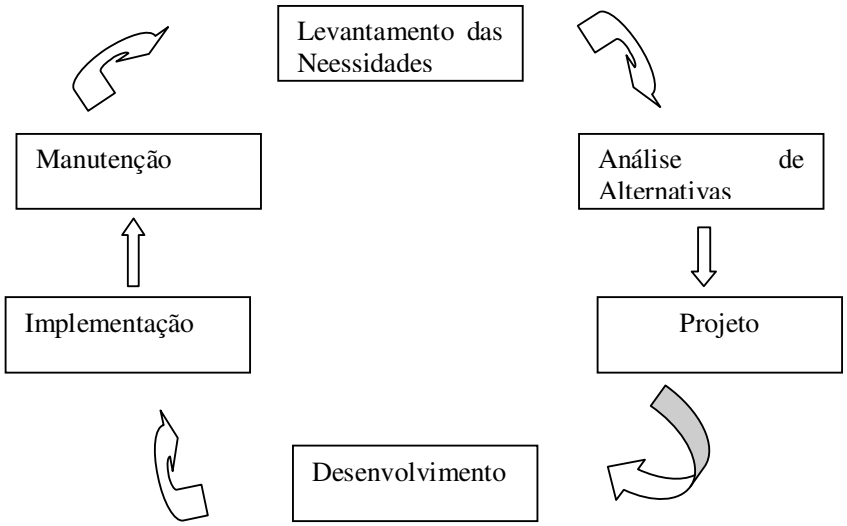
O Ciclo de Vida do Desenvolvimento de Sistemas[i]

O **Ciclo de Vida do Desenvolvimento de Sistemas** (SDLC – Systems Development Life Cycle), conhecido também com o "ciclo de vida do software" refere-se aos estágios de **concepção**, **projeto**, **criação** e **implementação** de um SI. Um desdobramento possível para SDLC é mostrado a seguir:



O **levantamento das necessidades** também chamado de **análise de requisitos** , identifica as necessidades de informações da organização.

A **análise de alternativas** consiste na identificação e avaliação de sistemas alternativos.

Projeto trata da construção das especificações detalhadas para o projeto selecionado. Essas especificações incluem o projeto das interfaces, banco de dados, características físicas do sistema, tais como número, tipos e localizações das estações de trabalho, hardware de processamento, o cabeamento e os dispositivos de rede. Deve especificar os procedimentos para **testar** o sistema completo antes da instalação

Desenvolvimento inclui o desenvolvimento ou aquisição do software, a provável aquisição do hardware e o teste do novo sistema.

Implementação ocorre após o sistema ter passado satisfatoriamente por testes de aceitação. O sistema é transferido do ambiente de desenvolvimento para o ambiente de produção. O sistema antigo (se existir) deve migrar para o novo.

Manutenção refere-se a todas as atividades relacionadas a um sistema depois que ele é implementado. Deve incluir atividades tais como a correção de software que não funcione corretamente, a adição de novos recursos aos sistemas em resposta às novas demandas dos usuários,...

Não há modelo de SDLC uniformemente aceito. Alguns modelos combinam desenvolvimento e implementação em uma única etapa. Outros combinam o levantamento e a análise das necessidades também em uma única etapa. Alguns modelos dividem o projeto em projeto lógico e projeto físico.

Desenvolvimento de Sistemas como um Processo

Muitas organizações vêem cada iniciativa de desenvolvimento de sistemas como um **projeto**. As organizações mais sofisticadas, entretanto, tratam o do desenvolvimento de sistemas como um **processo**. Elas reconhecem que, embora os sistemas possam diferir substancialmente uns dos outros, seu desenvolvimento segue um roteiro previsível, bem definido e manejável. Estas organizações aprendem com seus sucessos e erros. Elas criam manuais de instruções sobre temas que auxiliam os administradores e os desenvolvedores a repetir o êxito obtido em atividades bem-sucedidas e a evitar as malsucedidas. Eles gradualmente acumulam um conjunto de ferramentas que auxiliam a institucionalizar, automatizar e auditar as atividades associadas com o desenvolvimento de sistemas. Eles podem medir o sucesso de suas iniciativas de desenvolvimento de sistemas usando métricas e indicadores, tais com tempo para conclusão, a mão-de-obra necessária e as falhas no produto final relacionadas com as características ou complexidade do novo sistema.

