

GILVAN SOUSA DA SILVA – RA: 6820508946

LEONARDO CÉSAR VALENTINO COSTA – RA: 6247228721

Marcus alves ferreira – ra: 6277261905

rodrigo dourado nunes – ra: 6818457769

SGFarma – sistema de gestão de drogarias

BRASÍLIA

2016

Brasília / DF

2016

gilvan sousa da silva – ra: 6820508946

leonardo césar valentino costa – ra: 6247228721

marcus alves ferreira – ra: 6277261905

rodrigo dourado nunes – ra: 6818457769

sgfarma – sistema de gestão de drogarias

Trabalho de Conclusão de Curso apresentado à Faculdade de Negócios e Tecnologias da Informação – FACNET / Anhanguera, como requisito parcial para a obtenção do título de graduado em Bacharelado em Sistemas de Informação.

Orientador: Luiz Henrique Santana

Brasília

2016

Gilvan sousa da silva – ra: 6820508946

leonardo césar valentino costa – ra: 6247228721

marcus alves ferreira – ra: 6277261905

rodrigo dourado nunes – ra: 68184557769

sgfarma – sistema de gestão de drogarias

Trabalho de Conclusão de Curso apresentado à Faculdade de Negócios e Tecnologias da Informação – FACNET / Anhanguera, como requisito parcial para a obtenção do título de graduado em Bacharelado em Sistemas de Informação.

Aprovado em: \_\_/\_\_/\_\_\_\_

BANCA EXAMINADORA

Professor. Fernando Gonçalves de Oliveira

Professor. Guilherme Parente Costa

Professor. Wígenes Sampaio

Dedico este trabalho...

Ao grupo, professores e familiares

**AGRADECIMENTOS**

Primeiramente a Deus por nos proporcionar saúde e muita inteligência para desenvolvimento deste trabalho.

Aos professores da Anhanguera/Kroton, por suas dedicações em sanarem nossas duvidas e também por nos mostrar um direcionamento.

A instituição como um todo por nos permitirem a realização deste tema e nos fornecerem seus conhecimentos para a realização do mesmo.

Aos nossos familiares por estarem nos apoiando e nos incentivando para a conclusão do curso e no desenvolvimento do nosso trabalho final.

RESUMO

Este projeto tratará de um modelo de desenvolvimento de software de vendas para drogarias, nele serão mostrados processos e padrões utilizados na engenharia de software. No cenário atual as mudanças são muito frequentes isso já faz com que o projeto tenha a necessidade de mostrar as ferramentas de software onde elas sejam capazes de prestar os devidos auxílios aos clientes. Será feita pesquisa para entendimento do tema central do projeto, que é o desenvolvimento de software utilizando boas práticas, com o andamento do projeto será explicado cada padrão por meio dessas mesmas pesquisas. Faremos a aplicação das melhores práticas e padrões utilizados dentro da engenharia de software para solucionar os problemas no momento de uma determinada venda, mostramos a importância de uma boa engenharia de software para a qualidade de um projeto e também veremos como a sociedade vai poder usufruir dos benefícios que serão mostrados no projeto, como as empresas, os usuários de sistemas e até mesmo os estudantes. Esclarecendo assim a importância de se utilizar a engenharia de software no desenvolvimento de um projeto.

**Palavras-Chave: Engenharia De Software; Vendas; Projeto; Modelagem; Desenvolvimento de Software, Padrões De Engenharia de Software**

abstract

This projetct will broach a software development model of sales to drugstores, it will be shown processes and patterns of software engineering. In the current scenario, the changes are very commom, this already makes the project has the necessity to show the software tools where they are able to provide the appropriate aid to customers. Research will be central theme of understanding the project, which is to develop software using best practices, with the progress of the project will become standard explained by these same surveys. We will make the application of best practices and standards used within the software engineering to solve problems at the time of a particular sale, show the importance of good software engineering for the quality of a project and also see how the company will be able to enjoy the benefits that will be shown in the project, as companies, system users and even students. Thus clarifying the importance of using software engineering in the development of a project.

