UFSC / CTC / INE
Disciplina: Gerência de Projetos

Curso: Pós-Graduação

Prof. Dr. João Dovicchi*

1 Metodologias: XP, RUP e ITIL

Abranger todas as metodologias de projeto, neste curso, seria impraticável. Assim, vamos nos ater a alguns exemplos, tentando abranger metodologias *light* e *heavy*. Para isso, escolhemos XP, RUP e ITIL (Prince2) para nossa abordagem.

1.1 XP

XP é uma metodologia de desenvolvimento de software que foi criada por Kent Beck, Ron Jeffries e Ward Cunningham [**?**], voltada para equipes pequenas e médias, com requisitos vagos ou que mudam freqüentemente [**?**, **?**]. A metodologia XP enfatiza pouco os processos formais de desenvolvimento e mais a disciplina de desenvolvimento. Nesta metodologia, o foco principal é na codificação do software, muito mais do que na documentação e use cases. XP tem chamado muita atenção devido à sua ênfase na comunicação, simplicidade e práticas de desenvolvimento sustentado.

Os principais aspectos da metodologia XP são:

- Test Drive Development (TDD)
- Participação do cliente
- Revisão permanente do código
- Refactoring

*http://www.inf.ufsc.br/ dovicchi --- dovicchi@inf.ufsc.br

- Integração e comunicação contínua
- Refinamento contínuo da arquitetura
- Planejamento

Para isso é necessário que seja criada um canal sempre aberto de comunicação entre as pessoas; acompanhar rigorosamente o feedback do cliente, priorizar funcionalidades de acordo com suas especificações; prezar o design simples e a constante refactoring; e apresentar um método de deliver incremental e interativo [**?**].

1.1.1 XP e desenvolvimento de software

Os projetos de desenvolvimento de software são os que mais apresentam falhas. Cerca de 80% dos projetos falham em cumprir cronogramas e contratos. Esta cifra é bastante alta e tem levado ao descrédito de inúmeras software houses e equipes de desenvolvimento de software. Mas, porque os projetos falham? Quais os principais problemas? Se formos observar, as falhas se devem a:

- Atrasos no cronograma
- Falta de planejamento adequado
- Falha no tratamento de bugs
- Incompreensão de requisitos de negócio e de sistema
- Mudanças constantes podendo estar relacionadas a requisitos, equipe, cronograma...

No passado, o desenvolvimento de software era baseado em processos tradicionais, compreendendo a análise e o projeto; e só depois codificava-se o software. Este processo demandava muito tempo antes da codificação, elevando o custo (vide figura 1).

O desenvolvimento era baseado em previsões e existia os temores de alterações de requisitos. Nos dias de hoje os processos modernos incentivam as iterações, de forma que alterações em etapas posteriores se tornam mais baratas. Neste aspecto a engenharia de software é diferente das outras engenharias, podendo lançar mão de componentes, frameworks e bancos de dados, absorvendo o alto custo da mudança (vide figura 2).

A metodologia XP valoriza, essencialmente, os seguintes aspectos:

