



Aplicando as Sete Métricas de Controle de Projeto

Rational Software White Paper

**Moacyr Cardoso de Mello Filho
revisão 1.1**

Sumário

Resumo
Abstract

1.	Introdução	1
1.1	Propósito	1
1.2	Escopo	1
1.3	Acrônimos e Abreviaturas	1
1.4	Referências	2
1.5	Visão Geral	2
2.	Métricas no Desenvolvimento de Software	3
2.1	O Que é Métrica?	4
2.2	Métricas de Controle versus Métricas de Estimativa	4
3.	Métricas no Escopo do RUP	6
4.	As Métricas de Projetos Iterativos são Diferentes das Métricas Waterfall	7
5.	O Que um Projeto Precisa Medir?	8
6.	Descrição das Sete Métricas	9
6.1	Progresso – Trabalho realizado ao longo do tempo	9
6.1.1	Cenário de aplicação	11
6.1.2	Interpretação	12
6.2	Despesa - Gastos ao longo do tempo	12
6.2.1	Cenário de aplicação	15
6.2.2	Interpretação	16
6.3	Rotatividade - Mudança de pessoal ao longo do tempo	18
6.3.1	Cenário de aplicação	18
6.3.2	Interpretação	19
6.4	Estabilidade - Fluxo de mudanças ao longo do tempo	20
6.4.1	Cenário de aplicação	21
6.4.2	Interpretação	21
6.5	Modularidade - Fragmentação média por mudança ao longo do tempo	23
6.5.1	Cenário de Aplicação	24
6.5.2	Interpretação	24
6.6	Adaptabilidade - Retrabalho médio por mudança ao longo do tempo	25
6.6.1	Cenário de Aplicação	25
6.6.2	Interpretação	26
6.7	Maturidade - Taxa de defeitos ao longo do tempo	27
6.7.1	Cenário de Aplicação	27
6.7.2	Interpretação	28
7.	Conclusão	29



Rational Software White Paper

Resumo

Em muitos projetos comerciais a medição de componentes do projeto fornece os elementos para acompanhamento e supervisão. Este documento explora algumas métricas disponíveis para organizações que adotem o Processo Unificado Rational como parte de sua estratégia de desenvolvimento de software.

Abstract

In many commercial projects the measure of projects components provide the elements for tracking and oversight. This paper explores some metrics available to organizations adopting the Rational Unified Process as part of their software development strategy.

1. Introdução

A medição de um projeto de software é vital para o seu controle, neste documento abordaremos métricas de controle em um contexto de projeto bem definido, particularmente em um contexto onde se aplica o Rational Unified Process. Como este é um texto para um público técnico, não hesitamos em remeter nossos leitores para o glossário do Rational Unified Process sempre que pelas circunstâncias, ou por nossa própria inabilidade, usarmos termos que estão muito melhor explicados naquele documento.

1.1 Propósito

Descrever métricas de supervisão, controle e monitoramento de projetos de software.

1.2 Escopo

Nossa abordagem de métricas se restringirá ao âmbito de um processo iterativo bem definido.

1.3 Acrônimos e Abreviaturas

CMM	Capacity Maturity Model
COCOMO	Construtive Cost Model
CR	Change Request
Hh	Homens-hora
Hm	Homens-mês
Hw	Homens-week
LOC	Lines of Code
LR	Labor Rate
MTBF	Mean Time Between Failures
OO	Objected-Oriented
PMBOK	Project Management Body of Knowledge
PMI	Project Management Institute
RUP	Rational Unified Process
SCO	Software Change Order
SEI	Software Engineering Institute
SLOC	Source Lines of Code
UC	Use Case
UML	Unified Modeling Language
UT	Usage Time



1.4 Referências

- Software Project Management – A Unified Approach: Walker Royce, Addison Wesley Longman, 1998
- Object Solutions: Managing the Object-Oriented Project: Grady Booch, Addison Wesley Publishing Company, 1995
- Rational Unified Process 2002: Rational Software Corporation, 2002
- Object-Oriented Project Management Training Course – OOPM: Rational University, Rational Software Corporation
- Principles of Managing Iterative Development Training Course – PMID: Rational University, Rational Software Corporation
- PMBOK – Project Management Body of Knowledge: Project Management Institute, 1996
- Software Measurement for DoD Systems: Recommendations for Initial Core Measures: Anita D. Carleton, et al. Technical Report 1992 – SEI, Carnegie Mellon University, Pittsburgh, PA
- Don't Be Afraid of the Elephant – A Small Set of Metrics and How to Interpret Them – Rational Whitepaper, Rational Software Corporation, 2002

1.5 Visão Geral

Este documento inicia com uma discussão geral sobre o assunto métricas, define o conceito, analisa os recursos de métricas que o processo da Rational (RUP) possui para tratar o problema e, em seguida, aborda a utilização de métricas em projetos iterativos, comparando a utilização das métricas em processos waterfall com processos iterativos, onde predomina aplicações no paradigma orientado a objetos. Finalmente descreve cada uma das sete métricas recomendadas por Walker Royce (ver primeira referência) contextualizando sua aplicação na gerência de projetos.



2. Métricas no Desenvolvimento de Software

Poucos temas provocam tanta discussão na comunidade de desenvolvimento de software como o assunto métricas. Existe uma percepção geral de que métricas são necessárias mas, ao mesmo tempo, são difíceis de se coletar e a recompensa não é imediata. Esta é uma sensação sempre presente: a de que o valor agregado pela coleta e análise das métricas ocorre muito mais tarde, só para os próximos projetos. Creio que esta visão está ligada a uma experiência real da comunidade com as métricas da programação estruturada e com o ciclo de vida *waterfall*.

Apesar de acreditar que medir é sempre melhor do que estimar, a prática adota francamente o *feeling* e a estimativa - e aqui eu cometo um eufemismo – pois, na verdade, as condições de um projeto planejado de modo tradicional não contribuem para o uso imediato das métricas, sendo muito comum uma interpretação dos números somente ao final do projeto. Por consequência as métricas de um projeto acabam por contribuir para a melhoria do próximo projeto e muito pouco para o atual, tornando a gerência pouco receptiva a implementar a prática de medição.

Há também a questão da qualidade do histórico acumulado. Frequentemente o planejador de um novo projeto percebe que os dados históricos são inconsistentes e, por falta de um padrão, não são adequados para servirem de estimativa para o problema presente. Conceitos diferentes, computações diferentes, coletas em momentos diferentes, tudo isto contribui para uma falta de confiança no histórico acumulado e, por consequência, na falta de confiança na precisão das estimativas. Estas são as causas profundas de não termos confiança em usar modelos de custo de mercado, tais como pontos de função ou o COCOMO (ver acrônimos e abreviaturas), pois todos eles exigem um certo grau de ajuste interno, contextualizado para a realidade da empresa, que dificilmente possuímos.

Este ciclo vicioso - pouca utilidade para o projeto presente, falta de histórico para os projetos futuros - já não ocorre quando aplicamos o processo iterativo-incremental, diferentemente do processo *waterfall*, o iterativo-incremental utiliza muito rapidamente toda e qualquer informação de *feedback* que as métricas possam oferecer.

Caso além do processo iterativo, estivermos falando de um desenvolvimento orientado a objetos, então as métricas ganham uma nova e importante dimensão, pois neste paradigma encontramos uma integração muito maior entre os modelos que constroem o *software*, desde o levantamento de requisitos até o momento de teste e liberação. Uma das maiores razões do porque a indústria automobilística, em geral, cria produtos de mais alta qualidade do que a indústria de *software*, é que a medição de seus artefatos é mais fortemente integrada ao processo de projeto e construção de carros. Os *loops* de *feedback* existem em todas as áreas do processo de manufatura, e vão desde decisões de *design*, medições de chão de fábrica até relatórios de experimentação de campo.

Neste documento apresentaremos sete métricas que tem por objetivo o controle e o acompanhamento do projeto. Estas métricas são usadas, por exemplo, para implementar a área chave de processo que o modelo CMM denomina *tracking and oversight*, (SEI-CMM, ver acrônimos e abreviaturas) são, portanto, métricas para supervisão e não para estimativa de planejamento de um projeto. Não obstante, são fundamentais para construir um verdadeiro histórico de informações de projeto.

2.1 O Que é Métrica?

Podemos definir métrica como um atributo mensurável de uma entidade. Caso a entidade seja um projeto, uma métrica seria, por exemplo, o seu tamanho. Tamanho é um atributo de projeto e pode ser medido. (Como medi-lo é outra questão).

Métrica *Atributo mensurável de uma entidade. Define-se métrica primitiva como sendo aquele atributo mensurável que pode, por leitura direta, ser obtido da própria entidade. Define-se métrica derivada como aquele atributo mensurável que é obtido por cálculo a partir de métricas primitivas.*

Podemos perguntar qual o tamanho de um determinado carro e seremos imediatamente compreendidos pela maioria das pessoas. Alguns porém, antes de responder, perguntarão se entendemos por tamanho o comprimento do carro, ou seu volume ou mesmo sua potência. Nestes casos as respostas seriam dadas, por exemplo, em metros, metros cúbicos ou HPs. Isto mostra a necessidade de interpretação e indica os vários ângulos de abordagem de um problema de medição.

Outro exemplo: seja um determinado requisito de *software*, que foi definido durante o processo de levantamento de requisitos, mas continua, nas fases seguintes, sofrendo alterações, conforme os usuários mudam sua concepção ou visão do sistema. O atributo **estabilidade** pode ser uma métrica para o requisito e podemos quantificar esse atributo pelo número de vezes em que o requisito foi alterado. É fácil obter essa métrica diretamente da contagem do número de ordens de mudança (solicitações de mudança de *software*) que se referem a esse requisito.

Usualmente o que coletamos são métricas que podem ser medidas diretamente, mas que são de pouco significado numa interpretação isolada. Por exemplo, o número de classes de um modelo de design é encontrado por simples inspeção, mas tem pouco significado por si mesmo. Porém, podemos estimar o tamanho do sistema se combinarmos o número de classes com o número de operações e atributos de cada classe, além de medidas sobre a profundidade da árvore de herança. Estas são as chamadas métricas primitivas, são dados brutos que combinados com outros dados brutos podem levar a métricas significativas.

