AULA 5

INTEGRAÇÃO DE SISTEMAS LEGADOS

Prof. Rodrigo Cantú Polo

CONVERSA INICIAL

Nesta aula, no primeiro tema, serão explicados os tipos diferentes de arquitetura de sistemas legados. No segundo tema, será discutida a migração de sistemas legados. O terceiro tema tratará da migração de dados de sistemas legados para novos sistemas. No quarto tema, serão discutidas estratégias de migração de dados e, no quinto tema, serão tratados os desafios relacionados a uma migração de dados.

CONTEXTUALIZANDO

A aula passada tratou da reengenharia de sistemas legados, cujo objetivo era aprimorar um sistema legado existente. Nesta aula, serão descritas abordagens de substituição de sistemas legados. Antes de entrar em detalhes sobre como fazer isso, os tipos de arquiteturas diferentes de sistemas legados serão descritos. Essas arquiteturas diferentes determinam a dificuldade de modificação de um sistema legado, e é importante constatar em qual dessas arquiteturas um sistema legado se situa, para que seja possível avaliar posteriores modificações a serem feitas na sua estrutura.

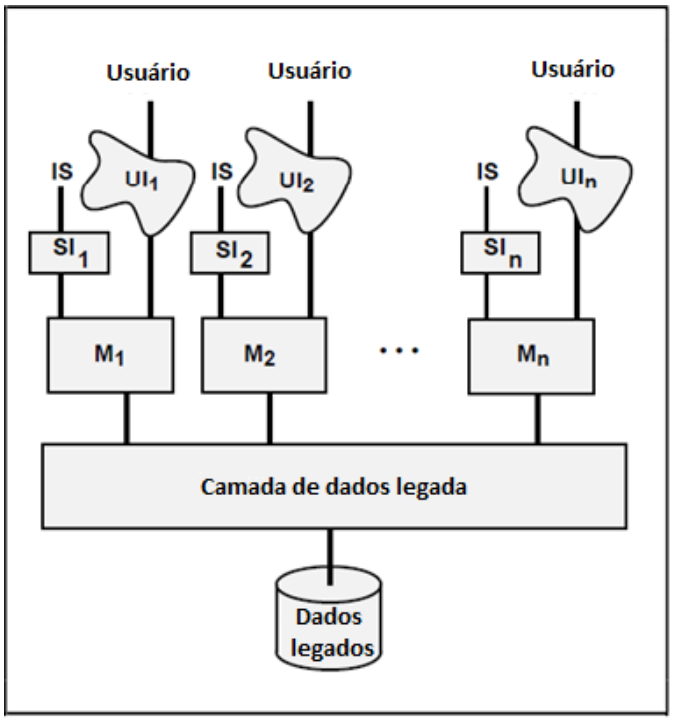
Os tipos de migração de sistemas legados serão abordados em seguida. Como a substituição de sistemas legados é um processo complexo, essas abordagens consideram esse problema. Depois, técnicas de migração de dados entre o sistema legado e o sistema alvo serão discutidas. A migração de dados é um processo extremamente importante e complexo: importante, porque envolve a continuidade de dados críticos para a organização; complexo, porque envolve a compreensão da estrutura dos dados originais, para que então esses dados possam ser migrados para o novo sistema, sem perda de dados. Esse é um tema que continuará a ser tratado na próxima aula.

TEMA 1 – ARQUITETURAS DE SISTEMAS LEGADOS

Todos os sistemas de informação podem ser classificados por terem três camadas: de interface, de aplicação e de acesso a dados. As arquiteturas de sistemas legados variam de bem estruturadas (modulares, fáceis de separar as camadas) a não estruturadas (sem separação em camadas). Segue a classificação dessas arquiteturas:

- **Sistema legado separado em camadas:** as camadas estão bem separadas. Isso facilita a manutenção do código, além de facilitar a substituição de componentes situados nessas camadas. Por exemplo: a camada de interface pode ser modificada sem que o código das camadas de aplicação, ou de dados, sejam afetadas. Da mesma forma, alterações na camada de dados não afetam necessariamente as camadas superiores. A figura 1 descreve um sistema com essa arquitetura, no qual UI significa "interface com o usuário", SI significa "interface com sistemas", IS significa "*information system*" ("sistema de informação") e M significa "módulo".