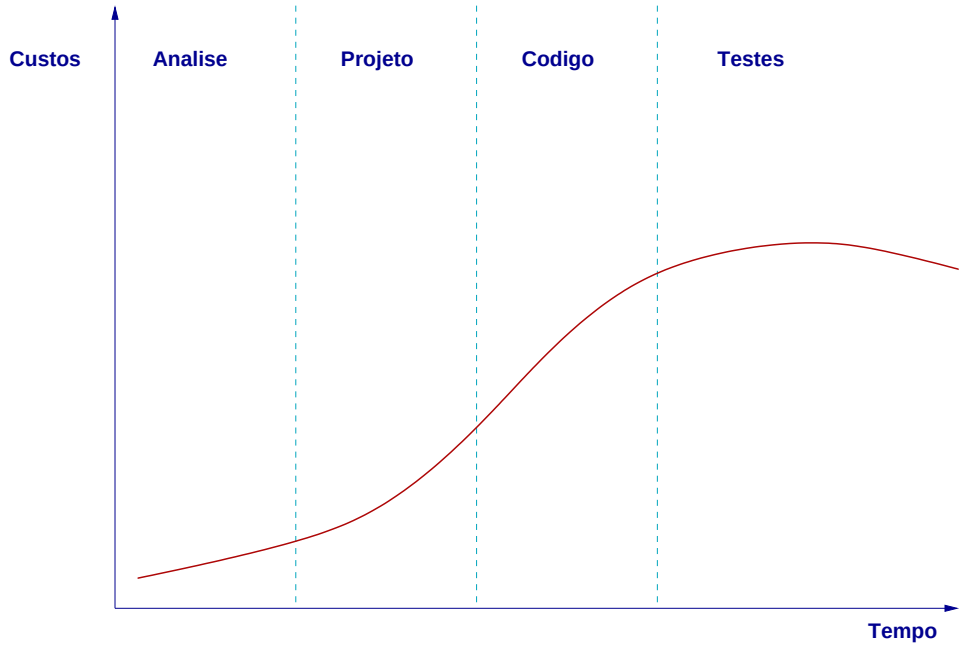

Figura 1: Projetos tradicionais: Análise de custos

Comunicação: Práticas valorizam a comunicação, não limitada por procedimentos formais.

Simplicidade: Incentiva ao extremo práticas que reduzam a complexidade.

Feedback: Práticas que garantem um rápido feedback de ambas as partes durante todo o projeto.

Coragem: Práticas aumentam a confiança do desenvolvedor.

Assim, o XP evidencia maneiras melhores de desenvolver software fazendo-o de forma cooperativa e valorizando mais os indivíduos e a interatividade do que processos e ferramentas; software funcionando do que documentação abrangente; envolvimento do cliente do que negociação e contratos; e, finalmente, respostas a mudanças do que um plano a ser seguido. Com isso, pode-se dizer que XP é uma metodologia voltada para a inovação de processos, centrada no indivíduo e na qualidade.

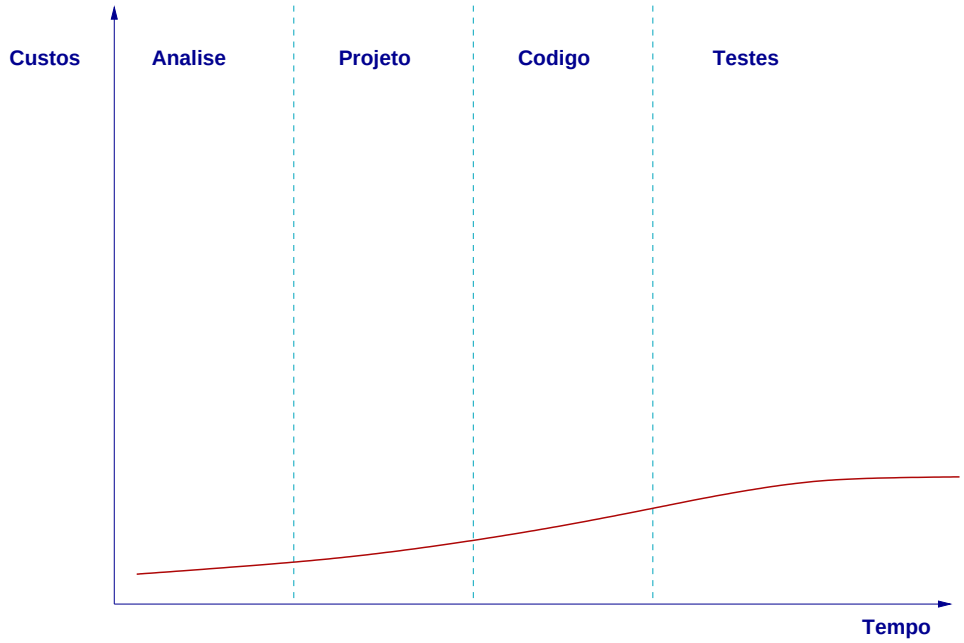

Figura 2: Projetos XP: Análise de custos

1.1.2 XP e qualidade

Os processos tradicionais priorizam as variáveis de tempo, escôpo e custo do projeto, manipulando a qualidade do produto para manter estas variáveis dentro de seus limites. Na metodologia XP, as variáveis priorizadas são tempo, qualidade e custo, manipulando o escôpo para mantê-las dentro do previsto.