2.2 Métricas de Controle versus Métricas de Estimativa

As sete métricas que apresentaremos são, como já dissemos, métricas para supervisão e não para estimativa de planejamento de um projeto. Portanto o seu uso típico é informar periodicamente ao gerente de projeto como está o andamento do trabalho. Para isto é conveniente que a coleta, computação e apresentação das informações estejam automatizadas. É fundamental a facilidade de obtenção dos valores, pois o período de atenção é contado em dias ou semanas.

Métricas de estimativa, por outro lado, não precisam estar disponíveis com facilidade, quer dizer, é perfeitamente possível gastar algum tempo pesquisando-se numa base de informações de métricas para obter estimativas para um novo projeto. Isto pode ser feito, por exemplo, através de *drill down* num *data warehouse* de métricas.

No caso de supervisão o gerente de projeto vai tomar decisões baseado em alarmes acionados ou na tendência dos indicadores. Já no caso do planejamento o gerente de projeto vai pesquisar semelhanças entre o projeto atual e os exemplos do passado, e então usar os valores para alimentar um ou mais modelos de custo que farão a computação das estimativas de planejamento, geralmente esforço (em homens-hora) e tempo.

Aplicando as Sete Métricas de Controle de Projetos

Em ambos os casos precisamos de métricas de volume (tamanho ou *size*), mas o uso é bem diferente. Para ilustrar tomemos o exemplo de planejamento com o modelo COCOMO. Um planejador deverá fornecer alguns dados para obter do modelo as estimativas de esforço e tempo, vejamos as fórmulas simplificadas:

$$\text{Effort} = C * \text{EAF} * (\text{size})^P$$

onde: Effort: é o número de homens-mês (esforço na unidade homens-mês)

C: é um coeficiente constante de escala

EAF: é um fator de ajuste de esforço que caracteriza o domínio (área médica, indústria, setor financeiro, etc), o pessoal, o ambiente e as ferramentas usadas para produzir os artefatos

P: é um expoente que caracteriza a economia de escala, inerente ao processo usado para o desenvolvimento. Mede, em particular, a capacidade do processo de evitar atividades que não acrescentem valor ao software (burocracia, retrabalho, reuniões, etc)

Size: é o tamanho ou volume do produto final, medido em pontos de função ou linhas de código conforme o estágio de pré ou pós-arquitetura

O tempo é função do esforço e também é afetado por um fator de escala.

(Nota: esta fórmula é um exemplo didático, utiliza COCOMO 81 e não entra em considerações sobre os modelos: Prototyping Model, Early Design Model e Post-Architecture Model definidos no COCOMO II)

Fica claro que para a aplicação do modelo, além dos fatores de contextualização ao ambiente da empresa, é preciso algum tipo de medida de volume, que no exemplo ficou sendo pontos de função ou linhas de código.

Quando analisamos o caso de supervisão a medida de volume também aparece, porém associada ao tempo e com outra finalidade. Isto é, o interessante é sabermos como o projeto está se comportando no tempo, com o objetivo de examinar se há convergência para o término do produto. Iremos terminar no prazo? Quanto de trabalho realizamos em relação ao que deveria ter sido realizado? A idéia por trás disto é identificar o **progresso**. Então *size* poderá ser medido em qualquer coisa que possa representar **progresso**, além das tradicionais linhas de código e pontos de função. Podemos usar:

- número de classes;
- número de cenários;
- número de Casos de Uso;
- pontos de Casos de Uso;
- pontos de função;
- número de Casos de Teste;
- linhas de código;
- quantidade de retrabalho;
- número de solicitações de mudança fechadas;
- número de critérios de avaliação atingidos.

Estas variações serão exploradas mais adiante quando entrarmos no assunto progresso. Agora, nos interessa ressaltar o fato de que as métricas coletadas para supervisão e controle não precisam necessariamente ser universais, apenas precisam servir como bons indicadores da informação que desejamos obter, no exemplo acima: progresso do projeto em direção ao término do produto.



3. Métricas no Escopo do RUP

A visão do RUP sobre métricas parte de uma questão fundamental: Porque medimos? A primeira e mais importante resposta é que medimos para ganhar controle sobre o projeto e portanto para sermos capazes de gerenciá-lo. Secundariamente medimos para melhor estimar novos projetos. Finalmente, medimos para saber como estamos evoluindo em determinadas áreas de nosso projeto ou Organização.

Como fazer medidas custa caro, não medimos tudo que encontramos só porque podemos medir. É preciso ser seletivo. Investigar na Organização quais são, precisamente, seus objetivos e coletar métricas que satisfaçam apenas a esses objetivos. Devemos avaliar necessidades sob os pontos de vista da Organização, do projeto e da parte técnica. Então construir um plano para aplicações das métricas onde estas estejam bem definidas e documentadas (**Measurement Plan**).

É muito fácil interpretar mal os dados coletados. Daí a insistência do RUP no planejamento e definição exata dos conceitos, cálculos e procedimentos em torno das métricas. Considere, por exemplo, um programador que, examinando um código, resolve otimizar determinado trecho muito mal escrito. O trecho era longo mas, depois da otimização, foi encurtado para um terço do tamanho original. Caso produtividade seja medida como geração de linhas de código, este trabalho aparecerá como produtividade negativa!

Para resolver estas e outras interpretações errôneas o caminho é utilizar as métricas, na prática, sempre a partir do plano pré-definido, realizando experimentos de coleta e interpretação. Nestes experimentos a interpretação é compartilhada com toda a equipe e, como num ensaio, decisões gerenciais são indicadas para mostrar a conseqüência da interpretação das métricas. Deve-se aceitar o fato de que as primeiras coletas poderão até ser descartadas, caso não se confirmem adequadas aos objetivos.

Portanto há duas atividades básicas no RUP com respeito a métricas: *Definir um Plano de Métricas*, que pode ser feito uma vez por projeto, e *Coletar Métricas*, registrando o resultado no documento de avaliação de status do projeto. Para o registro e manipulação o RUP propõe um repositório de métricas, sob responsabilidade do gerente de projeto, contendo as métricas primitivas e derivadas, assim como os procedimentos de obtenção das derivações.

O processo da Rational possui conceitos para a medição de qualidade tanto do produto quanto do processo, e possui diretrizes (*guidelines*) para diversos níveis de implantação de métricas. Desde um pequeno conjunto inicial (Small Set of Metrics), até um conjunto completo e abrangente. Fica para a personalização específica por empresa, definir o grau em que um projeto deseja começar a aplicar as medições.

Está embutida nesta abordagem a visão iterativa e incremental que facilita o ensaio inicial e prepara o caminho para a coleta correta de métricas históricas, as quais permitirão estimativas adequadas ao contexto da Organização.

Neste texto adotamos a perspectiva do Small Set of Metrics, na formulação feita por Walker Royce, “The Seven Core Metrics” (ver primeira referência, pág. 188 Cap. 13).



4. As Métricas de Projetos Iterativos são Diferentes das Métricas Waterfall

Quando usamos um processo iterativo, coletar as mesmas informações e interpretar da velha maneira que sempre fizemos, quando usávamos um processo *waterfall*, nos causará problemas. Por exemplo: com uma abordagem *waterfall* uma métrica usada para medir progresso pode ser a completude do artefato de requisitos: “o documento de requisitos deve estar completo em 25% do tempo de projeto”. Isto porém, não se aplica a uma abordagem iterativa pois, nesta última, o documento de requisitos não fica completo até chegarmos na última iteração. Um uso sem cuidado dessa métrica *waterfall* leva a interpretar que o projeto iterativo está atrasado gerando preocupações desnecessárias.

As métricas devem ser diferentes porque o processo e o produto são diferentes: lidamos com objetos e não com funções, criamos *releases* executáveis e não documentação. Porém a principal questão talvez seja que o processo iterativo necessite de uma mudança no foco das métricas. Precisamos desenfaturar a importância do valor absoluto e enfatizar a variação no tempo. Nos beneficiamos mais com uma métrica que permita a obtenção de um valor inicial, como uma semente, somando-se uma taxa de mudança desse valor. Isto quer dizer enfatizar a tendência.

Suponha dois projetos. Qual projeto criará o melhor sistema? Aquele em que o tempo médio entre falhas permanece, *release* após *release*, flutuando em torno de uma semana, ou aquele em que o tempo médio entre falhas vem aumentando significativamente, apesar de partir de um patamar mais baixo, por exemplo dois dias?

No primeiro caso a métrica absoluta aparentemente indica um sistema de melhor qualidade, com uma menor frequência de falhas que o segundo sistema. Mas, observando ao longo do tempo, o sistema com frequência de falhas em torno de uma semana se mantém estável, (com falhas a cada semana), enquanto o segundo sistema demonstra que está evoluindo. Isto pode significar que a equipe do segundo sistema está fazendo a coisa certa em termos de depuração. Podemos esperar um sistema de melhor qualidade no segundo caso.

A conclusão é que a tendência em relação ao tempo é mais importante que o valor absoluto num determinado instante. Com relação à pergunta acima, se pensarmos a métrica **maturidade** de software como robustez e baixa frequência de erros, verificamos que seu comportamento no tempo é claramente mais importante do que o valor absoluto. A maturidade deve aumentar acompanhando o aumento do valor do MTBF (ou pseudo MTBF, Mean Time Between Failures, conceito tomado emprestado do pessoal de *hardware*).

Este enfoque reconhece o caráter inerentemente dinâmico de um processo de desenvolvimento iterativo e ao invés de se concentrar no valor absoluto da medida, prefere, explicitamente, focalizar a primeira derivada ou a mudança com respeito ao tempo. A combinação do valor corrente com a tendência corrente, é capaz de fornecer um indicador muito melhor para a tomada de decisão gerencial.

5. O Que um Projeto Precisa Medir?

Muitas métricas diferentes podem ser de valor para um processo iterativo. Apresentaremos um conjunto de sete métricas básicas para qualquer projeto de *software* e que demonstraram sua utilidade na prática. Estes indicadores têm a virtude de serem simples, fáceis de se coletar e interpretar. Três são indicadores de gerenciamento e quatro são indicadores de qualidade.

Indicadores de Gerenciamento

- Trabalho e Progresso.
- Orçamento e Despesas.
- Alocação e Rotatividade na Equipe.

Indicadores de Qualidade

- Fluxo de Mudanças e Estabilidade.
- Fragmentação e Modularidade.
- Retrabalho e Adaptabilidade.
- Tempo médio entre falhas e Maturidade.