**Keywords: Software Engineering; Sales; Project; Modeling; Software Development, Patterns of Software Engineering.**

**LISTA DE ILUSTRAÇÕES**

[**Figura 1** - Fluxo de Processo 18](#_Toc464027924)

[**Figura 2** - Modelo Cascata - Pressman 19](#_Toc464027925)

[**Figura 3** - Modelo Cascata - Sommerville 20](#_Toc464027926)

[**Figura 4** - Modelo Incremental – Sommerville 21](#_Toc464027927)

[**Figura 5** - Modelo Incremental - Pressman 22](file:///C:\Users\leonn\OneDrive\Documents\GitHub\TCC_2\TCC_2\AVA\TCC%20Online%20-%20Versão%200.6.docx#_Toc464027928)

[**Figura 6** - Modelo Espiral 23](#_Toc464027929)

[**Figura 7** - Desenvolvimento e Evolutivo 24](file:///C:\Users\leonn\OneDrive\Documents\GitHub\TCC_2\TCC_2\AVA\TCC%20Online%20-%20Versão%200.6.docx#_Toc464027930)

[**Figura 8** - Cronograma do Processo Unificado 25](file:///C:\Users\leonn\OneDrive\Documents\GitHub\TCC_2\TCC_2\AVA\TCC%20Online%20-%20Versão%200.6.docx#_Toc464027931)

[**Figura 9** - Eixos do Fluxo de Trabalho do RUP 28](#_Toc464027932)

**SUMÁRIO**

[1 INTRODUÇÃO 13](#_Toc464224664)

[1.1 Problema de Pesquisa 14](#_Toc464224665)

[1.2 Objetivos 14](#_Toc464224666)

[1.2.1 Objetivo Geral: 14](#_Toc464224667)

[1.2.2 Objetivos Específicos: 14](#_Toc464224668)

[1.3 Justificativa 15](#_Toc464224669)

[1.4 Metodologia De Pesquisa 16](#_Toc464224670)

[2 FUNDAMENTAÇÃO TEÓRICA: ENGENHARIA DE SOFTWARE 17](#_Toc464224671)

[2.1 Processos Genéricos 18](#_Toc464224672)

[2.2 Modelo De Processo Prescritivo 19](#_Toc464224673)

[2.3 Modelo Cascata 19](#_Toc464224674)

[2.4 Modelo Incremental 20](#_Toc464224675)

[2.5 Modelo Espiral 22](#_Toc464224676)

[2.6 Processo Unificado 23](#_Toc464224677)

[3 PARADIGMAS DE PROGRAMAÇÃO 25](#_Toc464224678)

[4 RUP 26](#_Toc464224679)

[CONSIDERAÇÕES FINAIS 29](#_Toc464224680)

[REFERÊNCIAS 30](#_Toc464224681)

# INTRODUÇÃO

O projeto mostra um modelo de desenvolvimento de um software para drogarias, feito em um sistema de apoio a decisão na hora de fazer uma venda de determinado produto, foi abordado todas as etapas de desenvolvimento deste sistema, da parte inicial, ou seja, de sua concepção até a parte final que será sua implantação.

Aqui também serão apresentadas as regras de negócio, mostrando assim as necessidades reais para o projeto em si, ainda mais se observamos o cenário atual onde as mudanças são muito influenciadoras e acontecem rapidamente, onde já e possível ver a necessidade de uma ferramenta de software que seja capaz de prestar os devidos auxílios aos usuários.

Ressalta-se que será levando em conta uma serie de padrões mostrados pela engenharia de software, e que também serão a abordagem central nas pesquisas que foram feitas. Com a evolução do projeto, cada padrão apresentado foi explicado por meio de pesquisas, e para isso foi explicado o porquê de se utilizar o determinado padrão, sendo apresentado assim as vantagens de se fazer o uso do padrão em questão.

Neste projeto pretende-se fazer as melhores aplicações práticas encontrado na engenharia de software, para solucionar o problema no momento de se fazer uma determinada venda em um sistema de drogarias, o objetivo e mostrar como a engenharia de software e importante para a qualidade de um projeto. E ainda será possível ver como a sociedade poderá se beneficiar com o desenvolvimento do sistema e a utilização de suas técnicas, como as empresas, os usuários de sistemas e os estudantes. O objetivo e esclarecer para todos que chegarem a ler este projeto a real importância da engenharia de software no momento de se desenvolver um sistema de vendas para drogarias.

## 1.1 Problema de Pesquisa

Atualmente os sistemas de vendas em drogarias apresentam muitos problemas, derivados de sua má concepção e de processos ruins incorporados ao mesmo. Como criar um sistema de venda que satisfaça aos clientes e que atendam as especificidades de seus negócios?

## 1.2 Objetivos

## 1.2.1 Objetivo Geral:

Mostrar o processo de desenvolvimento de um software de vendas para drogarias utilizando padrões da engenharia de software que são mundialmente aceitos.