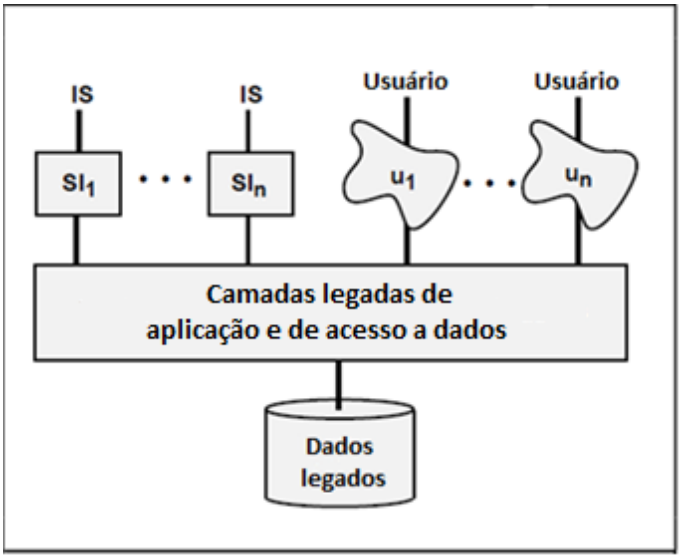
Figura 1 – Sistema legado separado em camadas



Fonte: Brodie (1993)

- **Sistema legado parcialmente separado em camadas**: Alterações na camada de aplicação afetam a camada de dados e vice-versa, formando apenas uma camada. A camada de interface fica isolada. Apesar do alto acoplamento entre processamento e dados, a manutenção do tratamento de clientes de tipos diferentes fica mais fácil. A figura 2 ilustra essa arquitetura.

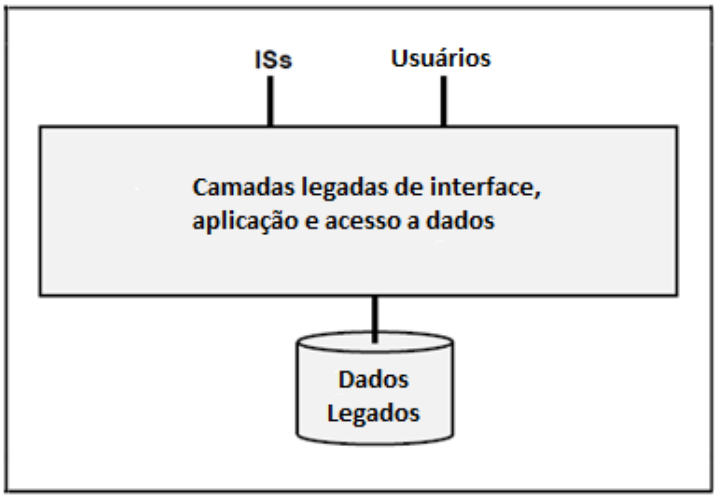
Figura 2 – Sistema legado parcialmente separado em camadas



Fonte: Brodie (1993)

- **Sistema legado sem separação em camadas (monolítico):** existe apenas uma camada. A manutenção desse tipo de sistemas é complexa, pois uma alteração na interface pode afetar o tratamento de dados, e vice-versa. A figura 3 ilustra uma arquitetura monolítica.

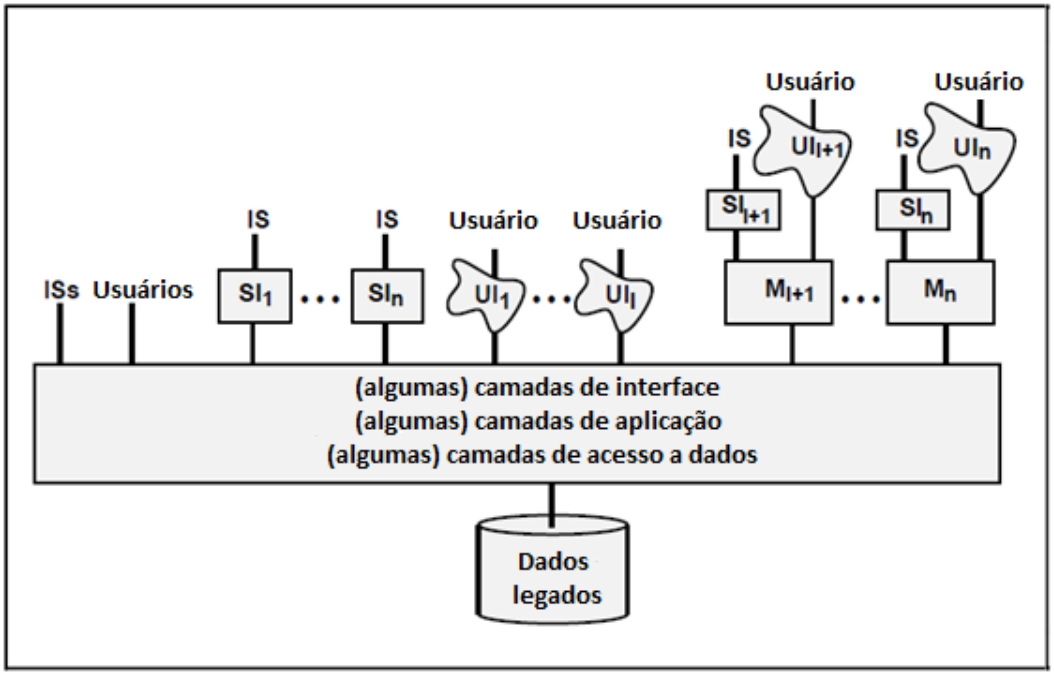
Figura 3 – Sistema legado monolítico (sem camadas)