Ao coletar estes indicadores ao longo do tempo devemos escolher a abrangência da coleta, isto é, definir se estaremos coletando por semana, por iteração, por módulo do sistema, por equipe, etc. Recomendamos, no mínimo, a coleta por iteração para acompanhamento do projeto e a coleta por módulo para exame da qualidade do produto.

As sete métricas são, portanto, categorias de métricas. São derivadas de métricas primitivas e geram os indicadores de gerenciamento e qualidade citados acima. Numa situação real, elaboramos o artefato do RUP **Measurement Plan** e especificamos como cada um dos indicadores será obtido: qual métrica será coletada e como será o cálculo. Em resumo, a seguinte lista deve fazer parte de qualquer plano de métricas de projeto:

Progresso	Trabalho e Progresso: trabalho realizado no tempo	por Iteração	por módulo
Despesa	Orçamento e Despesas: gastos no tempo	por Iteração	por módulo
Rotatividade	Alocação e Rotatividade na equipe: mudança de pessoal ao longo do tempo	por Iteração	
Estabilidade	Fluxo de Mudanças e Estabilidade: fluxo de mudanças no tempo	por Iteração	por módulo
Modularidade	Fragmentação e Modularidade: fragmentação média por mudança ao longo do tempo	por Iteração	por módulo
Adaptabilidade	Retrabalho e Adaptabilidade: retrabalho médio por mudança ao longo do tempo	por Iteração	por módulo
Maturidade	Tempo médio entre falhas e Maturidade: taxa de defeitos ao longo do tempo	por Iteração	por módulo

6. Descrição das Sete Métricas

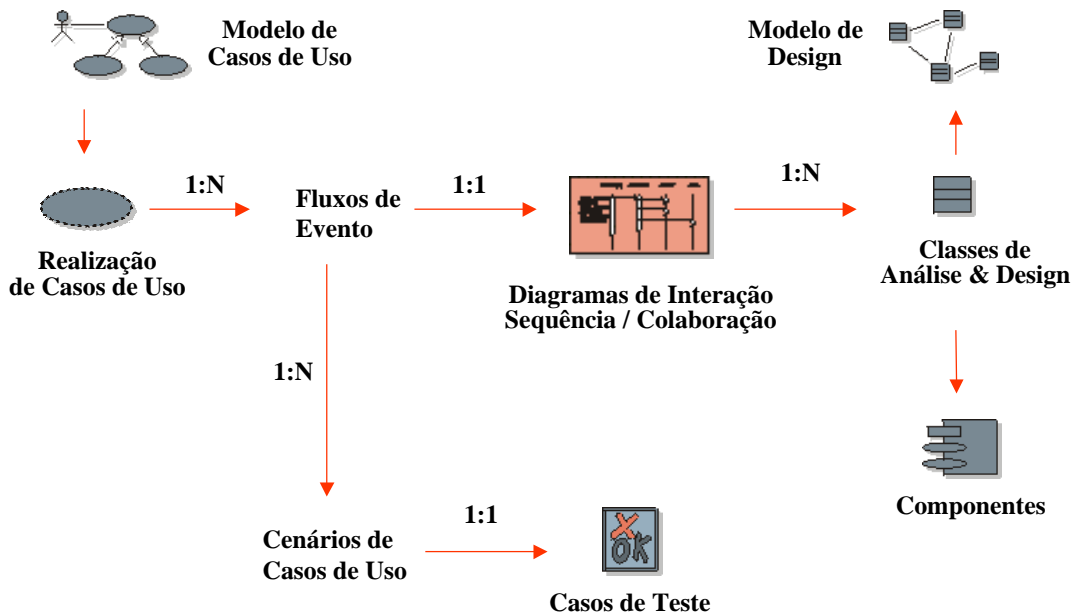
Para aplicar cada uma das 7 Métricas, imaginemos um contexto simples de projeto iterativo onde as informações que desejamos estão disponíveis em uma ou mais ferramentas. Estas informações não foram geradas intencionalmente pela equipe, mas coletadas através de métodos automáticos. É importante lembrar, (mesmo sendo óbvio) que deve existir um plano de projeto. Este plano, no contexto do RUP, é composto de um plano geral, o **Project Plan** e um plano específico para cada iteração, os **Iterations Plans**.

6.1 Progresso – Trabalho realizado ao longo do tempo

Para analisar o progresso precisamos confrontar o trabalho realizado ao longo do tempo com o trabalho planejado. Devemos usar uma medida objetiva como, por exemplo, casos de uso, cenários de casos de uso, classes ou objetos criados e guardados sob a baseline de um controle de configuração.

Surge uma questão: progresso do ponto de vista de quem? Do analista de sistemas, do designer, do implementador ou do testador? Para cada um destes pontos de vista a medida de progresso é diferente. Por exemplo, o designer pode considerar progresso o número de fluxos de eventos de casos de uso analisados e transformados em diagramas de interação. O testador pode considerar progresso o número de cenários de casos de uso analisados e transformados em casos de teste. Em geral o gerente do projeto deverá definir um ou mais pontos de vista para analisar e compor o que será denominado “progresso do projeto”.

Para definir a métrica apropriada precisamos entender o processo de desenvolvimento que está sendo aplicado. Suponha que o seguinte esquema, bem simplificado, represente o processo de um determinado projeto:



Aplicando as Sete Métricas de Controle de Projetos

Observem nesta figura as relações entre os diagramas de interação e os fluxos de eventos de um caso de uso. A proporção indicada é 1:1, (em geral isto é exato apesar de não ser uma lei absoluta). Por exemplo, se o caso de uso em análise possui 12 fluxos de eventos, esperamos criar 12 diagramas de interação para realizar a sua análise completa.

Compare a situação anterior com as proporções 1:N nos casos das relações entre um caso de uso e seus fluxos e um diagrama de interação e suas classes (ver figura). Nestes casos não sabemos de antemão quantos elementos precisaremos na etapa seguinte, sendo difícil avaliar progresso. É possível uma análise destes casos 1:N, mas precisaríamos de históricos de produtos similares, incorrendo novamente em imprecisões maiores de estimativas.

Verificamos que os melhores pontos para métricas de progresso são aqueles em que há transformações 1:1, pois permitem a comparação entre o esperado e o realizado. Boa parte do trabalho de implantação de métricas é identificar as relações 1:1, no caso específico de desenvolvimento do cliente, e indicar a métrica apropriada para capturar a tendência de progresso.

Elaboramos então uma tabela simples que relaciona o papel (quem ou qual ponto de vista) e as métricas primitivas adotadas. Por clareza podemos indicar a fonte e o uso principal, isto é, se a categoria for “1:1” temos uma medida para o progresso atual do projeto, se a categoria for “1:N”, temos apenas dados com finalidade histórica. Por exemplo:

Métrica	Coleta da Informação	
	Artefato fonte	Categoria/contexto
Designer/implementador		
Progresso		
n° de diagramas de interação X fluxos de evento	RequisitePro, Rose	“1:1”, Design
n° de classes	Rose	“1:N”, Histórico
n° de operações	Rose	“1:N”, Histórico
n° de atributos	Rose	“1:N”, Histórico
n° de linhas de código fonte (SLOC)	ClearCase	“1:N”, Histórico
Quantidade de retrabalho (em SLOC)	ClearCase	“1:N”, Histórico
Analista		
Progresso		
n° casos de uso detalhados X identificados	RequisitePro, Rose	“1:1”, Requisitos
Testador		
Progresso		
n° casos de teste X cenários de casos de uso	TestManager	“1:1”, Testes
Gerente de Projeto		
n° casos de uso realizados X casos de uso planejados	RequisitePro	“1:1”, Projeto
n° casos de teste com sucesso X n° casos de teste	TestManager	“1:1”, Projeto
n° de CRs fechadas X n° de CRs abertas	ClearQuest	“1:1”, Projeto
n° de critérios de avaliação atingidos	Plano de Projeto	“1:1”, Projeto
Composição do progresso de cada papel anterior	Fontes anteriores	“1:1”, Projeto



Aplicando as Sete Métricas de Controle de Projetos

Observe, por exemplo, a medida de progresso para o trabalho do testador: verificamos o número de casos de teste que o testador cria em sua atividade e comparamos com o número de cenários de casos de uso que deverão gerar casos de teste. É uma medida local ao time de testes. Por outro lado, o número de casos de teste passados com sucesso, versus o número total de casos de teste, fornece uma medida do progresso geral do projeto sob o ponto de vista do gerente de projeto.

A tabela acima não está completa, é apenas uma ilustração. O importante é perceber que o trabalho planejado para a iteração está sendo verificado em diversos níveis e comparado com o realizado. Nem todos os números estão disponíveis ao mesmo tempo porém, na medida em que a iteração prossegue, surgem os valores que irão possibilitar as demais comparações. Por exemplo: caso adotemos a medida de número de casos de uso, somente depois de detalhados alguns deles teremos o número de fluxos de eventos (ou cenários), que são a base para as medidas de progresso comparando diagramas de interação ou casos de teste.

Somente um conjunto de medidas pode confirmar as conclusões sobre as métricas e dessa forma contextualizar adequadamente o que medimos. Pontos de função e pontos de casos de uso podem ser usados mas dependem de um histórico e seriam classificados como categoria "1:N" em nossa visão. O número de linhas de código fonte (SLOC), depois de uma ou duas iterações, torna-se uma boa medida de trabalho, principalmente do ponto de vista do implementador. Também são úteis os números de CR's fechadas e os critérios de sucesso atingidos na avaliação de final de iteração.

6.1.1 Cenário de aplicação

Suponha um novo projeto, que tenha sido planejado direitinho... quer dizer adequadamente e, no momento, esteja na última iteração da fase de Elaboration, pronto para trocar de fase. Este projeto foi planejado com a previsão de 25 casos de uso, um projeto médio. Usaremos, por simplicidade, a métrica número de casos de uso, mas o raciocínio é equivalente para cenários de casos de uso (preferível) assim como para as demais métricas citadas anteriormente.

