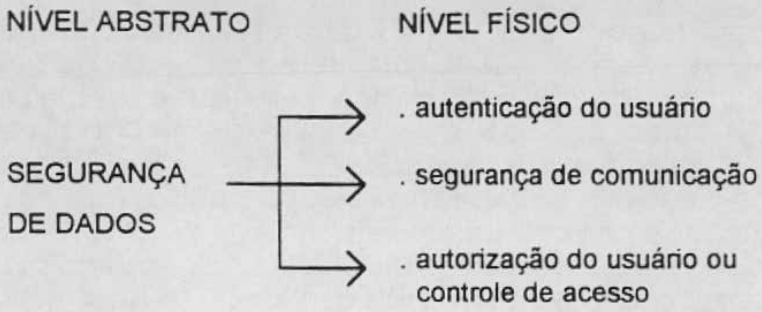

FIGURA 2.7 Mapeamento entre a Característica Abstrata *Segurança de Dados* e as Características Físicas no Projeto.

2.5 Disponibilidade dos Dados.

Um dos principais objetivos dos sistemas operacionais distribuídos, disponibilidade do sistema, não pode ser alcançado se falhas em servidores ou mesmo em conexões na rede de comunicação de dados afetarem constantemente o trabalho dos usuários. Portanto, disponibilidade de dados é a capacidade do usuário dispor de seus dados a despeito de falhas em qualquer componente do sistema [SAT 90a].

Devido à natureza dos ambientes distribuídos, é importante que disponibilidade de dados seja inserida a priori no projeto, pois do contrário, interrupções freqüentes de trabalho perturbam os usuários e o sistema perde credibilidade.

Segundo Satyanarayanan [SAT 90b], elevada disponibilidade de dados em um SAD pode ser obtida através de duas maneiras distintas: **replicação de arquivos** nos servidores e **cache de arquivos** nas estações de trabalho.

Tanto o sistema de arquivos Andrew quanto CODA utilizam replicação de arquivos a fim de obterem maior disponibilidade dos dados. A replicação é transparente aos usuários, encarregando-se o sistema de manter as cópias consistentes e retornar a cópia correta no momento de acesso. Replicação é útil no instante em que um servidor falha, pois os dados afetados continuam disponíveis através de outro servidor. Já cache de dados é usada, freqüentemente, mais para melhorar o desempenho dos sistemas de arquivos do que prover alta disponibilidade.

CODA foi projetado com o principal objetivo de prover disponibilidade de dados. Além do uso de replicação de arquivos, CODA utiliza cache de dados em dois níveis (primário e secundário) nas estações de trabalho. Como os arquivos que estão sendo acessados são mantidos integralmente nas estações de trabalho, CODA pode operar independente de falhas nos servidores ou nos enlaces, em modo de operação desconectada. Neste caso, além de melhorar o desempenho do sistema, a cache mantida por Venus nas estações provê maior disponibilidade dos arquivos.

Quanto à cache de dados, um aspecto importante para a característica disponibilidade de dados é o mecanismo adotado para o controle de concorrência aos arquivos do sistema.

Controle de concorrência (*Concurrency Control*) é um fator, assim como segurança dos dados, que limita o acesso aos dados, permitindo acesso simultâneo a um arquivo e garantindo sua integridade [SVO 84].

Quando diversas transações utilizam um mesmo arquivo simultaneamente, o sistema de arquivos deve garantir que o efeito destas transações transpareça ao usuário como se tivessem sido executadas seqüencialmente, ou seja, garantir que elas sejam serializáveis (*serializability*).

Duas são as técnicas comumente empregadas para atingir esta finalidade [GOS 91]: bloqueio (*locking*) e controle de concorrência otimista (*optimistic concurrency control*).

Bloqueio é o método tradicional para controle de concorrência e consiste em bloquear o acesso do arquivo a outros usuários até que ele seja novamente liberado (*unlock*). Bloqueio apresenta alguns problemas de desempenho, por exemplo: em primeiro lugar, mesmo que durante a utilização do arquivo, ele não venha a ser compartilhado, haverá o custo de bloquear e desbloquear o arquivo. Em segundo lugar, bloquear todo o arquivo durante o acesso, não permitirá a outros usuários, que utilizariam porções distintas, o acesso ao mesmo.

