UTILIZANDO O MÉTODO DE CUSTEIO POR ATIVIDADES NO DESENVOLVIMENTO DISTRIBUIDO DE SOFTWARE

Alexandre L'Erario
Engenharia de Produção - Escola Politécnica da Universidade de São Paulo
USP - São Paulo, Estado de São Paulo, CEP 05508-900, Brasil
alexandre.lerario@usp.br

Resumo. *Este artigo faz uma abordagem de como o método de custeio ABC pode ser utilizado para analisar o desenvolvimento distribuído de software. Para utilizar o método ABC, foi utilizado um modelo de dinâmica de desenvolvimento distribuído de software com o objetivo de extrais os procedimentos e direcionadores de custo.O modelo de dinâmica utilizado como parâmetro de análise é o M3DS, apresentado por L´Erario (2007).*

Palavras chaves*: Custeio baseado em atividades, Desenvolvimento distribuído de software,*

Abstract: *This work approaches ho activity-based costing method can be used to analyze the of distributed software development. To use the ABC, we used a dynamic model of distributed software development for extracting the procedures and cost drivers. The dynamic model used as a parameter of analysis is the M3DS, presented by L'Erario (2007).*

Keywords*: Activity-based costing, distributed development of software*

1. Introdução

Um dos efeitos da globalização, na indústria de software, é o crescimento de organizações que desenvolvem produtos de software de maneira distribuída. Este efeito, segundo Carmel (2001) é crescente e possibilitou que algumas organizações de software expandiram-se nos últimos anos.

O desenvolvimento distribuído de software ocorre quando vários nós/sites (unidade independente de produção de software distante das demais) cooperam e/ou colaboram para desenvolver um mesmo produto ou parte dele. Neste cenário, a complexidade do processo de desenvolvimento se amplia. A redução do tempo é a principal razão da divisão de tarefas na produção distribuída de software, porém o tempo de comunicação e de resposta entre os nós pode ser incomensurável. Segundo Herbsleb (2003) e Suzuki (1999), há muitas variáveis neste cenário, tais como a cultura, a língua, a capacidade de cada nó entre outras. Além disso, Becker (2001) afirma que desenvolver software em uma rede de produção requer gerenciamento mais eficaz de tarefas. L'Erario (2007) afirma que além do gerenciamento de tarefas é necessário e que o mecanismo de coordenação dos sites seja eficaz. Martin (1996) afirma também que este gerenciamento deve estender-se a ferramentas, capacidades e informação.

Identificar o custo do desenvolvimento distribuído pode auxiliar os gestores a decidirem a viabilidade de distribuir tarefas. Este artigo discute como o método de custeio

ABC pode ser utilizado para tal propósito. O modelo de dinâmica M3DS foi utilizado como referência na extração de procedimentos e direcionadores de custo.

O objetivo deste artigo é demonstrar como o ABC pode ser utilizado para determinar o custo do desenvolvimento distribuído de software. O custo do software em si, mensurado comumente em pontos por função, não faz parte do escopo deste trabalho.

2. Desenvolvimento distribuído de Software

O desenvolvimento distribuído de software (DDS) agrega várias vantagens sobre o desenvolvimento centralizado, como citan Battin (2001), Carmel (2001), L'Erario(2004) em seus estudos de caso. Segundo Battin (2001), foi possível desenvolver 511000 linhas de código para o projeto e cada centro de desenvolvimento colaborou significantemente para a implementação. L'Erario (2004) em seu estudo de caso, utilizou o desenvolvimento distribuído para a implementação de uma aplicação pervasiva. Para tanto foi elaborado um arcabouço de distribuição de tarefas para coordenar a entrega de partes da aplicação pelos nós. Carmel (2001) descreve que o número de organizações que aderem ao GSD (*Global Software Development*) é crescente. Estas organizações são empresas que distribuem filiais ou efetuam offshore com o objetivo de aumentar a produtividade, reduzir o custo e tornar a distribuição do software mais compatível com características locais.

Entretanto dividir a produção de um software entre diversos nós envolve uma série de problemas. Alguns problemas que localmente são insignificantes ganham proporção quando uma tarefa é dividida entre organizações fisicamente distantes.

Segundo Prikladnicki (2002), o desenvolvimento distribuído de software é caracterizado quando um dos atores no envolvidos projeto (*stakeholders*) estiverem fisicamente distante dos demais. Prikladnicki também classifica a dispersão geográfica em três escalas: nacional, continental e global.