	I	E		C		T	
	# 1	# 1	# 2	# 1	# 2	# 1	
# UC planejado	0	2	5	9	9	0	} Progresso Projeto
# UC plan. acumulado	0	2	7	16	25	25	
% UC plan. acumulado	0	8	28	64	100	100	
# UC realizado	0	2	5	0	0	0	} Progresso Requisitos
# UC real. acumulado	0	2	7	7	7	7	
% UC real. acumulado	0	8	28	28	28	28	
% Sistema			25%				
# UC identificados	25						
# UC detalhados	0	10	10				
# UC detalh. acumulado	0	10	20				
% UC detalh. acumulado			80%				
% Modelo de UC							



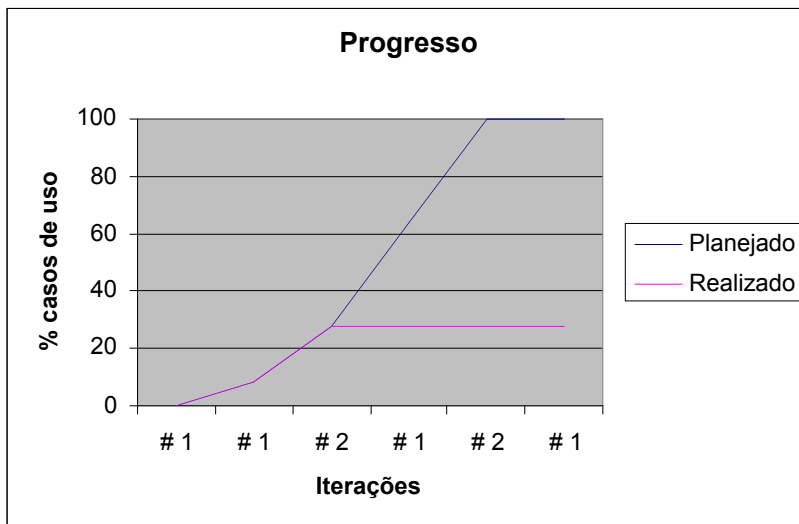
6.1.2 Interpretação

O projeto está se comportando como esperado: os casos de uso estão sendo realizados nas iterações previstas (2 e 5 na primeira e segunda iterações de Elaboration). Nenhuma surpresa alterou o planejado e, do ponto de vista do pessoal de requisitos, foi possível, além de detalhar os casos de uso necessários para as iterações, atingir a meta de detalhar 20 casos de uso até o final de Elaboration. Esta meta não é crítica do ponto de vista de prazos, mas é crítica do ponto de vista de alocação de recursos, pois a suposição do processo é que na próxima fase haverá uma mudança no perfil da equipe, de uma predominância de designers e analistas, para a predominância de implementadores.

Observem as medidas de boa formação em relação ao esqueleto de projeto planejado: 25% do sistema e 80% do modelo de casos de uso ao final de Elaboration. Estas são heurísticas ou “métricas” de boa formação que o processo oferece para sabermos se o plano geral do projeto está conforme a um projeto iterativo típico.

Em um projeto real seria necessário comparar as demais medidas (cenários, classes, casos de teste, etc), para verificar se estas confirmam as conclusões sobre a métrica casos de uso, mas neste exemplo estamos interessados apenas na mecânica básica.

Caso queira, o gerente de projeto pode criar um gráfico indicando o progresso ao longo das iterações e/ou ao longo do tempo. Neste gráfico fica evidente o cumprimento do planejado até a data de hoje.



6.2 Despesa - Gastos ao longo do tempo

As despesas ao longo do tempo devem ser comparadas com o custo orçado. Para a maioria dos projetos de software, onde é intenso o uso de serviço intelectual, o perfil de gastos geralmente segue o perfil de alocação de pessoal. Portanto, a menos de gastos pontuais, é possível criar uma curva de planejamento de custos baseada na curva de alocação de pessoal. (No RUP há informações adequadas para alocação e dimensionamento da equipe).

Aplicando as Sete Métricas de Controle de Projetos

Em todo projeto é importante uma visão geral das despesas em valores monetários. Por outro lado alguns projetos poderão ser melhor controlados se enxergarmos as despesas em termos de homens-hora. Despesa aqui seria o custo correspondente ao esforço para realizar uma atividade. Homens-hora [Hh], homens-mês [Hm], staff-month [Sm], etc, são medidas de esforço. Em diversos projetos a despesa em valor monetário é apenas a multiplicação do esforço por um fator fixo. Portanto podemos escolher uma unidade adequada para a medida de despesa em nossos projetos.

Além do valor intrínseco de se conhecer o comportamento das despesas de um projeto, percebemos que a comparação das despesas planejadas para incorporar o trabalho técnico, comparadas com os gastos efetivamente realizados pode nos dar uma medida de progresso e desempenho. O método que nos fornece esta medida é o conhecido Earned Value Analysis e pode ser encontrado na referência PMBOK – PMI.

Neste método combinamos orçamento, despesa e porcentagem do trabalho completado (medido na mesma unidade que orçamento ou despesa), para calcular indicadores para o projeto. O conceito fundamental que permite estes cálculos é o conceito de **earned value** ou valor atribuído, que também podemos interpretar como valor adquirido ou incorporado.

Earned Value *Porcentagem do orçamento igual à porcentagem do trabalho completado ou “o custo orçado do trabalho realizado”. Em determinadas implementações de Earned Value define-se apenas duas opções para a avaliação do trabalho completado: feito ou não feito, isto é, 0% ou 100%. Outras implementações admitem faixas, por exemplo, 0%, 30%, 70%, 100%, correndo o risco de maior subjetividade.*

Vejamos as possibilidades de unidades e seus respectivos indicadores na tabela seguinte:

Métrica	Natureza da Informação		
	Valor \$	Esforço	Duração
Gerente de Projeto			
Despesa			
Orçamento X Despesa	R\$	Hh, Hd, Hm	
Progresso: Earned Value Analysis			
Earned value	R\$	Hh	dias
Cv - Cost variance	R\$	Hh	dias
Sv - Schedule variance	R\$	Hh	dias
Cpi - Cost performance index	%	%	%
Spi - Schedule performance index	%	%	%
% Complete	%	%	%
Productivity	%	%	%

Estes indicadores são obtidos a partir do plano de custos do projeto juntamente com o plano de tarefas. Para calculá-los é mais fácil visualizar o seguinte exemplo:

Um projeto, com 2 tarefas, está sendo avaliado ao final da primeira tarefa (T1). A primeira tarefa foi terminada com atraso, coincidindo seu final com a metade do tempo da segunda tarefa (T2). Então, no momento da avaliação do projeto (tempo t) a segunda tarefa já deveria estar em plena execução. Portanto a tarefa T1 custou mais, tanto em tempo como em dinheiro.



Aplicando as Sete Métricas de Controle de Projetos

Orçamento total = 80.000 R\$
Orçamento T1 = 40.000 R\$
Orçamento T2 = 40.000 R\$

Como a tarefa terminou, independe do método de avaliação implementado, está feita com 100%.

Despesa até o momento $t = 50.000$ R\$
Orçamento até o momento $t =$ Orçamento T1 + Orçamento T2 = 60.000 R\$

Earned value = 40.000 R\$

$Cv = \text{Earned value} - \text{Despesa } t = 40.000 - 50.000 = - 10.000$ R\$
 $Sv = \text{Earned value} - \text{Orçamento } t = 40.000 - 60.000 = - 20.000$ R\$

$Cpi = (\text{Earned value}/\text{Despesa } t)100 - 100 = - 20 \%$
 $Spi = (\text{Earned value}/\text{Orçamento } t)100 - 100 = - 33 \%$

% Complete = $(\text{Earned value}/\text{Orçamento total})100 = 50 \%$

Productivity = $(\text{Earned value}/\text{Despesa } t)100 = 80 \%$

Em um relatório para supervisão do projeto poderíamos dizer:

- Gastamos R\$ 10.000 a mais que o previsto
- Estamos 20 % acima do orçamento
- Estamos 33 % além do prazo
- O trabalho está 50 % completo

Provavelmente nossas estimativas não estavam boas e podemos melhorá-las, desde que a tarefa restante seja similar a anterior e as demais condições de contexto não se alterem. Então, para ajustar o planejamento faríamos:

Orçamento T2 _{Revisado} = Orçamento T2/Productivity = 50.000 R\$

Um relatório contendo novas estimativas diria:

- Gastaremos mais R\$ 50.000 para terminar o projeto
- O custo total provável será de R\$ 100.000

O método do Earned value pode ser aplicado com unidades de esforço, neste caso é preciso compreender como as tarefas foram conduzidas. Vejamos:

Esforço estimado total = 480 Hh
Esforço estimado T1 = 240 Hh
Esforço estimado T2 = 240 Hh

Despesa até o momento $t = 50.000$ R\$, correspondente ao pagamento de 2 recursos durante o período original de execução da tarefa T1, mais 1 recurso que continuou trabalhando para completar a tarefa atrasada.

A tarefa original consumiu 120 horas de 2 pessoas, completar a tarefa consumiu 60 horas de 1 pessoa. Isto gerou a diferença de custo para a tarefa T1 (R\$ 50.000 e não R\$ 40.000 orçados).



Aplicando as Sete Métricas de Controle de Projetos

Esforço realizado até o momento $t = 240 \text{ Hh} + 60 \text{ Hh} = 300 \text{ Hh}$

Esforço estimado até o momento $t = \text{Esforço estimado T1} + \text{Esforço estimado T2} = 360 \text{ Hh}$

Earned value = 240 Hh

$Cv = \text{Earned value} - \text{Esforço realizado } t = 240 - 300 = -60 \text{ Hh}$

$Sv = \text{Earned value} - \text{Esforço estimado } t = 240 - 360 = -120 \text{ Hh}$

$Cpi = (\text{Earned value} / \text{Esforço realizado } t) 100 - 100 = -20 \%$

$Spi = (\text{Earned value} / \text{Esforço estimado } t) 100 - 100 = -33 \%$

$\% \text{ Complete} = (\text{Earned value} / \text{Esforço estimado total}) 100 = 50 \%$

$\text{Productivity} = (\text{Earned value} / \text{Esforço realizado } t) 100 = 80 \%$

Obtendo os mesmos indicadores do método com valores monetários, como não poderia deixar de ser.