O cumprimento do protocolo de implementação da metodologia XP segue um ciclo entre cliente e desenvolvedor, com base nas especificações e no escopo do projeto. O cliente define o escôpo, a equipe avalia a relação tempo / funcionalidade / custo. Então o cliente escolhe as prioridades e a equipe implementa a funcionalidade (ver figura 3). Deste modo, o escôpo é definido, mas o cliente prioriza as funcionalidades mais necessárias, sem perda de qualidade e aumento de custos.

O ciclo da figura 3 envolve:

Atividade: Revisão de código

XP: Programação em pares

Atividade: Testes freqüentes

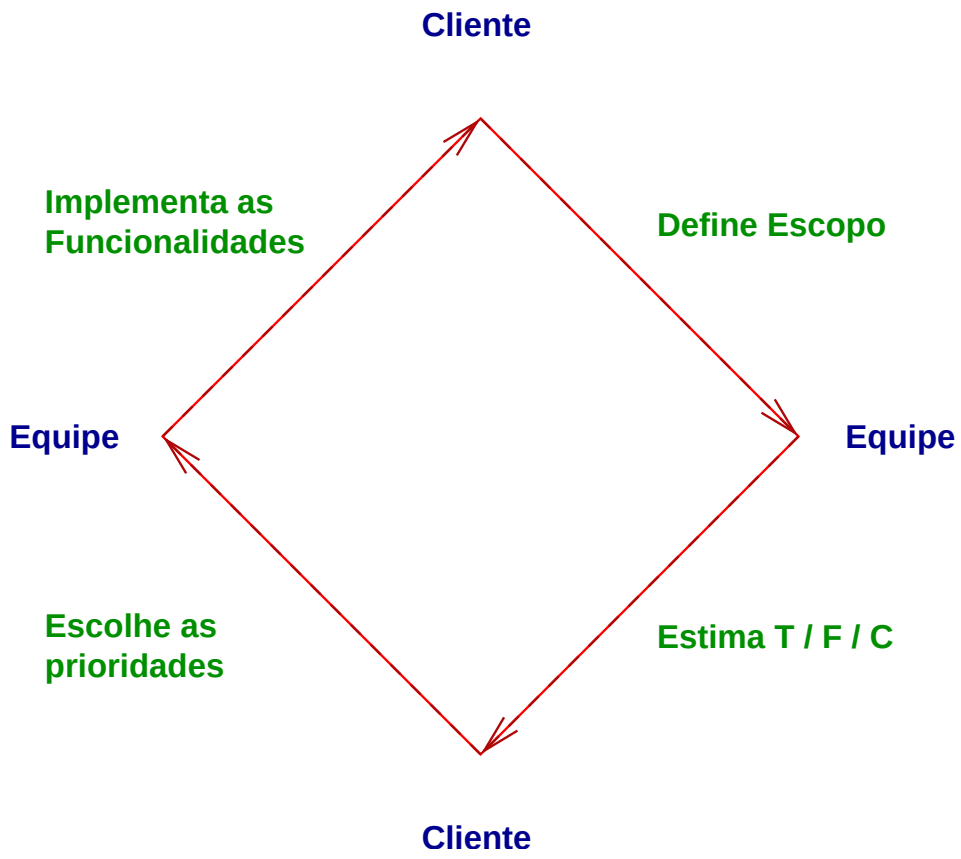

Figura 3: Ciclo de desenvolvimento de projetos em XP.

XP: TDD, automatizados

Atividade: Simplicidade

XP: Recursos não prioritários são descartados

Atividade: Projeto

XP: É realizado a qualquer hora

Atividade: Estimativas

XP: São revistas a cada iteração, apoiadas pelas metáforas

Atividade: Versões

XP: Pequenos lançamentos

Todo o processo envolve um outro ciclo de práticas nos níveis organizacionais, de equipe e dos pares de programadores.