## 1.2.2 Objetivos Específicos:

* Mostrar técnicas de desenvolvimento de software que melhoram a qualidade do software construído para os processos de vendas em drogarias;
* Apresentar conceitos das tecnologias utilizadas no desenvolvimento do sistema, de modo a justificar a escolha das mesmas;
* Modelar o sistema com base de padrões ditados pela UML;
* Criar funcionalidades capazes de facilitar e otimizar o processo de vendas da drogaria;
* Criar um sistema que forneça informações sobre vendas da drogaria com intuito de melhorar a tomada de decisão;
* Apresentar metodologia de desenvolvimento de um software vendas em drogarias.

Abordaremos neste tópico o respaldo teórico para a metodologia que utilizaremos para desenvolver o sistema de vendas para a drogaria.

## 1.3 Justificativa

As mudanças no atual cenário corporativo estão sempre acontecendo, surge então, a necessidade de as empresas possuírem sistemas de informações que sejam capazes de se adequar a essas mudanças. Portanto, sistemas obsoletos que não sejam flexíveis e escaláveis devem ser na medida do possível substituídos. Queremos assim, deixar mais transparente os benefícios trazidos pela modernização dos sistemas utilizados pelas empresas.

Outro alvo importante deste projeto é a importância da utilização dos padrões engenharia de software, por se tratar de padrões aceitos internacionalmente, que já foram testados inúmeras vezes. Construir um software tem suas particularidades, por ser algo mais lógico do que concreto, a necessidade de ter um controle do projeto é ainda mais fundamental, por meio de prazos e custos bem definidos, documentação clara e abrangente, entre muitos outros fatores que são aqui abordados.

É claro que, uma empresa possuir sistemas de informações atuais e desenvolvidos seguindo todos os padrões não garantirá seu bom desempenho, mas, sem dúvida, facilitará enfrentar este mercado cada vez mais difícil, no qual turbulências podem surgir a qualquer momento, ainda mais levando em conta a atual situação da economia do país.

## 1.4 Metodologia De Pesquisa

Com base na fundamentação teórica, o melhor paradigma para desenvolvimento do sistema é o Orientado a Objetos, então foi exposto a linguagem selecionada e as motivações de seu uso. Para o processo de desenvolvimento será utilizado o Modelo Cascata, que propõe primeiro a documentação do sistema de vendas em drogarias, já que o projeto não é de alta complexidade e como o contato com o cliente não é de extrema necessidade durante o processo de desenvolvimento, será definido os requisitos para o sistema depois os mesmos serão desenhados e organizados segundo os padrões da UML – Unified Modeling Language (Linguagem de Modelagem Unificada).

Estás escolhas se fundamentam também no fato de nem todos os membros da equipe possuírem experiência no desenvolvimento de sistemas e que o desenvolvimento deverá ser acompanhado pelo tutor.

# FUNDAMENTAÇÃO TEÓRICA: ENGENHARIA DE SOFTWARE

Nesta etapa foram abordados os principais conceitos da engenharia de software respaldado por autores reconhecidos no cenário internacional.

Cada vez mais o mundo ganha um dinamismo que influencia muito a vida das pessoas, a forma de se obter informações alcançou um nível que no passado era inimaginável, e os sistemas de informações têm um papel fundamental nesse acontecimento. Dessa forma, buscar formas de otimizar o seu processo de desenvolvimento torna-se cada vez mais necessário, e é esse o principal o objetivo da Engenharia de Software.

Conforme Pressman “A Engenharia de Software é uma tecnologia dividida em camadas (ferramentas, métodos, processo e foco na qualidade) ” (PRESSMAN, 2011, p. 15). “Engenharia de Software é a criação e a utilização de sólidos princípios de engenharia a fim de obter software de maneira econômica, que seja confiável e que trabalhe em máquinas reais. ” (FRIEDRICH LUDWIG BAUER). “Engenharia de Software é uma disciplina cujo foco está em todos os aspectos da produção de software, desde os estágios iniciais da especificação do sistema até sua manutenção...” (SOMMERVILLE, 2013, p. 5)

Há vários exemplos de sistemas, em que foram gastos recursos consideráveis e que não atenderam às necessidades de seus usuários, e ainda por cima, foram entregues fora do prazo estabelecido no cronograma. Esses acontecimentos são muitos graves, haja vista que as organizações estão cada vez mais dependentes dos sistemas de informações. Pressman descreve esta importância: “O software distribui o produto mais importante de nossa era – *a informação*. Ele transforma dados pessoais de modo que possam ser úteis num determinado contexto” (PRESSMAN, 2011, p. 31). Isso deixa muito claro o tamanho da importância da Engenharia de Software.