Fonte: Brodie (1993)

Em geral, a arquitetura da maioria dos sistemas legados pode não ser nenhuma das arquiteturas apresentadas. Durante sua longa evolução, um sistema legado pode ter tido partes adicionadas, que podem ser classificadas de acordo com as arquiteturas anteriormente descritas, como ilustrado na figura 4.

Figura 4 – Sistema legado híbrido



Fonte: Brodie (1993)

TEMA 2 – MIGRAÇÃO DE SISTEMAS LEGADOS

Para que o processo de migração dos sistemas legados para uma nova tecnologia tenha sucesso, é necessário ter um plano, ou estratégia, para essa migração, uma vez que os sistemas legados não podem parar. O sucesso de um projeto de migração de sistemas legados depende do desenvolvimento de uma metodologia que seja detalhada e bem definida, já que esse tipo de projeto é de enorme escala, complexidade e muito suscetível a falhas. Atualmente, duas metodologias são mais aceitas pela comunidade: *Chicken Little* e *Butterfly* (Mendonça, 2009).

2.1 *Chicken Little*

Na metodologia *Chicken Little*, propõe-se uma estratégia de migração composta de onze etapas genéricas que utilizam complexos *gateways*. Um *gateway* pode ser definido como qualquer módulo de *software* introduzido entre componentes de *software* com o objetivo de mediá-los. Essa metodologia deixa a cargo da equipe escolher qual método de migração utilizar: o método *Forward* ou *Database First* ou o método *Reverse* ou *Database Last*.

O método de migração *Forward* ou *Database First* envolve a migração inicial dos dados legados para uma nova base e, em seguida, a aplicação e as

interfaces são migradas incrementalmente. Um *forward gateway* é utilizado para que a aplicação original acesse os dados na plataforma destino.

O método de migração *Reverse* ou *Database Last* faz com que a aplicação seja migrada gradualmente para a plataforma destino, enquanto que a base de dados permanece na plataforma original. A migração dos dados é a última etapa desse processo. Um *reverse gateway* é utilizado para que a aplicação destino acesse os dados da plataforma original.

Nesses dois métodos, os sistemas de origem e destino operam paralelamente durante o processo da migração, o que adiciona complexidade. Os próprios autores dos métodos reconhecem que manter a consistência dos dados entre sistemas de informação heterogêneos representa um problema técnico complexo e que ainda não foi proposta uma solução geral. Os onze passos que compõem a metodologia *Chicken Little* são (Mendonça, 2009):

1. Analisar o sistema legado.
2. Decompor a estrutura do sistema legado.
3. Projetar a interface destino.
4. Projetar a aplicação destino.
5. Projetar a base de dados destino.
6. Instalar o ambiente destino.
7. Criar e instalar os *gateways* necessários.
8. Migrar as bases legadas.
9. Migrar as aplicações legadas.
10. Migrar as interfaces legadas.
11. Mudar para o sistema destino.