Controle de concorrência otimista surgiu para solucionar os problemas não resolvidos pelo bloqueio e prover melhor desempenho e disponibilidade dos dados. Neste caso, parte-se do princípio de que os arquivos não são, na maioria dos casos, compartilhados, razão pela qual não se

justifica bloquear o acesso a outros usuários. Em conseqüência, os arquivos são normalmente acessados e modificados. A modificação do arquivo, no entanto, provoca a criação de uma nova versão que conterá apenas as porções alteradas (*copy-on-write*). No final da transação, um teste é feito a fim de validar se as transações são serializáveis ou não. Caso não sejam serializáveis, a transação é abortada e, no caso oposto, as alterações passam a ser permanentes. Uma descrição completa de implementação de controle de concorrência otimista pode ser encontrada em [MUL 85].

O sistema de arquivos Andrew, versões posteriores à segunda, também utiliza o controle de concorrência otimista apesar de, ligeiramente diferente do AMOEBA []. O uso de *callback* permite que a estação utilize um arquivo da cache local sem verificar sua validade, deixando que o servidor notifique a estação no caso de uma modificação no arquivo por outro usuário. O controle de concorrência otimista, adotado por AFS-2, influenciou fortemente o desempenho do sistema, conseqüentemente aumentando a transparência do sistema de arquivos.

Deve ficar claro, no entanto, que os dois mecanismos (replicação de dados e cache de dados) não provêm disponibilidade completa de dados quando implementados isoladamente. Por exemplo, se todos os servidores que armazenam determinado arquivo falham ou se os enlaces que dão acesso a eles estiverem interrompidos, os arquivos estarão inacessíveis. Por outro lado, operação desconectada funciona apenas temporariamente.

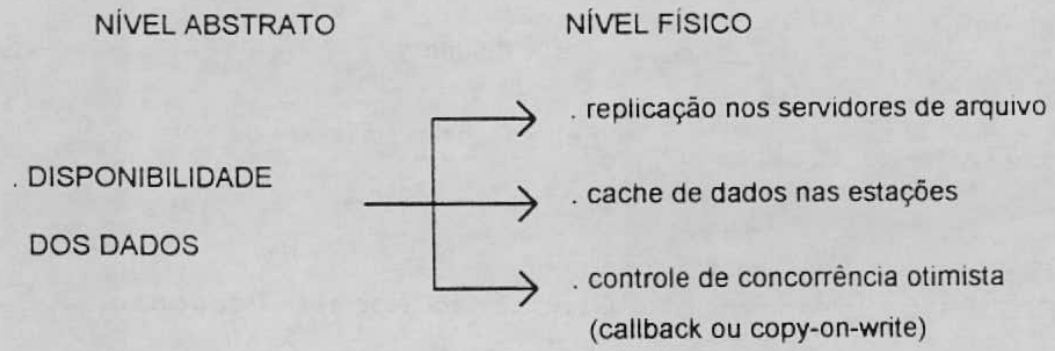

FIGURA 2.8 Mapeamento entre a Característica Abstrata *Disponibilidade de Dados* e as Características Físicas no Projeto.

O mapeamento entre a característica disponibilidade de dados e seus requisitos no projeto é mostrado na figura 2.8.