A variação da dispersão física influencia fortemente a possibilidade de reuniões presenciais e principalmente o horário de trabalho das equipes. Quando a dispersão física for nacional ou regional, há uma possibilidade de reunião presencial ou videoconferência mais eficaz. Este fato ocorre, pois não há uma diferença no idioma dos *stakeholders*, além disso, o fuso horário é praticamente o mesmo para todos os desenvolvedores A distância é a principal característica do DDS, entretanto, quanto global, pode agregar ao projeto diferenças culturais entre as equipes de desenvolvimento. Conseqüentemente o risco do projeto tende a aumentar.

O DDS é uma das conseqüências da globalização. Nesta era, as empresas podem organizar-se de maneira distribuída e conseqüentemente podem também estudar melhor os meios para otimizar a produção distribuída de software. Desta forma, um projeto de DDS pode ser resultado de uma distribuição de tarefas em um mesmo país/região ou até mesmo entre países diferentes, caracterizando o desenvolvimento global de software (Carmel, 2001).

Segundo Cataldo (2008), a habilidade de uma organização executar prosperamente suas tarefas depende da combinação apropriada da estrutura organizacional, processos, comunicação e mecanismos de coordenação. A visão mais generalista de Burton (1998) afirma que uma abordagem de coordenação é examinar a relação entre estrutura organizacional, processos e mecanismos de coordenação. A literatura de teoria organizacional sugere que a habilidade de uma organização realizar suas tarefas prosperamente depende da combinação apropriada da estrutura organizacional, processos, comunicação e mecanismos de coordenação.

A coordenação depende da forma na qual uma organização está estruturada. Compreender as maneiras nas quais os elementos da organização estão relacionados, criando o design organizacional, pode revelar os mecanismos de coordenação.

O design organizacional estratégico identifica segundo Burton (1998), o que a organização faz na estrutura organizacional e não nas tarefas, e que as tarefas específicas dependem de estrutura organizacional. O design organizacional estratégico especifica as unidades de agrupamentos organizacionais. Agregados a este design, informações como a relação entre as unidades organizacionais, controle, sistemas de incentivo e fluxo de informações, são identificadas e definidas.

Em um ambiente de Desenvolvimento Distribuído de software, o mecanismo de gerenciamento é a ferramenta que coordena a produção. Há diversas abordagens sobre este assunto. Em uma visão mais gerencial, autores como Mintzberg(2003) e Burton(1998) enfatizam o design organizacional como a principal estrutura de controle das organizações. Entretanto, autores como Cataldo(2007), Borden(2007) e Sinha (2007) enfatizam as ferramentas técnicas como mecanismos de coordenação.

2.1 O modelo de dinâmica M3DS

A dinâmica de um ambiente de DDS revela o funcionamento da rede, da sua concepção inicial até seu encerramento. Porém, abordar individualmente os sites na rede pode não revelar a influência que o projeto tem sobre sua existência na rede. Portanto, a dinâmica da rede, representada na figura 1, aborda conjuntamente dois componentes fundamentais em um ambiente de DDS. O primeiro componente indica a dinâmica de uma unidade de produção de software. O segundo elemento é atrelado ao projeto que é desenvolvido em uma rede de produção de software. Sobre esta dinâmica, há uma intersecção que ocorre quando o site precisa desenvolver um subproduto de rede, e há uma instância do projeto para vários sites.