6.2.1 Cenário de aplicação

Suponha o mesmo projeto, descrito no cenário de aplicação anterior, que estava prestes a mudar de fase. Nesse projeto estão previstos 25 casos de uso e eles estão sendo construídos dentro do esperado. Já foram elaborados 7 casos de uso de acordo com os prazos do planejamento original. Durante a revisão formal que está examinando os critérios para mudar de fase, todos estão contentes... O diretor financeiro porém, como não lhe foi servido café naquela manhã, resolveu pedir uma análise sobre os gastos do projeto. Então lhe foi mostrada a seguinte planilha:

	I		E		C		T		
	# 1	# 1	# 2	# 1	# 2	# 1			
Custo variável	30,0	70,0	180,0	220,0	220,0	30,0	[k \$]	} Planejado	
Custo fixo	5,0	5,0	5,0	5,0	5,0	5,0	[k \$]		
Orçamento	35,0	75,0	185,0	225,0	225,0	35,0	[k \$]		
Orçamento acumulado t	35,0	110,0	295,0	520,0	745,0	780,0	[k \$]		
% Orçamento acumulado	4	14	38	67	96	100	[%]		
Despesa	40,0	120,0	210,0	0,0	0,0	0,0	[k \$]		} Realizado
Despesa acumulada t	40,0	160,0	370,0	370,0	370,0	370,0	[k \$]		
% Despesa acumulada	5,1	20,5	47,4	47,4	47,4	47,4	[%]		
Earned value	35,0	110,0	295,0	520,0	745,0	780,0	[k \$]	} Earned Value Analysis	
Cv - Cost variance	-5,0	-50,0	-75,0	150,0	375,0	410,0	[k \$]		
Sv - Schedule variance	0,0	0,0	0,0	0,0	0,0	0,0	[k \$]		
Cpi - Cost perf. index	-12,5	-31,3	-20,3	40,5	101,4	110,8	[%]		
Spi - Schedule perf. index	0,0	0,0	0,0	0,0	0,0	0,0	[%]		
% Complete	4,5	14,1	37,8	66,7	95,5	100,0	[%]		
Productivity	87,5	68,8	79,7	140,5	201,4	210,8	[%]		

t →

Nesta planilha, Custo variável é a entrada de dados que corresponde a salário, taxas & benefícios e custos de infra-estrutura como luz, água, espaço, equipamentos individuais, etc. É o chamado “custo carregado” (*loaded cost*), que é variável de acordo com o número de recursos empregados.

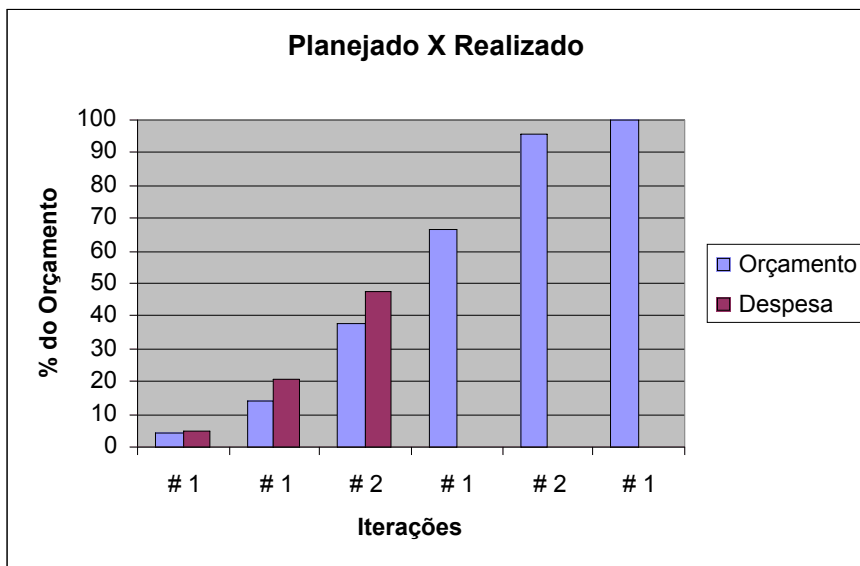
Custo fixo corresponde à compra e aluguel de equipamentos, ou serviços específicos para a realização desse projeto. (Neste exemplo, por coincidência, o custo fixo é constante ao longo das iterações).

A distribuição do orçamento seguiu o Plano de Projeto que, por definição foi planejado, “direitinho” como já dissemos anteriormente, respeitando o perfil de recursos e a duração das iterações como manda o RUP.

Despesa corresponde ao custo da materialização do trabalho realizado, é a soma de todos os custos diretos ou indiretos incorridos naquele período para a realização do trabalho naquelas iterações. Para encontrar culpados a Despesa poderia ser aberta em mais detalhes, mas não o foi. Para a análise que queremos será suficiente.

6.2.2 Interpretação

A partir de uma análise de despesa versus orçamento ficou claro que as iterações gastaram mais do que o previsto. Apesar de o projeto ainda estar dentro do orçamento, pois o gasto até o momento foi de 310 k\$ e o orçamento para o projeto é de 780 k\$, surge a questão de se esse comportamento geral vai se manter. Será possível reverter a situação realizando o restante do projeto dentro do orçado? Para mostrar o perigo ao diretor geral, o diretor financeiro utilizou o seguinte gráfico:



No entanto o gerente de projeto contra atacou com o argumento de que os prazos estavam sendo mantidos e isto indicava a saúde do projeto. Para reforçar sua posição mostrou os cálculos baseados no método Earned Value:

Orçamento total = 780 k\$

Orçamento até o momento t_i = 295 k\$

Despesa até o momento t_i = 370 k\$

Aplicando as Sete Métricas de Controle de Projetos

Earned value = 295 k\$

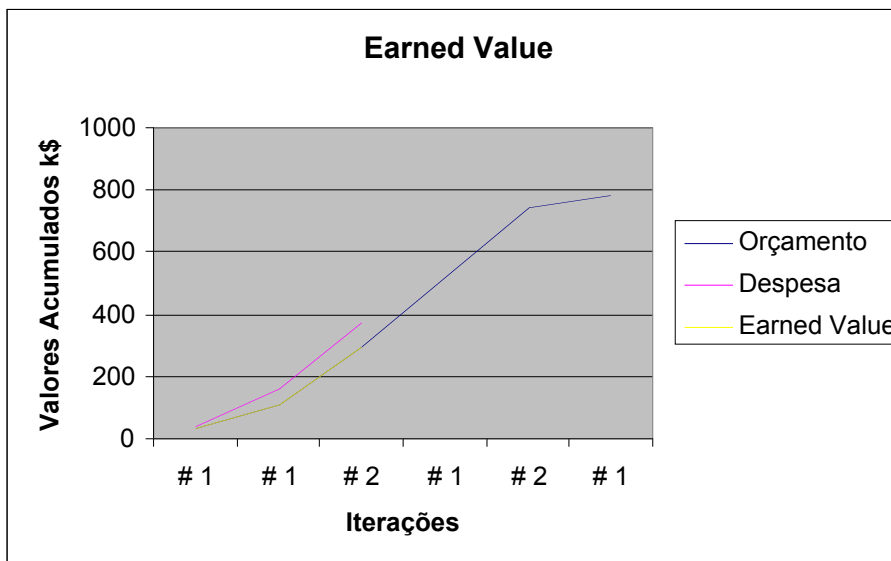
$Cv = -75,0$ k\$ indicando que foi gasto 75 k\$ a mais
 $Sv = 0,0$ k\$ indicando que não foi gasto nenhum tempo a mais
 $Cpi = -20,3$ % indicando ultrapassagem de orçamento em 20 %
 $Spi = 0,0$ % indicando cumprimento dos prazos

%Complete = 37,8 %
Productivity = 79,7 %

Orçamento da Iteração #1 de Construction Revisado = 282 k\$
Orçamento da Iteração #2 de Construction Revisado = 282 k\$
Orçamento da Iteração #1 de Transition Revisado = 43,9 k\$

Earned value final = 780 k\$
Despesa final = 977,9 k\$
Cpi final = - 20,2 %

Como o diretor geral não entendeu nada, o gerente de projeto resolveu mostrar um gráfico e fazer um resumo da situação:



- Gastamos apenas R\$ 75.000 a mais que o previsto
- Estamos infelizmente a 20,3 % acima do orçamento
- No entanto estamos rigorosamente dentro do prazo
- O projeto já está 37,8 % completo e, o que é mais importante, enfrentamos as questões mais difíceis e de risco maior em primeiro lugar, atacando o que chamamos aspectos de arquitetura do sistema. Logo a infra-estrutura da aplicação está construída, o que resta é acrescentar funcionalidade.
- Nossa produtividade foi de 79,7 %, isto é natural numa fase em que exploramos muito o domínio do problema. Após a mudança de fase podemos esperar que a produtividade aumente.

Aplicando as Sete Métricas de Controle de Projetos

Se esperamos realmente que a produtividade aumente, podemos usar a produtividade passada como um teste de hipótese: caso o projeto continue viável com a produtividade passada (mais baixa), então podemos aceitar que o projeto é viável pois a produtividade irá aumentar.

O gerente de projeto revisou o planejado para as próximas iterações e acrescentou:

- Gastaremos mais R\$ 607.900 para terminar o projeto
- O custo máximo total não ultrapassará R\$ 977.900
- O excesso máximo no orçamento não ultrapassará 20,2 %
- Terminaremos no prazo! (gulp!)

... Desta vez passou. _

6.3 Rotatividade - Mudança de pessoal ao longo do tempo

Em um projeto iterativo típico, uma equipe pequena irá trabalhar nas primeiras iterações para a criação de protótipos, levantamento de requisitos e definição de arquitetura. Após isto, o time crescerá para uma alocação de construção a pleno vapor. Além do volume de alocação variar conforme a fase, outro fator irá contribuir para a variação da equipe: a rotatividade. O acompanhamento gerencial consiste em monitorar o *turnover* – a variação da equipe, a adequação dessa equipe à demanda de trabalho na iteração, a saída prematura de membros da equipe, o clima interno, etc. É necessário um plano de alocação, com perfis e duração previamente estabelecidos, que permitirá comparar os recursos realmente engajados.