As organizações podem usar ferramentas de software de gestão de processos comerciais para auxiliá-las na padronização e melhoria dos seus processos de desenvolvimento de sistemas. A maioria dos produtos de software de gestão de processos enfatiza uma metodologia de desenvolvimento específica. Uma **metodologia** é um conjunto prescrito e

documentado de práticas, ferramentas, documentos, relatórios e, frequentemente, anotações. Os pacotes comerciais de software de gestão de processos fornecem suporte para várias metodologias populares, tais como o RUP (Rational Unified Process). Este software normalmente inclui gabaritos ou modelos para guiar os desenvolvedores na criação de produtos intermediários, tais como esboços, especificações e quaisquer outras saídas associadas com a metodologia que ele suporta.

As organizações às vezes designam um administrador de processos ou bibliotecário de processos para customizar templates ou modelos e tutoriais conforme os padrões da organização e para reunir as métricas e as melhores práticas para a customização da biblioteca de processos.

As organizações que tratam o desenvolvimento de sistemas como um processo frequentemente esforçam-se para aplicar aos seus métodos os princípios da gestão de qualidade total e da melhoria do processo. O Software Engineering Institute (SEI) desenvolveu e popularizou um modelo chamado CMM (Capability Maturity Model) que auxilia estas organizações a medir quão bem elas alcançam estes objetivos e as guia na melhoria de seus processos de desenvolvimento de sistemas.

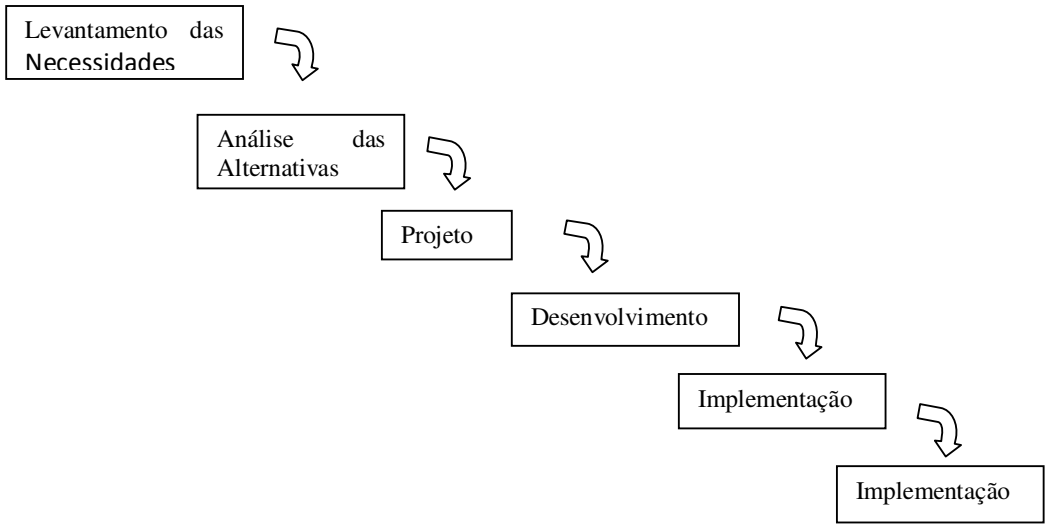
O software de gestão de projetos complementa o software de gestão de processos com ferramentas que os líderes de projetos podem usar como auxílio na administração de um projeto de desenvolvimento de software complexo. O MS Project é um exemplo destas ferramentas.

Caminhos no Desenvolvimento de Sistemas

O SDLC pode parecer sugerir que os novos sistemas sempre progridem de modo regular e seqüencial de um estágio para o seguinte. Na prática, os sistemas nem sempre seguem esta progressão. Os administradores e os profissionais de informática podem mover-se através do SDLC usando o modelo em cascata, a abordagem em espiral, a prototipagem ou a programação ágil.