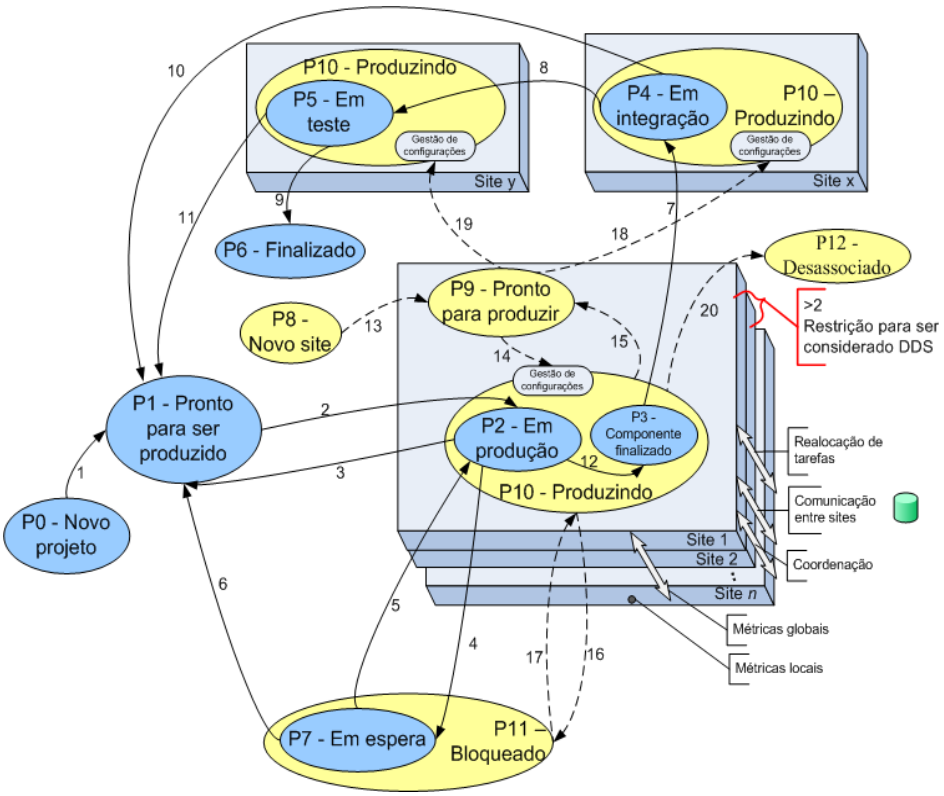

Figura 1 - M3DS

2.1.1 Descrição do modelo M3DS

Esta seção explica brevemente todos os estados do modelo e identificam quais estados existentes e quais condições necessárias são necessárias para a mudança de estado.

P0: Novo Projeto

Este estado indica a concepção de um novo produto de software por alguma organização, seja ela em rede ou não. Nesta fase, além da concepção do projeto há a elaboração de uma visão compartilhada para que todos os nós envolvidos na produção possam compreender a razão do produto final.

P1: Pronto para ser produzido

Para se alcançar o estado P1 é necessário que seja disparada a seta 1 (ou transição T1 na rede de Petri). Este disparo Indica que o plano gerencial do projeto em termos de componentização e divisão de tarefas foram realizados. Após esta seta, todos, ou parte dos artefatos estarão prontos para serem disponibilizados e distribuídos para que a rede e os *sites* possam desenvolvê-los. Este estado indica que um papel de distribuidor de tarefas entra em ação para gerenciar a distribuição das tarefas do projeto na rede. Representa a segmentação e alocação das tarefas nos diversos sites de produção. É neste estado que inicia-se do processo de ruptura.

P2: Em produção

Este estado representa uma tarefa que tem como resultado um subproduto de trabalho. Para almejar o estado P2, a seta 2 precisa ser disparada. Esta seta indica que o subproduto foi alocado em produção. Esta etapa implica em enviar para o site a tarefa e os dados sobre ela. Podem ser enviados, por exemplo, códigos mais atualizados, scripts de banco de dados, requisitos do software, etc.

P3: Componente finalizado

O componente é finalizado quando sua construção é concluída. Neste caso, indica o fim do processo de codificação de um componente ou módulo de software. Este estado sinaliza que o componente está apto a ser integrado com outro. As operações de *check-out* são efetuadas após a conclusão do componente. A gestão de configurações está presente explicitamente nesta etapa.

P4: Em integração

Após a codificação de dois ou mais componentes, o processo de integração é iniciado. Neste caso apenas um site é responsável em integrar dois ou mais componente, gerando outro produto de trabalho. A seta 7 indica o inicio deste estado.

P5: Em teste

Quando a seta 8 é disparada inicia-se o procedimento de teste. Neste caso o teste é sobre a integração do software. Após esta etapa, o projeto pode ser finalizado, mudando seu estado para P6, ou o teste pode gerar outro produto de trabalho, realimentando o ciclo por meio do estado P1.

P6: Finalizado

O projeto é concluído. Este estado representa que o projeto foi totalmente montado e concluído. Para alcançar este estado, a seta 9 precisa ser disparada

P7: Em espera

As dependências funcionais entre artefatos gerados pelos sites podem ocasionar em bloqueio em uma determinada tarefa. Este estado representa o bloqueio da tarefa em relação a um projeto.As setas 4 e 16 ocorrerem paralelamente quando há uma dependência de produtos de trabalho. Se isso ocorrer, para o projeto, o subproduto fica bloqueado. Este estado ocorre juntamente com o estado P11.

P8: Novo site

Indica que um novo nó da rede está pronto para associar-se a uma rede de produção de software. Para ser considerado um novo site, um mecanismo de coordenação, agrupamento e

processo de software devem ser estipulados. Este estado indica que o site está apto a trabalhar em equipe com os demais sites.