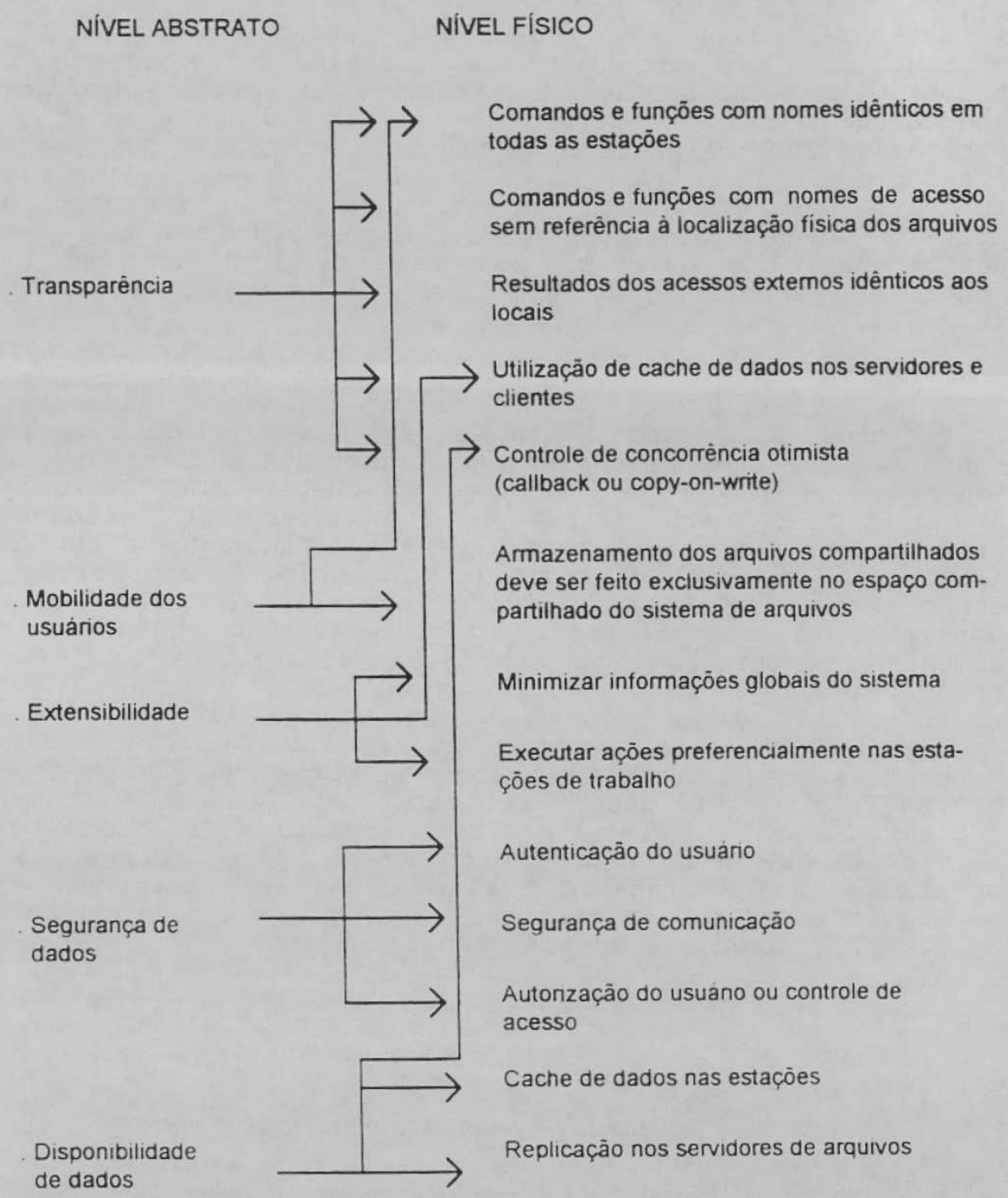

FIGURA 2.9 Mapeamento Global do Modelo Proposto.

Apos a analise individual de cada abstração selecionada, pode-se visualizar de forma mais ampla o modelo apresentado, através da figura 2.9. Atraves deste modelo, pode-se selecionar uma caracteristica abstrata e identificar seus quesitos necessarios no projeto.

3. CONCLUSÃO.

Através do modelo proposto é possível determinar se uma implementação de um sistema de arquivos é voltada para: sistemas operacionais de rede ou sistemas operacionais distribuídos. Para esta finalidade, o analista necessita somente investigar os critérios concretos de implementação apontados no modelo, e não aspectos abstratos de difícil identificação na documentação de cada sistema.

Em virtude da generalização do modelo, ele pode ser aplicado em arquiteturas de sistemas de arquivos implementadas em qualquer ambiente.