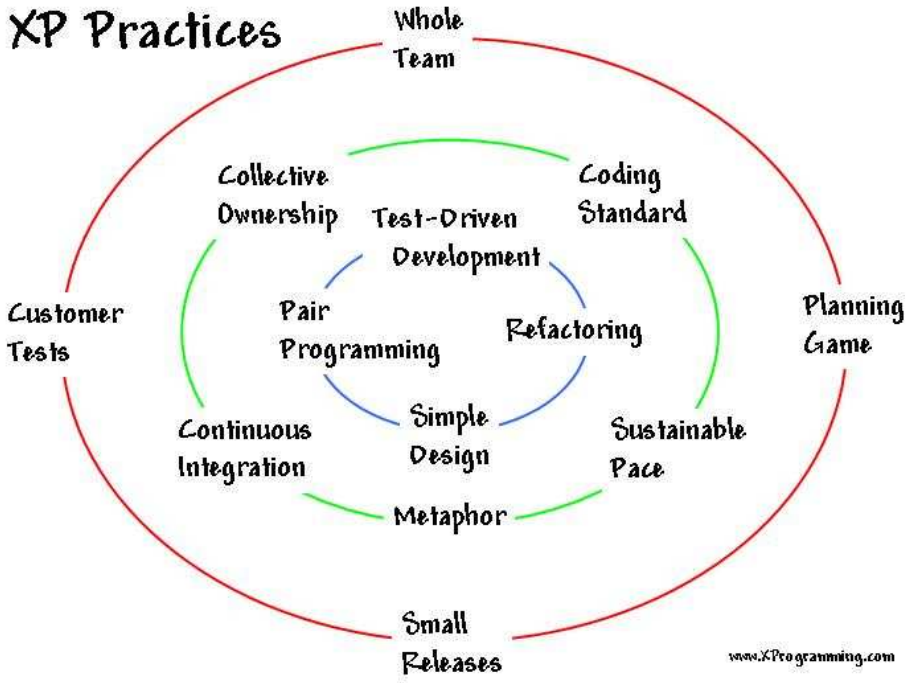

Figura 4: Níveis dos ciclos de projetos em XP. (Fonte: www.xprogramming.org)

Os ciclos da figura 4 representam os níveis de relacionamento do XP. O primeiro nível (mais externo) representa as práticas organizacionais; o segundo representa as práticas de equipe; e, finalmente, o terceiro (mais interno) representa as práticas de pares. Desta forma, a metodologia se torna centrada na qualidade de software, com agilidade e custo controlado.

1.2 Unified Process - RUP

Uma das mais utilizadas metodologias de projeto é a chamada de Unified Process. O *Rational Unified Process*, ou RUP, é uma metodologia de projeto de desenvolvimento de software estabelecida pela Rational Software Corporation e

descreve como desenvolver software de forma efetiva, usando técnicas testadas e aprovadas comercialmente.

Como metodologia, o RUP é classificado como um processo heavy (pesado) e, por isso, aplica-se a equipes grandes de desenvolvimento para grandes projetos baseados em modelos orientados ao objeto e a componentes. Um dos exemplos é o de aplicação do RUP é o desenvolvimento de sistemas J2EE (Plataforma Java 2, Enterprise Edition).

Esta metodologia está embasada em seis práticas de desenvolvimento de software:

- Desenvolvimento Iterativo
- Gerência de Requisitos
- Modelagem visual
- Arquitetura baseada em componentes
- Controle de qualidade
- Controle de versões

O processo analítico do RUP divide o ciclo de vida de desenvolvimento nas fases:

Descrição $\rightarrow$ Elaboração $\rightarrow$ Construção $\rightarrow$ Transição

Este processo é seqüencial e executado de forma cíclica, ou seja, para cada ciclo do processo existe um subproduto utilizável, mas que pode não atingir todos os requisitos do sistema. O RUP é como construir uma casa, e então adicionar a lavanderia, depois adicionar a garagem, até que no final se tenha a casa que foi planejada.