Um dos pontos mais importantes da Engenharia de Software é a divisão do desenvolvimento de software em fases definidas.

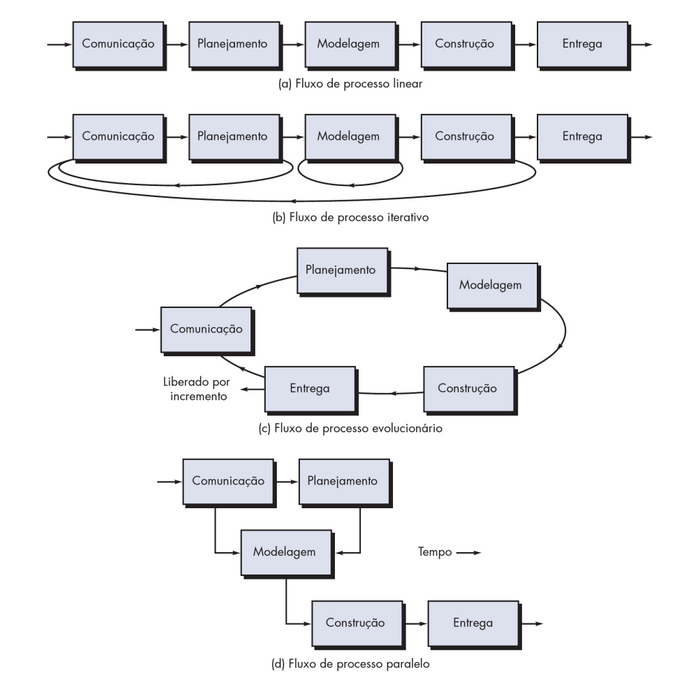
Para Pressman, a Engenharia de Software é dividida em cinco etapas:

* **Comunicação –** objetivos das partes interesssadas são compreendidos para que o levantamento das necessidades possa ser feito com maior eficiência.
* **Planejamento –** nesta etapa são definidas as técnicas a serem conduzidas, o levantamento de riscos, os recursos (humanos, financeiros, equipamentos, etc.) necessários e cronograma de trabalho.
* **Modelagem –** é criado um esboço do sistema a ser produzido a fim melhorar inteligibilidade pelos clientes e profissionais envolvidos no projeto.
* **Construção –** implementação do código, realização de testes a fim de avaliar a qualidade do produto e processo de entrega do sistema ao cliente que fornece feedback.

## 2.1 Processos Genéricos

Pressman defende que essas etapas, que ele chama de processos, possuem vários fluxos, vejamos na figura 1 esses exemplos.

**Figura 1** - Fluxo de Processo



**Fonte**: PRESSMAN (2011, p. 54)

## 2.2 Modelo De Processo Prescritivo

Foram propostos alguns modelos de processos para situações mais específicas, diante de uma enorme necessidade.

“A história tem demonstrado que esses modelos tradicionais proporcionaram uma considerável contribuição quanto à estrutura utilizável no trabalho de equipes de software. Entretanto, o trabalho de engenharia de software e o seu produto permanecem à beira do caos”. (PRESSMAN, 2011, p. 59).

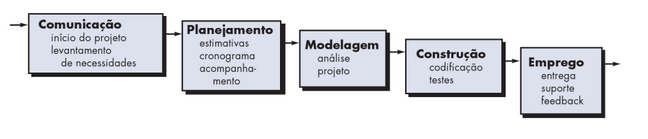
## 2.3 Modelo Cascata

É mais indicado em situações nas quais os requisitos estão em um nível de definição elevado, e que por isso tendem a mudar pouco. Isso é mais comum em mudanças bem definidas e um sistema existente.

O Modelo Cascata “...sugere uma abordagem sequencial e sistemática para o desenvolvimento de software”. (PRESSMAN, 2011, p. 59)

Vejamos na figura 2 como evolui o Modelo Cascata, segundo Pressman.