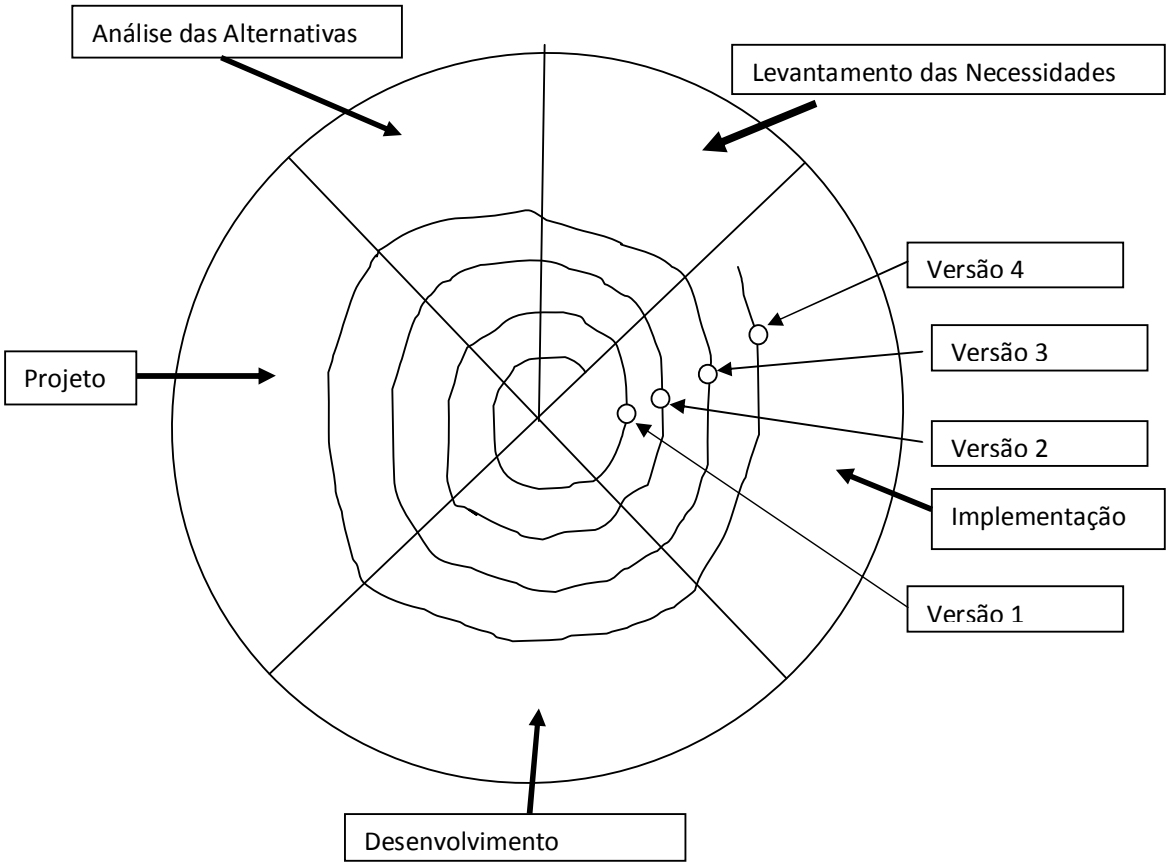
O Modelo em Cascata

O modelo em cascata segue o SDLC em sequência (como mostra a figura a seguir). Como a água que flui numa cascata, o desenvolvimento movimenta-se somente num sentido, de modo que as etapas não podem ser repetidas.



A estrutura linear da abordagem em cascata e a ausência da revisão a tornam relativamente fácil de administrar. O administrador do projeto pode determinar prazos finais e monitorar o progresso na direção destes prazos. Ao mesmo tempo este modelo é muito inflexível. Se, por exemplo, as necessidades dos usuários mudarem durante o projeto, não existe nenhum mecanismo formal para ajustar o processo de desenvolvimento. O uso deste modelo significa, também, que nenhum componente do sistema será entregue até a proximidade final do projeto. Frequentemente esta demora na entrega conduz a tensões entre usuários e desenvolvedores, especialmente se os prazos finais são ultrapassados.

A Abordagem em Espiral



A abordagem em espiral implementa os sistemas baseado no conceito de maior necessidade. Ela entrega o sistema em versões. Cada versão passa por todas as etapas do SDLC, exceto a implementação que pode ser adotada por algumas versões, e a manutenção que se aplica somente a última versão.