A iteração, no processo do RUP é uma abordagem nova, onde cada iteração resulta em melhorias de um produto que vai se tornando cada vez mais próximo dos requisitos do sistema completo. Esta iteração, no entanto, deve estar vinculada às práticas de desenvolvimento, de tal forma que o processo e as práticas regulem um ao outro (ver figura 5).

A primeria fase (Descrição) se refere à articulação da noção geral do sistema e o estabelecimento de um projeto formal para construí-lo. Na prática poderíamos comparar esta fase a um esboço detalhado que possa dar uma visão clara do projeto para se estabelecer como deverá ser sua elaboração.

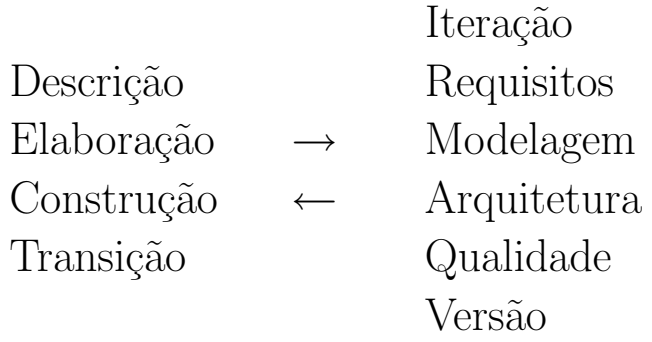

Figura 5: Regulagem entre práticas e processos

A fase seguinte (Elaboração) já envolve o detalhamento do projeto, detalhando os requisitos e especificações. Tantos os requisitos funcionais como os não-funcionais para o sistema devem ser definidos neste ponto. Os requisitos não-funcionais podem ser encarados como fatores críticos para o sucesso, que descrevem o grau de risco envolvido no desenvolvimento do sistema. Nesta fase a arquitetura básica do sistema e o cronograma para a sua produção a ser especificados na documentação.

Na terceira fase (Construção) são construídos os detalhes da arquitetura básica que resulta na arquitetura final. Qualquer alteração da arquitetura básica deve ser mínima nesta fase. É nesta fase que se estabelece as iterações do processo.

Na última fase (Transição) o sistema é apresentado ao cliente. O usuário deve testar o sistema, pode requisitar o treinamento e encontrar falhas do sistema que devem ser eliminadas. Podem ser necessárias várias iterações até que o cliente aceite, formalmente o sistema pronto.

A abordagem ciclica e iterativa da metodologia RUP apresenta uma vantagem em relação às metodologias seqüenciais ou com uma abordagem "em cascata". Cada iteração do RUP corresponde a uma geração, ou seja, uma versão do produto e sua respectiva documentação. Cada versão deve ser um subproduto utilizável, embora possa não ser a idéia completa do produto.

O gerenciamento dos requisitos é, para a metodologia RUP, um problema de gerenciamento do projeto, como se dá em qualquer metodologia. Requisitos adicionais podem ter um forte impacto nos custos do projeto e devem ser considerados em todos os seus detalhes.

A modelagem visual torna clara a comunicação inter-processos para a equipe técnica, responsável pelo desenvolvimento dos componentes, e mos-

tra como estes componentes devem interagir entre si. Para isso, a linguagem UML pode ser de grande auxílio no processo de modelagem.

De maneira análoga ao processo de alteração de requisitos, a alteração do software (ou produto) pode emperrar o processo de desenvolvimento do sistema. Assim, deve-se ter um rigoroso sistema de controle de versão (CVS, por exemplo), sendo fundamental a compreensão do impacto das alterações no sistema. A gerência do projeto deve sempre estar ciente das alterações.