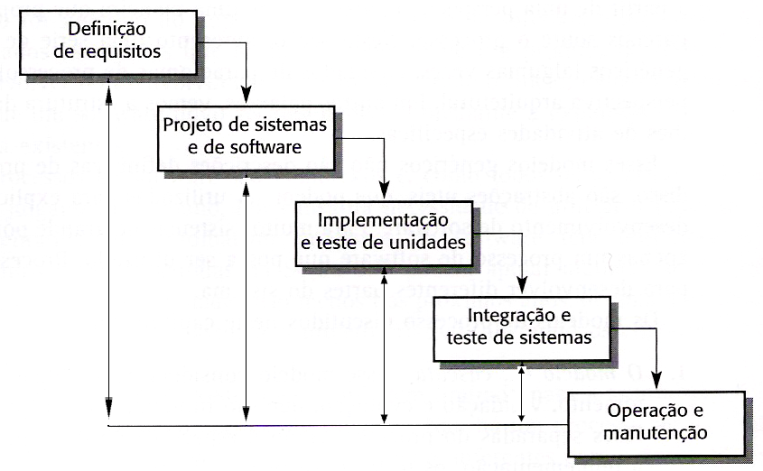
**Figura 2** - Modelo Cascata - Pressman



**Fonte**: PRESSMAN (2011, p. 60)

Vamos ver agora na figura 3 como o Modelo Cascata de acordo com Sommervile.

**Figura 3** - Modelo Cascata - Sommerville



**Fonte**: SOMMERVILLE (2013, p. 20)

O modelo Cascata tem suas qualidades, como organizar e dividir responsabilidades no desenvolvimento de software. Entretanto, tem sofrido inúmeras críticas por não atender a complexidade que é desenvolver um software, ainda mais nos dias atuais.

“Para maioria dos sistemas, esse processo não oferece custo-benefício significativo”. (SOMMERVILLE, 2013, p. 21)

Pressman cita três principais defeitos:

* Na prática os projetos dificilmente seguem um fluxo sequencial;
* É muito comum o cliente ter dificuldade em explicitar suas necessidades;
* Não é gerada nenhuma versão até que o projeto termine.

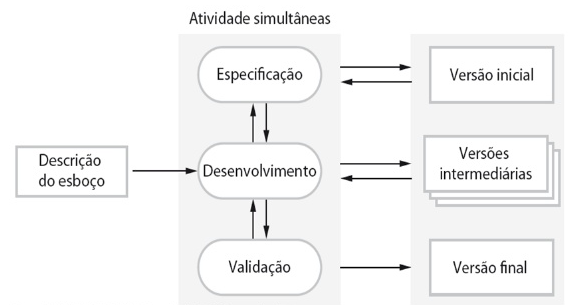
“Na média, 45% das características nos requisitos em cascata nunca são usadas e o cronograma e orçamentos iniciais, em cascata, variam até 400% do que ocorre na realidade”. (LARMAN, 2007, p. 61). Diante de tal afirmação, vemos que estas limitações podem fazer com que este model pode não ser capaz de atender de forma eficitente uma equipe de desenvolvimento de software.

## 2.4 Modelo Incremental

Este modelo foi proposto inicialmente por Mills em 1980 e veio para superar a dificuldade que o Modelo Cascata tem em lidar com mudanças frequentes durante o processo de desenvolvimento de software, principalmente por conta da sua maneira de lidar com os requisitos, é mais flexível, pois considera as mudanças de requisitos algo esperado e normal. “O desenvolvimento incremental é baseado na ideia de desenvolver uma implementação inicial, expô-la aos comentários dos usuários e continuar por meio de criação de várias versões até que um sistema adequado seja desenvolvido” (SOMMERVILLE, 2013, p. 21).

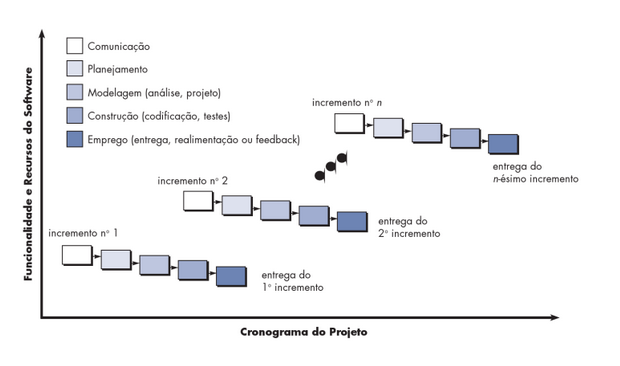
A figura 4 demonstra como o modelo incremental se comporta segundo Sommerville.

**Figura 4** - Modelo Incremental – Sommerville



**Fonte**: SOMMERVILLE (2013, p. 22)

A figura 5 demonstra como o modelo incremental se comporta segundo Pressman.