A "regra 80/20" dirige a abordagem em espiral: 80% das necessidades dos usuários podem ser satisfeitos por 20% das funções que eles desejam. A primeira versão teta obter um sistema básico que satisfaça a maioria das necessidades do usuário.

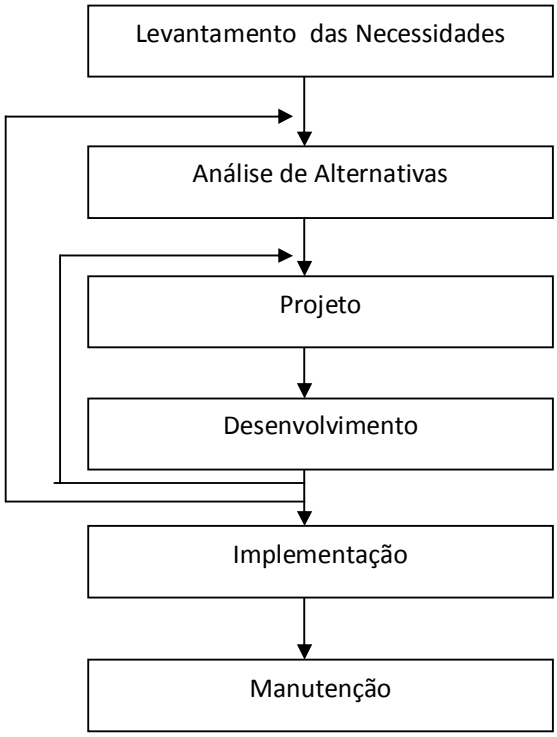
Os partidários da abordagem em espiral frequentemente usam um conceito de *time-box* (caixa de tempo) para regular o desenvolvimento. Um *time-box* é um período determinado, frequentemente, de três meses, dentro do qual os desenvolvedores devem completar cada versão. Não há especificações de produtos, porque satisfazer a estas especificações pode ser impossível dentro do tempo permitido. Em vez disso, os desenvolvedores agregam características pela ordem de prioridade até que o tempo se esgote. Então eles liberam a nova versão.

Uma alternativa para o *time-box* envolve o planejamento de cada versão, tanto quanto possível, no começo do projeto. De fato, isto divide o projeto em subprojetos menores, cada um administrado mais facilmente do que o projeto principal e cada um fornecendo um produto em funcionamento e pronto para a entrega. Ao final de cada projeto, os subprojetos restantes são redefinidos considerando o *feedback* do usuário.

A abordagem em espiral entrega o produto rapidamente. Não há documentação trabalhosa das especificações, porque os usuários podem revisar o produto em versões posteriores. Os usuários podem ver o progresso e julgar quanto tempo passará até que o sistema em desenvolvimento satisfaça suficientemente suas necessidades para que possa substituir o sistema existente.

Prototipagem

A **prototipagem** descreve uma abordagem que tenta satisfazer as necessidades do usuário focalizando a interface do usuário. Os estágios do projeto e de desenvolvimento, no que concerne à interface de usuários, repetem-se até que o usuário esteja satisfeito. A figura abaixo ilustra isso:



Uma variação de prototipagem chamada **Desenvolvimento Conjunto de Aplicações** (**JAD – Joint Application Development**) funciona, basicamente com segue:

Os usuários encontram-se com os desenvolvedores periodicamente, normalmente a cada dia, no início do projeto. Os usuários descrevem suas necessidades durante as reuniões iniciais. Os desenvolvedores de software usam software de prototipagem para cria um protótipo de um sistema que pareça satisfazer estas necessidades antes da próxima reunião conjunta. O protótipo criado pode incluir telas de entrada de dados, relatórios, telas de consulta e outras partes de interface de usuário, mas raramente executa o processamento. Os desenvolvedores as vezes criam dados fictícios para dar a impressãao de um sistema em funcionamento.

A prototipagem oferece diversas vantagens sobre a abordagem em cascata:

- Diminui o tempo entre a análise e a implementação
- Assegura que o novo sistema satisfaça as necessidades do usuário
- Mostra os benefícios de um novo sistema antes que o esforço e os custos se tornem excessivos
- Explora as potencialidades que os usuários têm em articular mais facilmente aquilo de que não gostam em um sistema do que aquilo que apreciam nele.