Apesar das características abstratas e mesmo as características concretas do modelo terem sido baseadas nos sistemas Andrew e CODA, algumas características concretas necessárias para justificar uma característica abstrata foram detectadas no momento da aplicação do modelo nos sistemas **Unix-to-Unix Copy** (UUCP), **Network File System** (NFS) e **Remote File System** (RFS) [FER 92]. Então, além do modelo ter sido testado diante de implementações comerciais, o mesmo foi refinado durante este processo. A referência acima pode ser utilizada pelo leitor interessado em verificar a aplicação do modelo apresentado.

Dado a aplicação do modelo, entende-se que o mesmo pode ser de extrema valia para analistas que estão envoltos com diversas implementações e não as compreendem completamente. Investigando as características concretas de cada implementação, rapidamente ele poderá verificar se o sistema atenderá a necessidade das suas aplicações.

Propõe-se a elaboração de modelos semelhantes ao desenvolvido neste trabalho, que permitam a avaliação de outros aspectos do sistema operacional, como por exemplo, proteção e gerenciamento de processos. Estes modelos permitiriam uma avaliação mais completa de um sistema operacional.

4. Biliografia.

[BRO 82] BROWNBRIDGE, D. R et all. **"The Newcastle connection-Or UNIXES of the world unite!"**, Software-Pratice and Experience, Vol 12, Dec, 1982.

[GOS 91] GOSCINSKI, Andrzej. **"DISTRIBUTED OPERATING SYSTEMS The Logical Design"**, Addison-Wesley, 1991.

[FER 92] FERREIRA, Marcos A. Masnik. **"Modelo de Avaliação Funcional de Arquiteturas de Sistemas de Arquivos Distribuídos"**, Dissertação de Mestrado, CEFET-PR, 1992.

[HOW 88] HOWARD, John H. et all. **"Scale and Performance in a Distributed File System"**. ACM Transactions on Computer Systems, Vol 6. Nº 1, Feb, 1988.

[MUL 85] MULLENDER, Sape J. & TANENBAUM, Andrew S. **"A Distributed File Service Based on Optimistic Concurrency Control"**, Second Edition, Reading, Addison-Weslev Publishing Company, 1985.

[PET 85] PETERSON, James L. & SILBERSCHATZ, Abraham, **"Operating System Concepts"**, Second Edition, Reading, Addison-Weslev Publishing Company, 1985.

[SAT 89] SATYANARAYANAN, M. **"Integrating Security in a Large Distributed System"**. ACM Transactions on Computer Systems, Vol 7, Nº 3, Aug, 1989.

[SAT 90a] SATYANARAYANAN, Mahadev, et all. **"Coda: A Highly Available File System for a Distributed Workstation Environment"**. IEEE transactions on Computers, Vol 39, Nº 4 Apr, 1990.

[SAT 90b] SATYANARAYANAN, Mahadev. **"Scalable, Secure, and Highly Available Distributed File Access"**. IEEE Computers, May, 1990.

[SAT 91] SATYANARAYANAN, Mahadev, & KISTLER, James J. **"Disconnect Operation in the Coda File System"**. Submitted for publication, Feb, 1991.

[SIL 90] SILBERSCHATZ, Abraham. & LEVY, Eliezer. **"Distributed File System: Concepts and Examples"**. ACM Computing Surveys, Volume 22, Nº 4, Dec, 1990.

[SUM 89] SUMMERS, R. C. **"Local-area Distributed Systems** IBM Systems Journal, Vol 28, Nº 2, 1989.

[SVO 84] SVOBODOVA, Liba. **"File Servers for Network-Based Systems"**, Computing Surveys, vol 16 Nº 4, Dec 1984.

[TAN 87] TANENBAUM, Andrew S.,**"OPERATING SYSTEMS Design and Implementation"**, Prentice-Hall International Editions, 1987.

[TAN 90] TANENBAUM, Andrew S. et all, **"AMOEBA"**. Communications of the ACM, Volume 33, Nº 12, Dec, 1990.

[WIL 90] WILDGRUBER, R.**"Ein Vergleich von Munix/Net NFS und RFS"** Chip Plus 3, Mär, 1990.