2.2 *Butterfly*

Ao contrário da metodologia *Chicken Little*, a metodologia *Butterfly* questiona a necessidade de interoperabilidade entre as aplicações. Ela leva em consideração o fato de que o sistema legado não estará em produção enquanto o processo de migração ocorre. A metodologia propõe o desenvolvimento de um motor de migração de dados, a fim de que o sistema legado fique fora do ar o mínimo possível. Ela difere da metodologia *Chicken Little*, visto que o sistema legado deve ficar fora do ar por um tempo considerável para facilitar a migração dos dados antes de o sistema destino entrar em funcionamento. Essa é uma abordagem mais simples, porém mais arriscada, pois todo o fluxo de informação da aplicação passará a ser executado em um sistema não confiável. Os passos que compõem a metodologia são (Mendonça, 2009):

1. Planejar a migração.
2. Entender a semântica do sistema legado e desenvolver o esquema de dados de destino.

3. Migrar todos os componentes, exceto os dados, do sistema legado para a arquitetura destino.
4. Gradualmente migrar os dados legados para o sistema destino e treinar os usuários a usar a aplicação destino.
5. Passar a utilizar a aplicação destino.

Na fase 2, quando se destaca a importância de entender a semântica do sistema legado, geralmente se faz com base no negócio atual da empresa. É incomum, e na maioria dos casos faltam fontes de informação, levantar requisitos históricos, ou seja, procurar saber como o sistema legado funcionava em suas diversas versões e em como ele usava e atualizava os dados da empresa naqueles momentos. Essa perda de memória institucional é, também, uma fonte de problemas quando se precisa definir as necessárias transformações sobre os dados legados para migrá-los para o novo sistema, principalmente quando se precisa migrar dados históricos. O quadro 1 apresenta uma comparação entre as metodologias *Chicken Little* e *Butterfly*.

Quadro 1 – Quadro comparativo entre as metodologias Chicken Little e Butterfly

Características	Chicken Little (Forward)	Chicken Little (Reverse)	Butterfly
Necessidade de gateways	Sim	Sim	Não
Complexidade	Alta	Alta	Média
Sistema legado fora do ar	Não	Não	Sim
Dados migram primeiro	Sim	Não	Sim
Porte da migração	Médio / Grande	Médio / Grande	Pequeno
Interoperabilidade entre as aplicações	Sim	Sim	Não

Fonte: Mendonça (2009)

TEMA 3 – MIGRAÇÃO DE DADOS

Frequentemente, os dados de uma nova aplicação vêm de outras aplicações existentes. Caso esses dados estejam disponíveis em sistemas existentes e seu volume seja muito grande, eles devem ser migrados das aplicações existentes para a nova aplicação, em vez de recriados. O processo de transferir dados de um sistema para outro é chamado de migração de dados.

Projetos de migração de dados são complexos, e o foco deve ser a migração da menor quantidade de dados possível para que a aplicação destino possa entrar em funcionamento. Nem sempre todos os dados de origem serão necessários, logo, é importante filtrar os dados relevantes. Essa análise de relevância dos dados deve ser realizada em conjunto com os usuários que serão

impactados diretamente pela migração. Porém, em alguns casos, por força de lei ou imposição do negócio, todos os dados devem ser migrados e esses cenários de migração são os que requerem mais planejamento e esforço.

A migração de dados tipicamente envolve o planejamento do projeto, a definição do escopo, a extração, a transformação (para o formato adequado), a transferência e a verificação dos dados. É geralmente desenvolvida em duas etapas: a extração e a carga dos dados.

A extração de dados é o processo de extrair dados de um sistema existente e armazená-lo em um arquivo. Se o volume dos dados for relativamente pequeno, eles podem ser extraídos para um arquivo que será referenciado diretamente pela aplicação destino. No entanto, se o volume dos dados for grande e os sistemas utilizam diferentes ambientes computacionais, eles devem ser armazenados em alguma mídia, e então, carregados na aplicação destino.

A carga de dados é o processo de transferir o conteúdo do arquivo gerado na etapa de extração para a base de dados da aplicação destino. Se os domínios dos dois sistemas possuem uma interpretação comum entre valores e os relacionamentos entre os dados são bem definidos, então o processo de mapeamento é relativamente direto. Caso os domínios sejam diferentes ou os relacionamentos não estejam bem definidos, é necessário passar por um processo de transformação dos dados, que pode ocorrer na etapa de extração ou de carga (Mendonça, 2009).