1.2.1 RUP e agilidade

Uma questão relevante sobre a metodologia RUP é se ela pode ser usada como uma metodologia ágil. Martin [**?**] descreve uma versão ágil do RUP denominada de dX (leia a sigla de ponta cabeça para compreender a brinceira) que propõe uma metodologia RUP para quem deseja usar XP:

Quando RUP foi desenvolvida por Kruchten [**?**], ela foi feita como um processo complementar para UML. No entanto, como um modelo (*framework*) de processo ela pode ser aplicada a diversos processos [**?**]. A maior crítica ao RUP está no fato de que ele pode ser "qualquer coisa que acaba sendo nada [1]" [**?**].

Esta crítica se deve ao fato de que o RUP preve um processo cíclico, sem um fim previsível. O que nem sempre é verdade ou, melhor ainda, o que pode acontecer, também, para qualquer metodologia não cíclica.

Embora o RUP seja uma metodologia pesada, novas abordagens têm sido feitas no sentido de usá-la como uma metodologia ágil. Uma delas é a abordagem `dX`. Assim, da mesma forma, o RUP pode ser usado como uma metologia pesada ou ser adaptado para uma forma de metodologia ágil.

1.3 IT Infrastructure Library

ITIL [2] (*IT Infrastructure Library*) é uma recomendação documentada e uma filosofia para a prática adequada de serviços na área de tecnologia da informação. Apesar de "pregar" uma metodologia de projeto, podemos considerá-la mais como uma filosofia de trabalho que promete prover resultados de qualidade nos serviços de TI e facilitar o suporte nesta área [**?**].

Consiste, basicamente, de uma série de documentos que pretende orientar os projetos de TI com respaldo de várias organizações para satisfazer a solução dos problemas advindos do aumento de requisitos tecnológicos nas organizações

[1] Traduzindo Martin Fowler: "... it can be anything it ends up being nothing."

[2] `http://www.itil.co.uk/`

modernas. Neste aspecto, a metodologia pretende ser não proprietária, permitindo a troca de informações e conhecimento entre as empresas que a adotam, aumentando a possibilidade de suporte aos produtos gerados pela sociedade da informação. Entretanto, todo o material não é gratuito e a certificação é paga.

Basicamente, a ITIL tem como base:

1. Promover a qualidade dos serviços de tecnologia e organização de TI;
2. Implementar uma metodologia de gerência de TI na organização;
3. Estabelecer um ambiente estável com base no aumento da pprodutividade;
4. Aumentar a eficiência do pessoal de apoio de TI e melhorar os serviços ao cliente; e
5. Identificar pontos críticos para melhoria de qualidade nos serviços de TI.

1.3.1 ITIL e Gerência de Projetos

No que diz respeito à gerência de projetos de software, a metodologia ITIL está mais voltada para a gerência do processo em si, com a finalidade de garantir a aplicação dee processos e métodos padronizados para a identificação, controle e desenvolvimento. A função da gerência é responsável pelo cumprimento do cronograma, alocação de recursos e fiscalização do cumprimento dos requisitos e de suas especificações.

ITIL considera a gerência de projetos como ações em etapas, associadas a atributos que permitam facilitar o ciclo de vida de um projeto como um todo, incluindo: planejamento, análise, design, implementação, teste, avaliação de qualidade. Do ponto de vista do projeto, a ITIL usa a metodologia PRINCE2 [3] (Project IN Controled Environment). Na verdade podemos considerar que ITIL está para metodologia assim como PMI, PMBOK, PRINCE2 ou XP estão para gerência de projetos e ISO está para processos e procedimentos.

2 Engenharia, Qualidade e Referências

Podemos, agora, explorar com mais detalhes a gerência de projetos de software a partir de um conjunto de referências que nos ajudarão a compreender os dife-

[3] `http://www.ogc.gov.uk/prince2/`, ver, também `http://www.prince2.com/whatisp2s.html` para uma explicação em espanhol.

rentes aspectos da engenharia de software e a sua gerência. Isto é fundamental para as atividades que envolvam a utilização e desenvolvimento de sistemas computacionais ou que utilizam a tecnologia da informação.