A prototipagem tem também desvantagens em relação ao modelo em cascata:

- Ela tende a elevar as expectativas dos usuários a níveis que os desenvolvedores não podem atender dentro de seu orçamento.
- Os programas de software que permitem aos desenvolvedores de software desenvolver rapidamente a interface de um novo sistema e de customizá-la rapidamente em atendimento às solicitações do usuário, atualmente, custam caro
- Ela atrasa a demonstração da funcionalidade do sistema. Metade da funcionalidade pode não aparecer até que se atinjam os 10% finais do cronograma de desenvolvimento

Programação Ágil

Programação ágil é um nome dados às diversas metodologias que se concentram na reação rápida às mudanças nos requisitos do usuário e que visam a pequenos grupos de desenvolvimento e projetos que requeiram um mínimo de documentação. Um exemplo de programação ágil que recentemente tornou-se popular é a programação extrema (XP – eXtreme Programming), uma metodologia que junta dois programadores com um dos clientes para o qual o software está sendo desenvolvido. Apesar de contradizer antigas crenças sobre o desenvolvimento de software, a experiência mostra que a XP pode reduzir o tempo de desenvolvimento e os defeitos de software, ao mesmo tempo que aumenta a satisfação entre desenvolvedores e usuários.

A Seleção de um Caminho

A melhor abordagem para um determinado projeto depende, em grande parte, da natureza do projeto e da natureza da organização. A abordagem em cascata funciona melhor com projetos de grande porte, complexos, que têm numerosos interessados, afetam a empresa toda e não podem ser facilmente divididos em subprojetos. Ela também funciona bem com organizações que têm uma cultura formal e uma estrutura hierárquica.

A abordagem em espiral e a programação ágil funcionam bem nas organizações dinâmicas, que podem tolerar a ambiguidade e necessitam obter resultados rapidamente. O caminho em espiral pode apresentar melhores resultados quando adotado para projetos que se dividem facilmente em subprojetos e para projetos mais simples, em especial o desenvolvimento de sistemas de usuário único ou que afetam um pequeno departamento. A programação ágil é bem-sucedida em ambiente onde as necessidades do usuário são difíceis de especificar ou mudam rapidamente.

A prototipagem funciona melhor para projetos de pequeno e médio portes. Ela funciona bem onde a cultura suporta equipes funcionalmente mistas. A prototipagem pode ser combinada com a abordagem em espiral e ser usada para um ou mais dos subprojetos em um desenvolvimento em espiral.