**Fonte:** PRESSMAN (2011, p. 61)

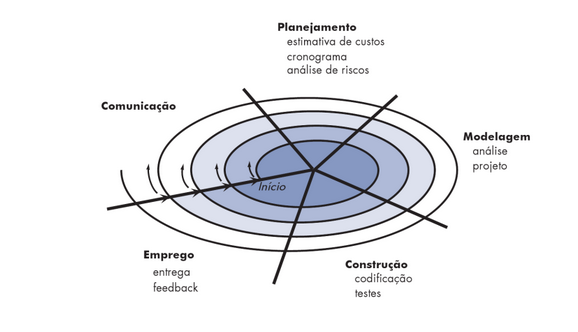
**Figura 5** - Modelo Incremental - Pressman

O modelo incremental se mostra eficaz em muitas situações, porém há algumas específicas em que não é possível obter resultados satisfatórios com sua utilização, como em sistemas de bancários de grande porte, nos quais várias equipes de trabalho têm de estar envolvidas, exigindo dessa forma, uma arquitetura bem definida previamente.

## 2.5 Modelo Espiral

Segundo PRESSMAN 2011 p. 65 “o Modelo Espiral é um modelo de software evolucionário que acopla a natureza iterativa da prototipação com os aspectos sistemáticos e controlado do Modelo Cascata”. Neste modelo o cliente vai recebendo e testando o sistema em várias versões entregues, as primeiras podem ser um protótipo até chegar na versão final.

**Figura 6** - Modelo Espiral



**Fonte**: (Pressman, 2011, p.65)

O Modelo Espiral é eficaz na redução de riscos a que o processo de desenvolvimento de software está sujeito por utilizar a prototipação em qualquer estágio.

Pressman explica o motivo do nome do modelo ser espiral:

“Assim que esse processo evolucionário começa, a equipe de software realiza atividades indicadas por um circuito em torno da espiral no sentido horário começando pelo centro. Os riscos são considerados à medida que cada evolução é realizada[...]”(PRESSMAN, 2011, p. 41).

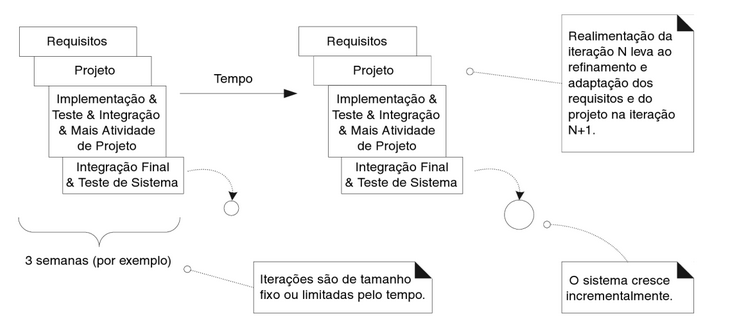
Como nenhum modelo é perfeito, o Modelo Espiral também tem suas limitações, como o fato de exigir avaliação de riscos frequentes, pois se algum risco importante não for detectado e controlado poderão ocorrer sérios problemas.

## 2.6 Processo Unificado

O Processo Unificado é um processo para desenvolvimento de software iterativo que busca tirar proveito dos melhores recursos dos modelos mais tradicionais citados anteriormente. “Processo Unificado surgiu comoum processo iterativo popular para o de o desenvolvimento de software visando à construção de sistemas orientados a objetos”. (LARMAN, 2007, p. 46).

O Processo Unificado segue uma abordagem de desenvolvimento iterativa, que se baseia em incrementos e melhorias de acordo com a evolução do desenvolvimento, as mudanças são vistas como inevitáveis e essenciais para sucesso do projeto, ou seja, a adaptação faz parte do modelo. Vejamos na figura 7 como funciona o fluxo do desenvolvimento iterativo. Iterativo

**Figura 7** - Desenvolvimento e Evolutivo



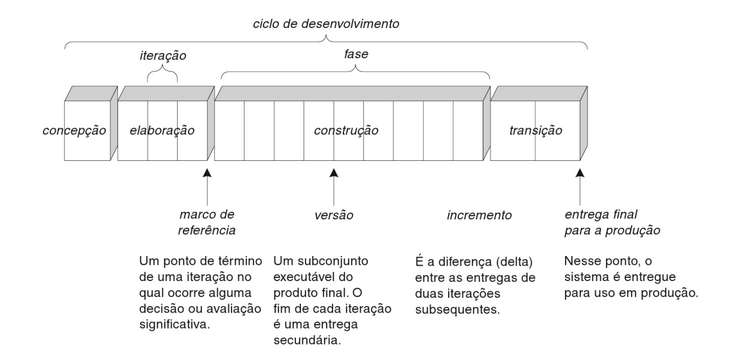
**Fonte:** LARMAN (2007, p. 48)

LARMAN descreve as quatro fases do Processo Unificado:

* **Concepção –** são realizadas estimativas vagas e é criado o escopo inicial;
* **Elaboração –** são realizadas estimativas mais detalhadas, é feita a implementação iterativa da arquitetura central, riscos de com classificação alta são resolvidos, requisitos em sua maioria são identificados;
* **Construção –** é realizada a implementação também iterativa do restante que possui menor risco e também é realizada a preparação para implantação;
* **Transiçao –** realização de testes, entrega da versão beta e implantação.