Além de examinar a variação de pessoal, precisamos avaliar os momentos críticos quando a carga de trabalho exigirá mais recursos. Dada uma determinada alocação de pessoal, projetamos a **Labor Rate** (taxa de trabalho) esperada. Quando ocorre uma variação na equipe, verificamos, observando a Labor Rate, o quanto de sobretrabalho esse evento ocasionou.

$$\text{Labor Rate} = \text{Recursos alocados} \times \text{tempo} / \text{Carga de trabalho}$$

6.3.1 Cenário de aplicação

A reunião de revisão formal deu o sinal verde para que o projeto vá para a fase de Construction. No elevador, após a saída da reunião, o comentário geral foi de que o gasto a mais era irrelevante... e de certa forma natural, pois o projeto havia contratado serviços de consultoria para cobrir a falta de um membro importante que pedira demissão. O gerente de projeto aproveitou para jogar a culpa de pequenas falhas sobre o ex-colega e anotou mentalmente a tarefa de verificar se a alocação de recursos continuava adequada. Em seu gabinete construiu a seguinte planilha:

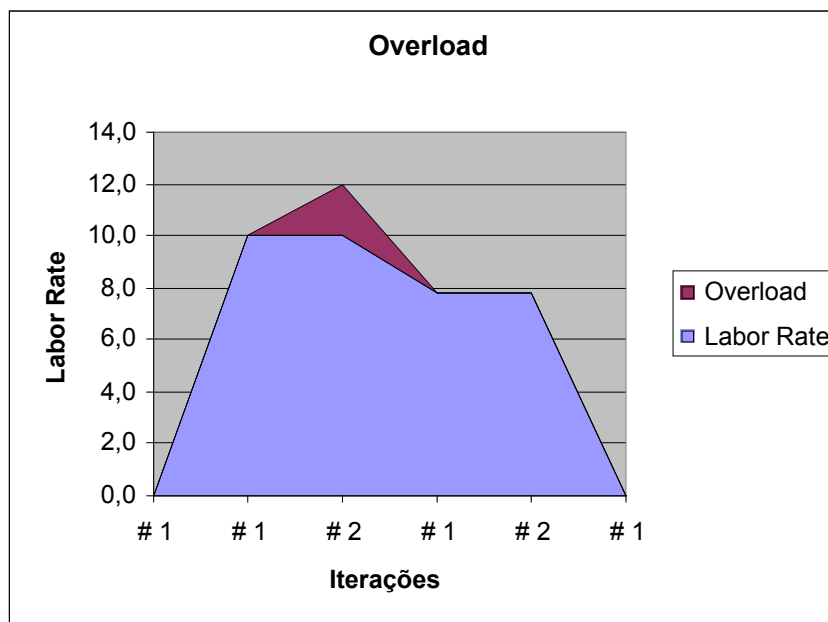
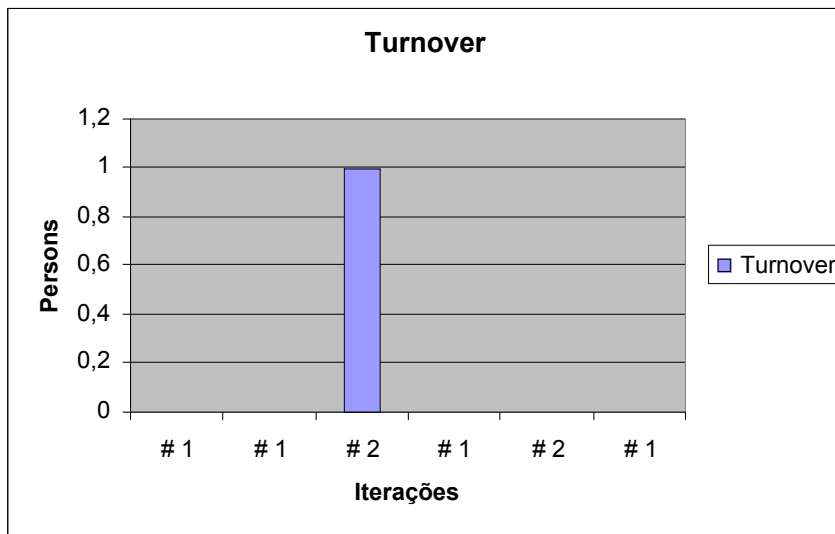
	I		E		C		T	
	# 1	# 1	# 2	# 1	#	2		
# UC planejado	0	2	5	9	9	0		
Δt Iteração planejado	4	5	10	10	10	4		
# Recursos planejados	2	4	5	7	7	4		
Labor Rate planejada	0,0	10,0	10,0	7,8	7,8	0,0	[Hw/UC]	43 [w] total
# Recursos alocados	2	4	4	7	7	4		
Labor Rate alocada	0,0	10,0	8,0	7,8	7,8	0,0	[Hw/UC]	
ΔLR overload	0,0	0,0	2,0	0,0	0,0	0,0		
Turnover	0	0	1	0	0	0		saída / entrada



6.3.2 Interpretação

Um membro da equipe saiu na segunda iteração de Elaboration, isto ocasionou a rotatividade indicada na última linha da planilha e abaixou a Labor Rate de possíveis 10,0 Hw/UC para 8,0 Hw/UC. Isto quer dizer que, se o projeto foi planejado no limite da carga de trabalho, qualquer diferença entre a LR planejada e a LR realmente alocada pode significar uma sobrecarga (LR_{overload}) que deve ser resolvida gerencialmente.

Observem que, em termos de medidas de boa formação, o projeto encontrava-se ligeiramente defasado em relação a um projeto iterativo típico (LR entre 12 e 8 Hw/UC para Elaboration e Construction respectivamente - para equipes em aprendizagem do processo iterativo). Nosso gerente de projeto não deu maior atenção a este fato, provavelmente considerando a boa experiência de sua equipe e ao fato do assunto do projeto ser de relativa simplicidade. Em todo o caso construiu dois gráficos para analisar mais rapidamente os acontecimentos.



A sobrecarga de 2,0 Hw/UC ficou evidente no gráfico, isto prova que o projeto foi planejado um pouco “tensionado” e portanto não estava imune à rotatividade. As vezes, esta situação é inevitável, outras vezes, é perfeitamente evitável. Em todo o caso, se tivesse construído o gráfico antes talvez as coisas pudessem ter sido diferentes mas... águas passadas não movem moinho.

6.4 Estabilidade - Fluxo de mudanças ao longo do tempo

O fluxo de mudanças (ou de alterações no projeto) ao longo do tempo é definido como o número de solicitações de mudança abertas e fechadas durante o ciclo de vida de desenvolvimento. A estabilidade é vista como a relação entre solicitações abertas versus fechadas. Quanto mais próximas as duas medidas, mais estável é o processo. Esta informação, associada ao plano de *releases*, nos dará previsibilidade de cronograma e avaliação da carga de trabalho por fazer (ou refazer).

Seja N o número total de solicitações de mudança.

Fluxo de mudanças, será o número total de solicitações de mudança abertas menos as fechadas:

$$\text{Fluxo de mudanças} = N_{\text{abertas}} - N_{\text{fechadas}}$$

Chamemos CR (Change Request) uma solicitação de mudança.

Seja: CR_0 (tipo0) n° de mudanças referentes a defeitos críticos.

CR_1 (tipo1) n° de mudanças referentes a defeitos normais.

CR_2 (tipo2) n° de mudanças referentes a melhorias no sistema.

CR_3 (tipo3) n° de mudanças referentes a novas características.

$$\text{Fluxo de mudanças de novas características (tipo3)} = CR_3_{\text{abertas}} - CR_3_{\text{fechadas}}$$

Mede quantos novos requisitos o seu cliente solicitou e você implementou.

$$\text{Fluxo de mudanças de melhorias (tipo2)} = CR_2_{\text{abertas}} - CR_2_{\text{fechadas}}$$

Mede quantas melhorias foram sugeridas a partir de avaliações do sistema em desenvolvimento.

$$\text{Fluxo de mudanças de defeitos (tipos 0 e 1)} = (CR_0 + CR_1)_{\text{abertas}} - (CR_0 + CR_1)_{\text{fechadas}}$$

Mede quantos defeitos foram descobertos e corrigidos.

Definimos estabilidade como a tendência das solicitações de mudança ao longo do tempo. Esta definição, mais genérica, atende ao fato de podermos analisar estabilidade de várias formas. Seja pelo fluxo líquido das mudanças implementadas ou, diretamente, pelo número de CRs ao longo do tempo.

Note que um alto fluxo de mudanças, isto é, um alto valor absoluto de $N_{\text{abertas}} - N_{\text{fechadas}}$, pode significar um problema pontual para o projeto (por causa de consumo de mão de obra, por exemplo). No entanto, a observação da tendência desse valor ao longo do tempo é mais informativa do que seu valor absoluto: terá o projeto condições de absorver o fluxo de mudanças que virá no futuro? Essa tendência é denominada estabilidade.

6.4.1 Cenário de aplicação

O mesmo projeto descrito nos cenários anteriores encontra-se na primeira iteração de Construction. O gerente de projeto está planejando a próxima iteração e para isto precisa verificar, mais uma vez, se o dimensionamento de recursos está correto. Então resolveu consultar todas as novas requisições de funcionalidade feitas ao projeto, pois estas devem se somar aos casos de uso e cenários que faltam por fazer.

Ao examinar as requisições elaborou a seguinte planilha, a qual demonstra que há diversas naturezas de trabalho ainda por fazer:

- O trabalho inicialmente alocado (os casos de uso planejados);
- O trabalho decorrente da correção de defeitos;
- O trabalho adicional de novas características.

	I	E		C		T	
	# 1	# 1	#	2	#	1	#
N total abertas	0	45	40	20		0	0
N total fechadas	0	42	30	5		0	0
N abertas - N fechadas	0	3	13	28		0	0
# CR0 abertas	0	12	10	5			
# CR1 abertas	0	30	26	10			
# CR2 abertas	0	0	0	0			
# CR3 abertas	0	3	4	5			
# CR0 fechadas	0	12	10	5			
# CR1 fechadas	0	30	20	0			
# CR2 fechadas	0	0	0	0			
# CR3 fechadas	0	0	0	0			

Um exame mais detalhado deixou preocupado nosso gerente, apesar do esforço da equipe em eliminar as pendências, há uma “dívida” de trabalho acumulada, evidenciada pela relação crescente $N_{abertas} - N_{fechadas}$.