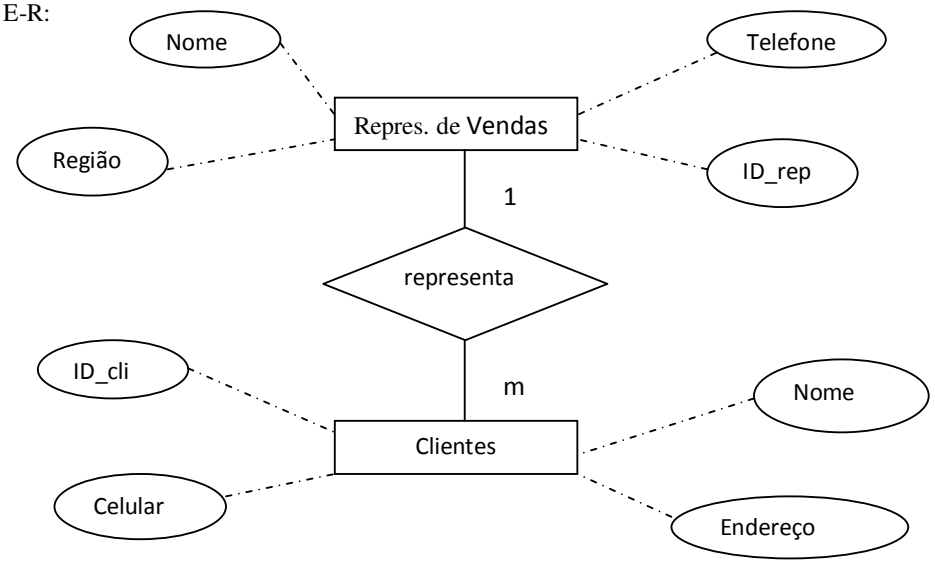
Modelagem de Sistemas

Os desenvolvedores de sistemas usam dados, processos e modelos de objeto para compreender os sistemas existentes e projetar os novos. Estes modelos fornecem uma linguagem que os analistas, os projetistas e os desenvolvedores podem usar para comunicar-se eficientemente. Os administradores de equipes de desenvolvimento de sistemas se beneficiarão da compreensão destes modelos de modo que possam melhor comunicar suas necessidades.

Os produtos de software que geram programas de computador diretamente dos modelos de sistemas podem acelerar, de maneira acentuada, o desenvolvimento de software. Os softwares que geram modelos de sistemas a partir dos programas existentes podem auxiliar os desenvolvedores a compreender e manter estes programas. Muitos produtos suportam, também, a tradução entre modelos do mesmo tipo, por exemplo, de um modelo de dados para outro.

Modelos de Dados

Os modelos de dados descrevem os relacionamentos entre os elementos de dados que uma organização usa. O modelo E-R (Entidade – Relacionamento) é um dos modelos de dados mais extensamente usados. A figura abaixo ilustra um modelo de dados E-R:



Modelos de Processos

Os modelos de processos dividem um processo em suas partes, mostram como estas partes se relacionam entre si e indicam quais saídas de um processo são entradas para outros processos. Os modelos de processos mais populares incluem **diagramas de estrutura** e **diagramas de fluxos de dados**.

Os diagramas de estrutura mostram o relacionamento entre os programas (ou módulos de programas) e subprogramas que compreenderão o sistema acabado. A figura abaixo exibe o diagrama de estrutura de um sistema de folha de pagamento, enfatizando o projeto modular do sistema. A execução de uma determinada tarefa, como o cálculo do pagamento líquido, requer o encerramento de todas as tarefas abaixo (por exemplo , cálculo do pagamento bruto e o cálculo das deduções)