Vale ressaltar que o processo pode ser composto de várias iterações de acordo a evolução, vejamos na figura 8 mais detalhes.

**Figura 8** - Cronograma do Processo Unificado



**Fonte:** LARMAN (2007, p. 62)

Esse modelo é seguido por uma metodologia de desenvolvimento de software muito difundida pertencente à IBM, o RUP (Rational Unified Process), que será mais detalhado posteriormente.

# PARADIGMAS DE PROGRAMAÇÃO

Atualmente existem diversos paradigmas de programação, cada um com suas características e com sua melhor aplicabilidade. Ter conhecimento sobre os paradigmas de programação é essencial para que se possa saber qual a melhor maneira de desenvolver um projeto. Os principais paradigmas de programação são:

* **Programação Orientada a Objetos (POO):** A programação orientada a objetos, do inglês *Object Oriented Programming,* é um modelo de programação que procura representar as coisas do mundo real através de objetos, neste modelo tudo é um objeto, e esses objetos possuem características e comportamentos, conhecidos como atributos e métodos, respectivamente, estas características e comportamentos são organizados dentro de um modelo do objeto, que é chamado de classe. O principal objetivo da POO é aproximar o mundo real do virtual, através de representações e linguagens de fácil entendimento. Alguns exemplos de linguagens orientadas a objetos são: C#, C++, Objective C, Java, Objective Pascal, Python, entre outras;
* **Programação estruturada:** Tipo de programação que é voltada para estruturas de decisão como, “se....então” e “caso algo...então faça”, além de se utilizar de interadores, que determinam condições para que algo seja repetido. Estás estruturas de decisão e integradores são definidos dentro de funções que podem ser chamadas uma ou várias vezes dentro de um mesmo programa, estás funções ainda podem ser abstraídas para subprogramas. A programação estruturada ainda permite a definição de estruturas que são dados organizados para representar algo mais complexo. Exemplos de linguagens de programação estruturada são Pascal e C;
* **Programação modular:** Modelo em que os programas são divididos em vários módulos, estes, são conectados através de uma interface em comum;
* **Programação procedural:** Neste modelo os programas são executados através de que podem ser chamados sucessivas vezes. Exemplos de linguagem são o Fortran e o BASIC.

O problema a ser solucionado indica qual paradigma deve ser utilizado, e logo o paradigma irá influenciar na linguagem escolhida. Para resolver o problema proposto deve se utilizar o paradigma POO, para que caso seja necessário alterar ou expandir, torne-se mais simples de se realizar está tarefa. “Sistemas orientados a objetos são mais fáceis de mudar do que os sistemas desenvolvidos em abordagens funcionais”. (SOMMERVILLE,2011).

# RUP

O RATIONAL UNIFIED PROCESS (RUP) é um processo de engenharia de software criado para otimizar e apoiar o desenvolvimento orientado objeto, com uma sistemática para facilitar o uso da UML, podendo ter três definições.

* 1. Uma maneira de desenvolvimento de software iterativa e centrada na arquitetura e seguida por casus de uso.
  2. Provê uma estrutura bem definida para o ciclo de vida de um projeto, um processo bem definido e bem estruturado na parte de engenharia de software.
  3. É um produto que oferece para a engenharia de software uma estrutura de processo flexível.

Para a aplicação do RUP não a uma formula definida pois pode ser aplicado de diferentes formas para cada organização e projeto que ele é apresentado.

O RUP apresenta alguns elementos principais segundo KRUTCHEN.

* **Papéis –** os papéis são definidos de acordo com as responsabilidades de um indivíduo um grupo com características comuns;
* **Atividades –** é basicamente o trabalho que um indivíduo executa no exercício de seu papel produzindo resultados importantes na elaboração do projeto;
* **Artefatos –** são produtos produzidos durante um processo determinado do desenvolvimento do software, podem muito bem ser modificados;
* **Fluxo de trabalho –** sequência de atividades executadas com objetivo de produzir um resultado que tenha algum valor para o projeto.