6.4.2 Interpretação

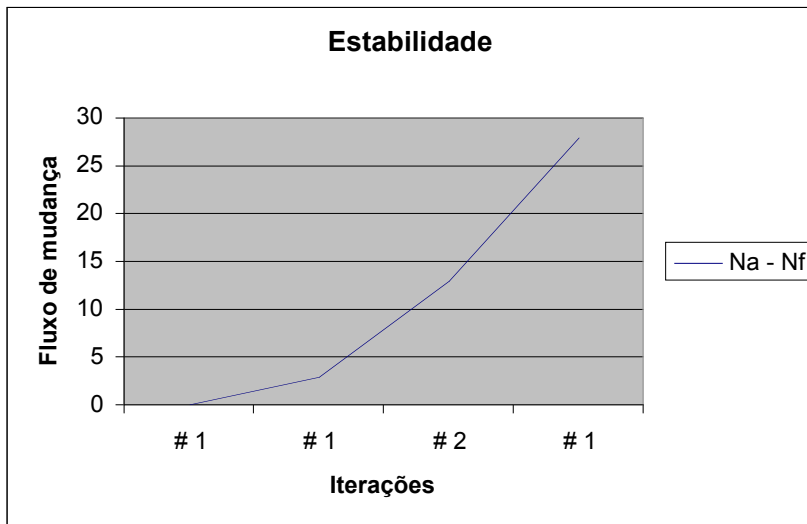
A direção do projeto permitiu que o cliente solicitasse novas características e estas foram aceitas como novos requisitos. As CRs de tipo 3 apontam para 3, 4 e 5 solicitações que se transformaram num número igual ou maior de novas características. Isto é normal num projeto iterativo, mas deve ser previsto na alocação de recursos. Provavelmente devido à carga de trabalho, estas novas solicitações não foram implementadas, ficando como *backlog* a ser saldado nas últimas iterações.

A equipe atacou todos os defeitos apontados, porém só eliminou totalmente os defeitos críticos (provavelmente uma decisão tática). Porém, os defeitos não críticos também precisam ser eliminados e constituem *backlog* para as próximas iterações.

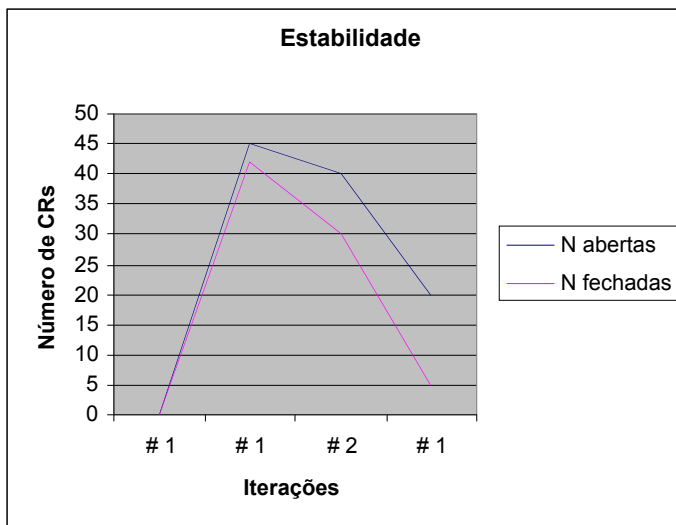
Aplicando as Sete Métricas de Controle de Projetos

Um projeto pode ser considerado estável se a carga de trabalho aumenta dia após dia? A resposta é pode, preservadas certas condições e desde que a capacidade de realização de trabalho – não importa o motivo – também aumente em proporção.

Para o projeto em questão, como tem sido a tendência geral dos números? Para representar essa tendência temos duas alternativas: efetuar a conta Nabertas – Nfechadas e construir a curva no tempo com o resultado do fluxo de mudanças acumulado (como no primeiro gráfico), ou simplesmente plotar duas curvas com os valores diretamente obtidos para Nabertas e Nfechadas (como no segundo gráfico).



Neste gráfico vemos o fluxo de mudanças crescente, não há estabilidade.



Neste gráfico vemos a divergência entre as curvas, quanto maior o *gap*, maior o perigo para o projeto.



As duas representações podem ser úteis para fins diferentes: A primeira demonstra mais intuitivamente a estabilidade do projeto, a segunda nos dá detalhes quanto à intensidade do fluxo de mudanças.

A estabilidade pode ser medida, portanto, como a relação entre N abertas e N fechadas ou resolvidas. Caso precisemos de maior profundidade na análise podemos usar indicadores mais detalhados como: CRs tipo0, tipo1, tipo2 e assim por diante. Podemos construir curvas no tempo para os abertos e fechados e comparar a proximidade ou afastamento entre elas.

6.5 Modularidade - Fragmentação média por mudança ao longo do tempo

Fragmentação é um termo que se refere a extensão da mudança, isto é, a uma certa quantidade de software que está em *baseline* e que precisa ser retrabalhado. Imagine que num dado momento haja uma bem definida quantidade de software produzida e validada, registrada na ferramenta de controle de versão como uma determinada *baseline*. Suponha que o relatório vindo de uma avaliação indique que uma certa quantidade desse software precisa ser refeita. Esta quantidade é denominada fragmento de mudança.

Medimos a fragmentação em duas perspectivas: A fragmentação devida ao retrabalho em aberto (a grandeza B, veja adiante), que é aquilo que está por fazer, mas que já identificamos. E a fragmentação devida ao retrabalho já fechado, isto é, aquilo que já fizemos (caracterizado pela grandeza F, veja adiante).

Definimos modularidade como a medida da localização da fragmentação, que indica quanto da fragmentação está localizada ou espalhada. Um aspecto importante da modularidade é a indicação da tendência da fragmentação média ao longo do tempo. Em um projeto com bom design, a expectativa é que a fragmentação fique estável ou diminua.

Seja N o número total de solicitações de mudança.

Seja B a quantidade estimada, em linhas de código, do retrabalho em aberto devido às mudanças originadas de defeitos e melhorias. (CRs de tipo0, tipo1 e tipo2).

Seja F a quantidade medida, em linhas de código, do retrabalho fechado devido às mudanças originadas de defeitos e melhorias. (CRs de tipo0, tipo1 e tipo2).

(O efeito das mudanças originadas de novas características, CRs do tipo 3, é muito variado e deve ser analisado à parte, refletindo um novo trabalho e não um re-trabalho).

Fragmentação média por CR = B/N na unidade [LOC]

ou

Fragmentação média por CR = F/N na unidade [LOC]

Definimos modularidade como a tendência de B/N e F/N ao longo do tempo. Estes indicadores fornecem uma visão do impacto da mudança em relação à futura manutenibilidade.

Por exemplo, num projeto que utilize técnicas orientadas a objeto, conforme a maturidade do software evolui, é razoável esperar que a fragmentação fique cada vez mais localizada. Afinal, é esta uma das principais características da técnica de orientação a objeto, encapsular e localizar mudanças.

6.5.1 Cenário de aplicação

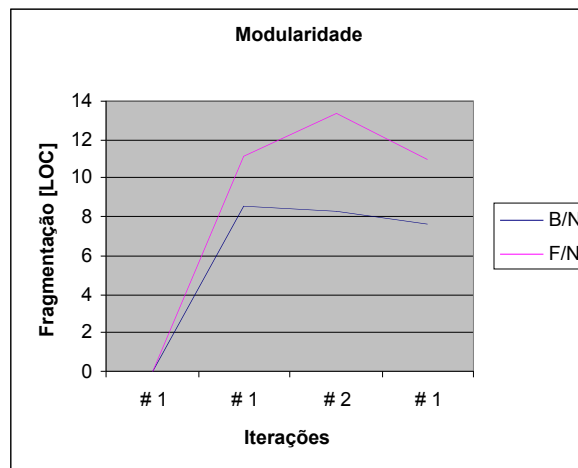
Nosso conhecido projeto está para iniciar a segunda iteração de Construction, neste momento preocupações quanto à qualidade do software começam a assaltar a mente do gerente de projetos. Um tanto tardiamente, é verdade, ele se pergunta se as constantes alterações não estão degradando a arquitetura elaborada com tanto cuidado. Será que meu produto terá qualidade ao cabo de sua construção? Resolve então montar uma planilha que indique como as mudanças afetaram o produto:

- Contou o número de linhas de código de cada iteração e acumulou até o presente;
- Contou o número de linhas de código estimadas para mudanças em cada iteração (B);
- Contou o número de linhas de código realmente alteradas em cada mudança (F);
- Identificou o número total das mudanças em aberto;
- Identificou o número total das mudanças fechadas;
- Calculou as fragmentações médias por solicitação de mudança: B/N e F/N

	I		E		C		T		
	# 1	# 1	# 2	# 1	# 2	# 1			
# SLOC acumulada	0	7000	15000	44000				44000	[LOC]
B	0	360	300	115				775	[LOC]
F	0	470	400	55				55	[LOC]
N abertas (tipo 0, 1, 2)	0	42	36	15					
N fechadas (tipo 0, 1, 2)	0	42	30	5					
B/N	0	9	8	8	0	0			[LOC]
F/N	0	11	13	11	0	0			[LOC]

6.5.2 Interpretação

Apesar de um surto inicial a tendência geral da fragmentação ao longo do tempo é de queda, e esta é a mais importante observação que estes indicadores permitem fazer. Um crescimento indicaria uma provável degradação na arquitetura, o que não está ocorrendo. Também os números absolutos de B/N e F/N são favoráveis, representam menos que 1% do código criado na iteração. Isto quer dizer que a mudança média tem impacto de menos de 1% no código existente. Gráficamente teremos:



No entanto podemos ver, pelo gráfico ou pela tabela, um outro dado curioso: o número de linhas de código realmente alterados numa mudança está sempre maior do que o estimado. Isto indica um comportamento otimista da equipe em relação à quantidade de trabalho que teria por fazer; mostra um desequilíbrio no processo de estimativa que a equipe está empregando. Esperemos que nosso gerente de projeto tenha percebido isto.

6.6 Adaptabilidade - Retrabalho médio por mudança ao longo do tempo

Podemos definir retrabalho como o esforço gasto com mudanças, isto é, o esforço ou carga de trabalho para analisar, implementar e re-testar as alterações no código que está em baseline. A adaptabilidade será, então, definida como a tendência desse retrabalho ao longo do tempo. Para um projeto saudável esperamos uma diminuição do retrabalho.

Seja N o número total de solicitações de mudança.

Seja E o esforço, em homens-hora, do retrabalho devido às mudanças originadas de defeitos e melhorias. (CRs de tipo0, tipo1 e tipo2).

$$\text{Retrabalho médio por CR} = E/N \text{ na unidade [Hh]}$$

Adaptabilidade será a tendência de E/N ao longo do tempo. A adaptabilidade nos permite uma visão sobre a facilidade com que o produto pode ser alterado; se a tendência do retrabalho é aumentar de valor, podemos interpretar como um mau sinal sobre a manutenibilidade futura do sistema.