TEMA 4 – ESTRATÉGIAS DE MIGRAÇÃO DE DADOS

Migração de enormes volumes de dados não ocorrem rotineiramente na maioria das empresas. Geralmente, a área de TIC da empresa não possui vasta experiência nesse tipo de operação. Para realizar um projeto de migração de dados, é necessário que uma estratégia de migração seja adotada no início do projeto, já que seu planejamento e as ações necessárias dependem dessa estratégia. Há dois tipos principais de estratégias de migração de dados: *Big Bang* e *Trickle* (Mendonça, 2009).

4.1 *Big Bang*

As migrações que adotam a estratégia *Big Bang* são realizadas de uma só vez. Nesse caso, o sistema original deve ficar fora do ar enquanto os dados são extraídos, transformados e carregados na aplicação destino. Essa estratégia tem

a vantagem de a migração ser finalizada no menor tempo possível. Porém, ela apresenta riscos: algumas organizações não podem ter o seu sistema fora do ar por muito tempo, o que aumenta a grande carga de pressão sobre a execução da migração e a realização de seus testes. Empresas que adotam essa estratégia deveriam realizar uma migração de testes antes da migração real, mas poucas o fazem, o que compromete a qualidade dos dados. Normalmente, quando essa estratégia é adotada, o processo de execução da migração ocorre durante um fim de semana ou feriado (Mendonça, 2009).

4.2 *Trickle*

As migrações que adotam a estratégia Trickle são realizadas de forma incremental. Os dois sistemas executam em paralelo e os dados são migrados em fases. Isso pode ser desenvolvido utilizando processos em tempo real para transferir os dados da base de origem para a base destino e para manter os dados da base destino atualizados.

Essa estratégia, que adiciona complexidade ao projeto quando adotada, possui duas possíveis maneiras de ser desenvolvida: na primeira, os dois sistemas executam paralelamente e os usuários devem chavear entre as aplicações, dependendo de onde está a informação que precisam no momento. Na segunda, os usuários utilizam o sistema antigo até que a migração termine, porém, quaisquer mudanças nos dados da aplicação fonte exigem que os novos registros sejam migrados, de modo que a base destino permaneça atualizada. A quadro 2 apresenta uma comparação entre as estratégias *Big Bang* e *Trickle* (Mendonça, 2009).

Quadro 2 – Quadro comparativo entre as metodologias *Big Bang* e *Trickle* (Mendonça, 2009)

Características	Big Bang	Trickle
Necessidade de sincronismo	Não	Sim
Complexidade	Média	Alta
Sistema legado fora do ar	Sim	Não
Dados migram primeiro	Sim	Não
Porte da migração	Pequeno	Médio / Grande
Interoperabilidade entre as aplicações	Não	Sim
Compatibilidade com as metodologias de migração de sistemas legados	Butterfly	Chicken Little

Fonte: Mendonça (2009)

4.3 Passos de uma migração de dados

Migrações de dados são realizadas frequentemente, porém, nem todas obtêm êxito. Por meio de migrações de sucesso, boas práticas foram compiladas, de modo que é possível enumerar uma sequência de passos que ajudam a aumentar a probabilidade de sucesso do projeto. Os passos necessários são destacados a seguir (Mendonça, 2009).

4.3.1 Planejamento

O processo de planejamento envolve descrever em detalhes o escopo do projeto, determinar os requisitos da migração, identificar os ambientes atual e futuro, criar e documentar o plano de migração e definir um cronograma a ser seguido. Os requisitos e projeto incluem a arquitetura de migração, os requisitos de *hardware* e *software*, o procedimento de migração e os planos de teste e implantação. O plano de migração deve determinar se a migração será realizada em fases, incluindo quantas delas serão necessárias e que tipos de dados serão migrados em cada uma. Deve também determinar o responsável por cada tarefa e seus prazos, além de descrever o impacto da migração no negócio em termos de necessidade de treinamento de pessoal, custos envolvidos e o tempo de

parada do sistema. Deve ainda estabelecer um plano de contingência, especificando como lidar com processos de negócio críticos em termos de tempo, como pagamentos, e em caso de atraso no cronograma. Quanto maior a importância dos dados para as operações da empresa e maior a complexidade do ambiente, mais crítico é o planejamento da migração.

O cronograma e o orçamento devem levar em consideração todo o material e tempo necessários para auditar os dados, desenvolver as especificações de mapeamento, escrever o código de migração, construir as regras de transformações e limpeza dos dados, carregar e testar a migração. O cronograma também deve incluir quando o dado estará pronto para análise e teste, quando o sistema de origem ficará fora do ar (dependendo da metodologia adotada) e quando o novo sistema estará pronto para os usuários. Um projeto de migração típico tem uma duração média de seis meses a dois anos com uma equipe de cinco desenvolvedores em tempo integral (Mendonça, 2009).