Os fluxos de trabalho do RUP são os Fluxos de trabalho principais, associados com cada disciplina, o Fluxos de trabalho de detalhe, para detalhar cada fluxo de trabalho principal e os planos de iteração, que mostram como a iteração deverá ser executada.

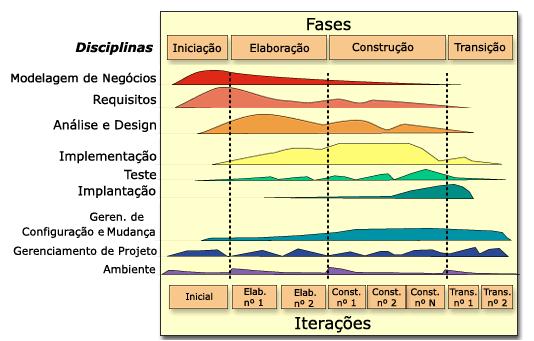
Os fluxos de trabalho podem ser representados por diagramas de sequência, diagramas de colaboração e diagramas de atividades da linguagem de modelagem unificada.

De acordo com KRUTCHEN o RUP possui nove disciplinas:

* Modelagem de negócios;
* Requisitos;
* Análise e projeto;
* Implementação;
* Teste.
* Configuração e gerenciamento de mudanças;
* Gerenciamento de projeto;
* Ambiente.

Como mostra a figura [Martins 2007] A arquitetura do RUP e constituída por dois eixos horizontal e vertical.

**Figura 9** - Eixos do Fluxo de Trabalho do RUP



Disponível em <http://www.sgp-ti.uerj.br/site/metodologia/>

O eixo Vertical representa as disciplinas que agrupam as atividades de maneira lógica naturalmente, e representa o modo estático do processo que é descrito em termos de componentes, disciplinas, atividades, fluxos de trabalho, artefatos e papéis do processo.

O eixo horizontal apresenta o tempo mostrando os aspectos do ciclo de vida do processo à medida que ele se desenvolve. Apresentado o aspecto dinâmico do processo em termos e fases.

O ciclo de vida do RUP constitui-se em 4 fases, são elas:

* Concepção: Define o escopo do projeto.
* Elaboração: Detalha os requisitos e a arquitetura.
* Construção: Desenvolve o sistema.
* Transição: Implanta o sistema.

CONSIDERAÇÕES FINAIS

Nosso objetivo neste trabalho e passar para os leitores a total importância de se criar um sistema de vendas para drogarias, e que a base de um bom projeto se encontra na engenharia de software, mostramos alguns dos modelos mais utilizados como modelo cascata, modelo incremental, modelo espiral entre outros modelos que ainda não foi falado neste trabalho.

O objetivo e montar um sistema para área de vendas em drogarias que poderá ser adaptado a qualquer senário dentro do mercado atual. Este trabalho foi muito importante para nós como equipe, pois mostrou o valor de se criar algo como um grupo, e juntos conseguimos aprender muito mais sobre os processos de se desenvolver um sistema usando metodologias da engenharia de software. Esperamos que os leitores deste trabalho consigam aprender muito mais com o conteúdo do mesmo e que este projeto sirva como uma base de conhecimento para todos os usuários de sistemas de informação e a população que se interesse pelo assunto.

REFERÊNCIAS

**Livros:**

SOMMERVILLE, Ian. **Engenharia de Software**. 9ª ed. São Paulo. Pearson. 2013;

PRESSMAN, Roger S. **Engenharia de Software.** 7ª ed. São Paulo. Bookman. 2011;

LARMAN, Craig. **Utilizando UML e Padrões.** 3ª ed. São Paulo. Bookman. 2007;

**Sites:**

**Engenharia de Software** <http://sbc.org.br/images/flippingbook/computacaobrasil/computa\_28/cb2015-03.pdf> Acesso em 17 de setembro de 2016;

**Programação Orientada a Objetos: Uma Introdução** <http://www.hardware.com.br/artigos/programacao-orientada-objetos/> Acesso em 20 de setembro de 2016;

**Os 4 pilares da programação orientada a objetos** <http://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264> 21 de setembro de 2016;

**Rational Unified Process – RUP** <http://www.baguete.com.br/artigos/04/11/2009/rational-unified-process-rup> Acesso em 26 de setembro de 2016;

**RUP como Metodologia de Desenvolvimento de Software para Obtenção da Qualidade de Software** <http://www.boente.eti.br/publica/seget2008rup.pdf> Acesso em 25 de setembro de 2016.