6.6.1 Cenário de aplicação

Para confirmar as conclusões tiradas com a modularidade, de que a qualidade está sendo preservada apesar das mudanças, o nosso gerente de projeto resolveu verificar como foi o gasto de esforço com essas mudanças. Isto é necessário porque a modularidade aborda a questão do ponto de vista do produto, enquanto que nada é dito sobre os recursos usados para implementar as mudanças. Ele então, criou a seguinte planilha fazendo:

- Computou o esforço correspondente a implementação das mudanças em cada iteração (E);
- Lembrou-se de que havia definido apenas 2 pessoas das 4 da equipe para cuidar das implementações de mudanças;
- Identificou o número de mudanças realmente atendidas (fechadas);
- Calculou o valor do esforço médio por mudança (E/N).

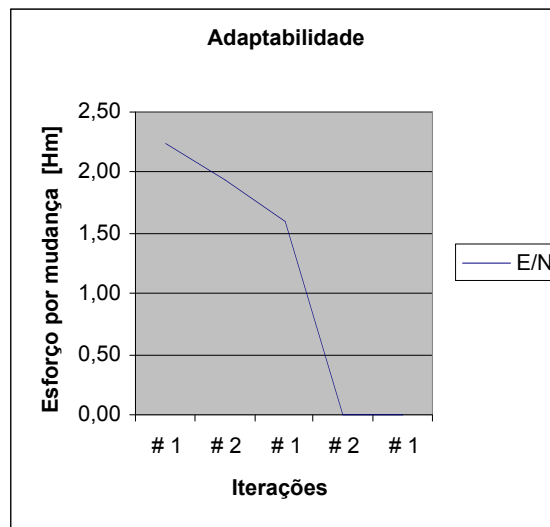
	I		E		C		T	
	# 1	# 1	# 2	# 1	# 2	# 1		
Esforço total acumulado	0	600	1800	3900				3900 [Hh]
E	0	94	58	8				160 [Hh]
N fechadas (tipo 0, 1, 2)	0	42	30	5				
E/N	0,00	2,24	1,93	1,60	0,00	0,00		[Hh]

6.6.2 Interpretação

O esforço médio por mudança vem decrescendo o que significa que as mudanças estão sendo cada vez mais fáceis de se implementar. Esta conclusão é corroborada pelo fato de que o número de mudanças diminuiu, o que exige que o esforço (E) tenha diminuído ainda mais.

Em um projeto iterativo, centrado em arquitetura, é normal que haja uma certa agitação inicial sobre o design da arquitetura. Soluções são refinadas e postas para testar, consumindo tempo e esforço durante o processo. Mas é um gasto benéfico, pois mudanças nesse momento irão indicar que o projeto está evoluindo. Porém, em breve, até o final da fase de Elaboration, o design de arquitetura deve estabilizar e o efeito do paradigma de objetos deve se fazer sentir: as mudanças seguintes precisam se mostrar localizadas, isto é, com classes bem definidas e responsabilidades bem projetadas para que a maioria das alterações recaiam na criação de métodos internos às classes. Então, de um modo geral, esperamos uma diminuição do esforço de implementação ao longo do projeto e, portanto, também uma diminuição do esforço médio por mudança.

Para visualizar construímos o seguinte gráfico:



Aparentemente o produto de software deste nosso exemplo está convergindo para uma situação de boa qualidade.

6.7 Maturidade - Taxa de defeitos ao longo do tempo

A primeira coisa que nos vem à cabeça ao falarmos de defeitos é o número absoluto de defeitos que um software possui. Esses defeitos se apresentam em falhas de operação ou são descobertos durante as fases de testes. É conveniente fazer uma distinção entre defeito e falha, pois o que observamos como usuários são as falhas ou sintomas, mas o que corrigimos no software são os defeitos (a doença).

Falha *Afastamento dos resultados esperados seja por comparação aos requisitos, seja por apresentar interrupção no funcionamento.*

Defeito *Imperfeição na especificação técnica, no design de um componente ou na escrita do código que, quando executada em determinadas condições, causa a falha observada.*

Assim, ao longo do tempo de operação ou durante os testes, vão aparecendo falhas que, expostas num gráfico, constituem a famosa curva de detecção de falhas. Por sua vez, os defeitos também podem ser analisados construindo curvas de descoberta de defeitos e curvas de densidade de defeitos.

Como, de maneira geral, os defeitos mais fáceis de detectar são os primeiros a serem detectados, o trabalho de testes fica mais difícil conforme vai avançando. Isto faz com que o formato da curva tenda a zero mas não o alcance. Como é freqüente, novas iterações ou novas manutenções introduzem novos defeitos, produzindo um formato dente de serra na curva geral.

Esse padrão de comportamento deve ser cuidadosamente acompanhado pela gerência para verificar a convergência em direção à qualidade do que se está construindo. Novamente: os valores absolutos são menos importantes do que a tendência; uma tendência a zero, mesmo partindo de patamares mais altos, significa um processo que está fazendo o seu trabalho de depuração.

Uma forma de medir a maturidade de um produto é determinar o tempo médio entre falhas (MTBF - Mean Time Between Failures). Este reflete a taxa de descoberta de defeitos quando certas condições de operação são satisfeitas. Portanto, maturidade pode ser definida como a taxa de descoberta de defeitos ao longo do tempo e nos fornece uma maneira de estabelecer o nível de confiança no produto.

Seja N o número total de solicitações de mudança.

Seja UT , tempo de operação, (ou Usage Time) o número de horas que certa baseline vem operando ou vem sendo testada, exercitando seus cenários sob condições realistas de operação.

$$MTBF = UT/N_{\text{tipo0+tipo1}} \text{ na unidade [h]}$$

Maturidade será a tendência do MTBF ao longo do tempo.

6.7.1 Cenário de aplicação

Nosso já familiar projeto está se aproximando do fim da fase de Transition. Nesta situação a maioria dos testes e verificações já foram feitas e o produto será liberado em breve. Alguns membros da alta gerência, dando ouvidos à conversas de corredor, questionaram a oportunidade de lançar um produto de qualidade duvidosa no mercado, a última “experiência” já foi o bastante, disseram.

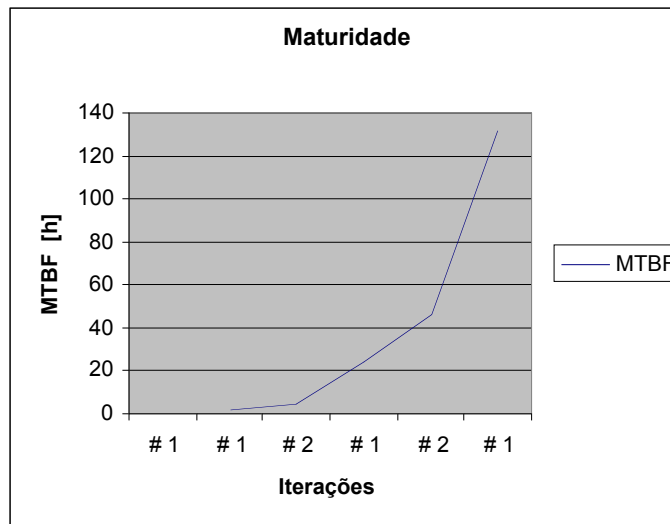
O gerente de projetos, pacientemente, resolveu demonstrar através de números que o seu software tem maturidade suficiente para ser liberado como prometido. Nem que para isso tivesse que “cozinhar” um pouquinho. Construiu a seguinte planilha computando:

- O tempo total em que o software esteve sob teste, considerando teste como operação (UT);
- Como a abordagem de testes foi iterativa e incremental, acumulou o tempo de teste de uma iteração com a anterior, considerando o acumulado como teste de regressão;
- Identificou o número total de mudanças abertas, originadas de defeitos ou falhas;
- Calculou o tempo médio dividindo UT/N.

	I		E		C		T	
	# 1	# 1	# 2	# 1	# 2	# 1		
UT (tempo de operação)		80	160	360	560	660		[h]
N abertas (tipo 0 e 1)		42	36	15	12	5		
MTBF		2	4	24	47	132		[h]

6.7.2 Interpretação

Realmente a curva de MTBF apresenta uma melhora ao longo das iterações, podemos visualizar mais facilmente no gráfico.



Pode-se perceber que houve uma melhora constante da operação do sistema. Onde ficaram as conhecidas degradações que acontecem no momento das integrações? Isto talvez esteja mascarado devido ao intervalo escolhido ter sido muito grande, uma iteração completa. De qualquer forma o resultado geral indica maturidade do processo, pois a equipe, com certeza, está fazendo corretamente a tarefa de depurar o produto.

No entanto, para chegarmos à mesma conclusão sobre a maturidade do produto ainda falta algum contexto. Que tipo de software é este que está sendo construído? Qual o nível de MTBF exigido pelos requisitos da área de domínio? Qual o nível de qualidade que a empresa deseja apresentar?

Nosso já conhecido diretor financeiro fez a pergunta mais difícil: Mas, será aceitável um ‘bug’ a cada duas semanas? E ainda consideramos nosso produto de qualidade! Observem que ao final do processo o sistema apresentará 132 horas de MTBF, uma falha provável em duas semanas ou em pouco mais do que cinco dias de operação contínua. E aí o debate pegou fogo na reunião e se estendeu por mais de cem anos de solidão..._

7. Conclusão

Métricas de tendência em relação ao tempo fornecem uma visão de como o processo e o produto estão evoluindo. O desenvolvimento iterativo lida com o gerenciamento de mudanças, e medir a mudança é o mais importante aspecto da coleta de métricas. Estas métricas são, portanto, ferramentas do gerente de projeto para diagnosticar e corrigir o rumo do projeto.

A preparação da equipe e do contexto do projeto são importantes e a automação da coleta de métricas é fundamental. Somente automatizando garantimos consistência e continuidade. Está claro também que o controle do projeto é só o início. Embutida nesta abordagem está a possibilidade de preparar o caminho para a correta coleta de métricas históricas, as quais, por sua vez, permitirão estimativas adequadas ao contexto da Organização.

—

Rational®

the software development company

Corporate Headquarters

18880 Homestead Rd.

Cupertino, CA 95014

Rational Software Latin America

Av . Dr. Cardoso de Melo, 1450 8º Andar

04548-005 - São Paulo - SP

Email: brasil-info@rational.com

Web site: www.rational.com

Rational, Rational logo, Rational the software development company, Rational Unified Process are trademarks of Rational Software Corporation. References to other companies and their products use trademarks owned by the respective companies and are for reference purpose only.

© Copyright 2002 by Rational Software Corporation.

Subject to change without notice. All rights reserved.