4.3.2 *Profiling* e auditorias

Quando dados são migrados para um novo sistema, pode ficar aparente que eles contêm redundâncias e imprecisões. Enquanto esses dados são perfeitamente adequados para o funcionamento do sistema original, eles podem não ser adequados em termos de estrutura e conteúdo para a nova aplicação. Sem o entendimento suficiente de ambos os sistemas, a transferência de dados de um para o outro pode ampliar o impacto negativo de qualquer dado incorreto ou irrelevante.

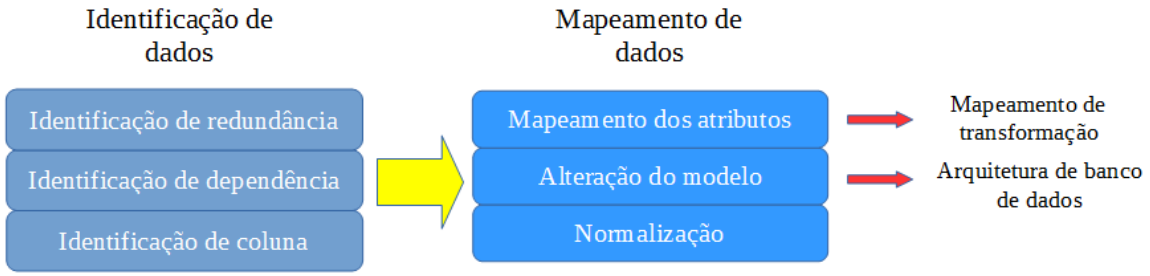
Para desenvolver um projeto de migração de dados eficaz, é importante entender por completo as fontes de dados antes de especificar o código de migração. Isso é mais bem alcançado por meio da realização de uma auditoria completa dos dados que fazem parte do escopo da migração, nos estágios iniciais do projeto. Por intermédio do uso de ferramentas de *profiling* e auditoria, é possível analisar o conteúdo dos dados e identificar quais valem a pena ser migrados, já que uma migração requer tempo, dinheiro e esforço. O principal resultado alcançado com a análise dos dados é o refinamento do escopo e a definição do que será migrado automaticamente, anualmente e o que não será migrado. Os benefícios trazidos por essa prática são:

1. Com uma visibilidade completa de todos os dados de origem, a equipe pode identificar potenciais problemas que permaneceriam escondidos até um estágio mais avançado do projeto.
2. As regras para planejamento, mapeamento, execução e teste da migração são baseadas na análise de todos os dados de origem em vez de uma pequena amostra.
3. As decisões podem ser realizadas baseadas em fatos em vez de suposições.
4. Uma auditoria completa dos dados pode reduzir o custo de reparos no código na etapa de testes em até 80%.

A seguir, será considerada uma estratégia de *profiling* e mapeamento. A figura 5 mostra os passos para identificação e mapeamento de dados. Nessa estratégia são adotados seis passos (Mendonça, 2009):

- **Identificação de coluna:** essa fase procura analisar os valores em cada coluna ou campo da fonte de dados, inferindo características como tipo do dado, tamanho, domínio, frequência, distribuição, cardinalidade, nulidade e unicidade.
- **Identificação de dependência:** essa fase procura identificar dependências entre colunas de diferentes fontes de dados. Geralmente, não é realizada manualmente.
- **Identificação de redundância:** essa fase compara dados entre tabelas das mesmas fontes de dados ou diferentes, determinando que colunas contenham sobreposição ou conjuntos idênticos de valores. Ela procura padrões de repetição. Essa fase também procura identificar atributos que contêm a mesma informação, mas com nomes diferentes (sinônimos) e atributos que têm o mesmo nome, mas informações diferentes (homônimos). Também ajuda a determinar que colunas sejam redundantes e podem ser eliminadas. Esse processo não pode ser realizado manualmente.
- **Normalização:** procede a normalização dos dados baseada no modelo relacional.
- **Alteração do modelo:** faz a adequação do modelo ante a novas necessidades de informação detectadas.
- **Mapeamento dos atributos:** define regras de transformação entre atributos fonte e atributos alvo.