Evidentemente, a quantidade de recursos e materiais sobre o assunto é tão vasta que um curso desta natureza tem que optar por uma abordagem informativa, no lugar de uma abordagem formativa. Assim, na prática, vamos apresentar as referências necessárias para que cada um possa explorar estes aspectos por si mesmo, usando a informação disponível para a resolução de problemas na área de engenharia de software e gerência de projetos de software.

Planejar um projeto de software tem algumas características particulares que devem considerar todas as estimativas, análise de risco, cronograma, qualidade e controle. Deve-se, na prática, estimar o custo, esforço, recursos e tempo para o desenvolvimento de um sistema ou produto.

Para isso, existem numerosas ferramentas para cada tipo de atividade prevista de um projeto. A começar com ferramentas de estimativa até ferramentas de modelagem. Tratar de todas elas demanda um grande tempo e esforço de pesquisa. No entanto, estaremos apresentando de forma resumida algumas referências:

2.1 Estimativa

Existem numerosas ferramentas de estimativa para projetos. Apresentamos abaixo, algumas referências onde pode ser encontradas informações e ferramentas para estimativa de projetos de software.

- SCEW - Software Cost Estimation Website
  `http://www.ecfc.u-net.com/cost/index.htm`
  Este site contém muitas informações úteis para estimativa de custo de projeto de software e apresenta algumas ferramentas online.

- NASA Cost Estimation
  `http://www1.jsc.nasa.gov/bu2/`
  Excelente site com muitos recursos e ferramentas para estimativa de custo de software, incluindo o NASA Cost Estimation Handbook.

- DACS - Data and Analysis Center for Software
  `http://www.dacs.dtic.mil/databases/url/key.hts?keycode=4`
  Técnicas, métodos e ferramentas disponibilizadas pelo Departamento de Defesa Americano para projetos de software.

- Construx Estimate `http://www.construx.com/resources/`
  Ferramenta livre e gratuita para estimativa de projetos de software.

2.2 Cronograma

Uma vez selecionado o modelo apropriado, identificadas as etapas, estimados a quantidade de trabalho, pessoal, riscos, custos etc., é necessário estabelecer o cronograma de trabalho para que o trabalho seja feito no tempo previsto. Alguns recursos e ferramentas podem facilitar o trabalho de organização e cronograma, como por exemplo: Earned Value Analysis (EVA) Resources [4] que é Uma excelente fonte de informações (FAQ, papers e links) para ajudar na organização e cronograma de projetos.

2.3 Qualidade de software

Qualidade de software, ou SQA (*Software Quality Assurance*), é uma das grandes preocupações da gerência de projetos. Antes, porém, é necessário que se defina o que vem a ser "Qualidade de Software", que se crie um conjunto de atividades e procedimentos que garantam esta qualidade, aplicar estes procedimentos, entender (e utilizar corretamente) as métricas para identificar e implementar a qualidade de software em um projeto.

A gerência de qualidade e *Software Quality Assurance* (SQA), é baseada em padrões definidos e estabelecidos por várias organizações:

- ISO 9000-3 e ISO 9001
  `http://www.tantara.ab.ca/iso_list.htm`
  Documentações e recursos sobre a padronização ISO.

- ASQ - American Society for Quality
  `http://www.asq.org/`
  Controle de qualidade de software e outras informações relevantes sobre qualidade, oferecido pela Sociedade Americana para Qualidade.

- NASA - SATC: Software Assurance Technology Center
  `http://satc.gsfc.nasa.gov/fi/fipage.html`
  Documento da NASA para auditoria de software ou *Software Formal Inspection* (SFI) que compreendem vários Technical Reviews e Technical Reports, principalmente o *Formal Inspection Guidebook*.

[4] `http://www.nnh.com/`

- The Quality Tools Cookbook
  `http://www.sytsma.com/tqmtools/tqmtoolmenu.html`
  Fonte de informações sobre ferramentas de qualidade. Uma espécie de "livro de receitas".

- Quality Cost Analysis: Benefits and Risks
  `http://www.badsoftware.com/qualcost.htm`
  Artigo que discute o custo benefício da qualidade [5].

[5]Publicado por Cem Kaner, no Software QA, Volume 3, #1, 1996, p. 23