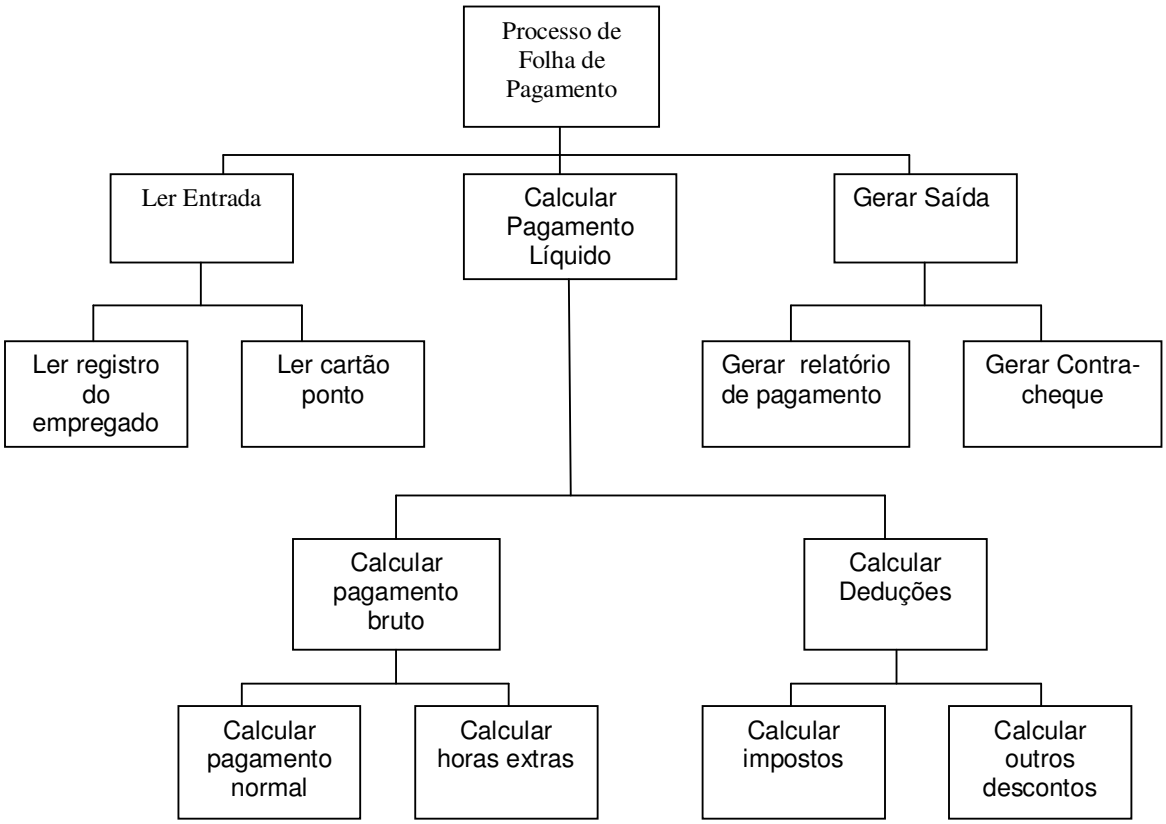
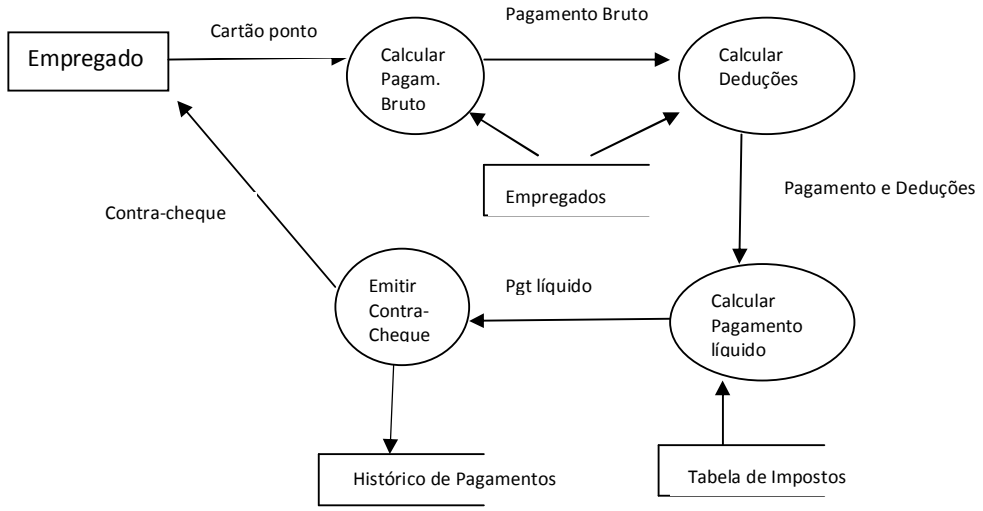


Diagrama de Fluxo de Dados (DFD)

Os DFDs modelam o fluxo de dados entre processos. Eles não modela a ordem em que as tarefas são executadas. A figura abaixo ilustra um sistema simplificado de folha de pagamento, modelado com DFD. As setas indicam o fluxo dos dados; os retângulos abertos lateralmente representam dados armazenados (repositório de dados); os círculos indicam processos e os retângulos fechados representam as fontes das entrada ou os usuários das saídas.



Modelos de Objeto

Os modelos de objetos descrevem as propriedades dos objetos, seus relacionamentos entre si e as funções que executam. Os modelos de objetos incluem, normalmente, os diagramas de herança, que mostram como s objetos herdam suas propriedades de outros objetos. Os modelos de objeto podem incluir diagramas de estado para mostrar como as características do objeto mudam à medida que os eventos externos afetam um objeto e como o objeto responde diferentemente às mensagens, dependendo do seu estado.

Os objetos incorporam tanto os dados quanto as operações que podem ser executadas sobre os dados. Os relacionamentos entre objetos, dados e processos motivaram o desenvolvimento de modelos que incorporam todos os três elementos. Entre os mais populares está a Linguagem de Modelagem Unificada (UML – Unified Modeling Language).

[i] Texto extraído do livro Sistemas de Informação Uma Abordagem Gerencial

Autores: Steven R Gordon & Judith R Gordon

LTC 2006