Figura 5 – Passos para identificação e mapeamento de dados



Fonte: Mendonça (2009)

4.3.3 Construção e *design*

Nessa etapa são desenvolvidas as especificações de mapeamento. Projetos de migração são mais eficientes quando segmentados, pois os dados podem ser auditados, mapeados, testados e transferidos em fases, facilitando o seguimento do cronograma, orçamento e possibilitando a realização de revisões de progresso. Uma vez que as especificações de mapeamento forem convertidas em código de migração, elas devem ser verificadas individualmente, a fim de identificar possíveis erros (Mendonça, 2009).

4.3.4 Execução

Nesse passo, os dados são extraídos da fonte, transformados e carregados na aplicação destino utilizando regras de migração carregadas em uma ferramenta de migração escolhida durante a etapa de planejamento (Mendonça, 2009).

4.3.5 Testes e validação dos dados

Apesar de os dados terem sido validados ao longo do processo de migração, testes unitários, de sistema e de carga devem ser feitos para garantir que todos os dados foram migrados e que o novo sistema se comporta como esperado. Caso os passos dois e três não tenham sido realizados de maneira eficaz, a chance da ocorrência de erros aumenta, acarretando em repetição de trabalho, o que leva a um aumento no custo do projeto e um possível atraso no cronograma (Mendonça, 2009).

TEMA 5 – DESAFIOS DE UMA MIGRAÇÃO DE DADOS

Os principais desafios impostos por um projeto de migração de dados são:

- A necessidade de minimização do tempo em que o sistema ficará fora do ar, pois normalmente a organização congela a entrada de dados enquanto realiza a migração dos dados;
- A necessidade de aquisição e instalação de *software* de migração de dados em servidores;
- A ocorrência de inconsistência de valores: isso acontece quando múltiplos sistemas de origem possuem o mesmo conceito (entidade), mas as representações (valores) variam (e.g. em um sistema o campo que armazena números de telefone utiliza o formato XXXXXXXX, já em outro sistema, o mesmo campo utiliza o formato (XX) XXXX-XXXX);
- A necessidade de preservação da integridade referencial entre as entidades e relacionamentos da aplicação destino durante o processo de carga;
- A sincronização dos dados: apesar de eles serem migrados do sistema original para o sistema destino, o sistema original pode continuar em operação. Para preservar a integridade dos dados, quando eles forem adicionados e/ou modificados no sistema original, também devem ser adicionados e/ou alterados no sistema destino;
- Necessidade de escrita das especificações e códigos de mapeamento orientada a conteúdo, em vez de orientada a metadados. Para o propósito de migração de dados, a informação que descreve a origem (e.g. o nome da base de dados, o nome da tabela, o nome da coluna) e as características de cada coluna (e.g. o tipo, o tamanho, a escala, a precisão) é considerada metadado. Conteúdo é o que está contido em cada registro da coluna. Uma migração de dados orientada a metadados assume que o conteúdo reflete sua descrição, o que nem sempre acontece. Somente a realização de *profiling* e auditorias pode confirmar de fato o conteúdo do registro;
- Possibilidade de perda e corrupção de dados durante o processo;
- Alocação insuficiente de tempo para a realização de testes;
- Cronograma imprevisível: mesmo com um planejamento meticuloso, situações inesperadas podem surgir durante a migração, ocasionando problemas que impactam no cronograma (Mendonça, 2009).

FINALIZANDO

Nesta aula, foram apresentadas as diferentes arquiteturas de sistemas legados. Depois, foram apresentados dois tipos diferentes de migração de sistemas legados. Nos temas seguintes, foi discutida a migração de dados de sistemas legados, descrevendo metodologias, estratégias e desafios de migração de dados de sistemas.

REFERÊNCIAS

MENDOÇA, M. H. R. de. **Metodologia de migração de dados em um contexto de migração de sistemas legados**. 2009, 151 f. Dissertação (Mestrado em Ciências da Computação) – Universidade Federal de Pernambuco, Recife, 2009. Disponível em: <http://repositorio.ufpe.br/bitstream/handle/123456789/1934/arquivo1908_1.pdf?sequence=1&isAllowed=y>. Acesso em: 24 out. 2017.

BRODIE, M. L.; STONEBRAKER, M. DARWIN: On the Incremental Migration of Legacy Information Systems. **Relatório Técnico TR-022-10-92-165 GTE Labs Inc**. California, 1993. Disponível em: <http://db.cs.berkeley.edu/papers/S2K-93-25.pdf>. Acesso em: 24 out. 2017.