Um site pode ser criado a partir de um modelo já existente, importando processos e modelos de desenvolvimento. Os procedimentos de replicação são detalhados por Fabri (2004).

P9: Pronto para produzir

Para que o estado P9 seja alcançado é necessário que a transação indicada pela seta 13 seja disparada. A seta 13 representa os procedimentos iniciais de produção.

P10: Produzindo

Este estado representa que o site está executando uma tarefa e retornará como resultado um subproduto. Pode ser alcançado após o disparo da seta 14. Neste caso, a produção inicia-se, ou seja, as atividades referentes ao desenvolvimento do produto que foram atribuídas para o site são executadas. Esta etapa ocorre paralelamente a seta 2.

P11: Bloqueado

Podem ocorrer bloqueios na produção por dependências funcionais. Por exemplo, um site precisa de um produto de trabalho de outro site para prosseguir. Este estado representa que o site está bloqueado para uma determinada tarefa.Para tanto, as transições 4 e 16 ocorrem quando há uma dependência de produtos de trabalho. Se isso ocorrer, para o projeto, o subproduto fica bloqueado, pois, não pode ser desenvolvido pelo fato de sofrer dependência funcional.

P12: Desassociado

A ação indicada pela seta 20 indica a desassociação do *site* à rede que é removido da mesma e deixa de fazer parte do ambiente de DDS. Esta transição ocorre quando o site deixa de atuar neste projeto/grupo. Há uma desassociação do site com relação aos demais sites e projeto.

3. Custeio ABC

Segundo Araujo (2005), atualmente existe um grande movimento mundial de capitais e de recursos produtivos que procuram localizações mais favoráveis para atingir seus objetivos corporativos. Estas alterações são motivadas por uma lógica competitiva, em que as empresas podem decidir sobre a re-orientar capitais, equipamentos e pessoas de todo o mundo, na sua busca de novos mercados. As informações sobre a rentabilidade de produtos, processos e canais de comércio em diferentes países, e mesmo em operações entre eles, são indispensáveis para o projeto e para analisar a viabilidade de novos negócios. A adequada estruturação dos processos e produtos de engenharia nos fabricantes favorece uma avaliação dos custos de produção considerando não apenas os aspectos econômico-financeiros, mas também operacionais tecnológicos e institucionais existentes.

O ABC surgiu segundo Magalhães (2007), em meados da década de 1980, quando muitas organizações notaram que os sistemas de custos tradicionais geravam informações inexatas referente ao custo de produção dos produtos. Este método, segundo o mesmo autor, tem uma capacidade maior de gerenciabilidade de custos indiretos na produção de um determinado produto ou prestação de um determinado serviço. Para tanto, desmarca as distorções causadas pelo *overhead* gerado a partir dos custos indiretos.

Segundo Sutton (1991), o método de custeio por atividades (ABC – *Activity-based costing*) agrega os custos das atividades que foram executadas para a construção do produto. O método assume que as atividades consomem recursos e que os produtos consomem atividades. Desta maneira, custos indiretos de produção são agregados e calculados de

maneira mais eficaz quando o objetivo é gestão. O resultado obtido pelo ABC tem o foco voltado para a estratégia, a reestruturação da manufatura e nem sempre pode ser empregado como mecanismo de controle financeiro, por razões legais.

Magalhães (2007) afirma que para a aferição do método ABC, é necessário o levantamento dos processos organizacionais. As atividades organizacionais são utilizadas como elemento chave para análise do comportamento do custo, associando a estes despesas organizacionais.

Para o uso do ABC, segundo Sutton (2001), é necessária a padronização de direcionadores de custo que possam ser mensurados, por exemplo, em horas, unidades, etc. Estes direcionadores de custo (*Costing Drivers*) são utilizados nas atividades para medir o custo de um determinado produto. Diferente dos métodos tradicionais em que os valores dos custos indiretos de fabricação (CIF), não são inseridos no final da contabilidade, o ABC divide os CIF´s entre as atividades, associando-os indiretamente com a mão de obra direta (MOD) utilizada em cada atividade.

O método ABC é ilustrado pela figura 1. Para a construção do produto, é necessária a execução de várias atividades. Para cada atividade, ilustrada na figura 1, é associada o custo de mão de obra direta (MOD), materiais diretos e o custo indireto de fabricação (CIF). Cada atividade pode ser analisada em dois aspectos. O primeiro aspecto é a visão econômica e de custeio, que indica o custo financeiro da atividade, associando os recursos a esta. O segundo aspecto está relacionado ao aperfeiçoamento do processo. Neste segundo caso, os direcionadores de custo são definidos e o objetivo é medir o desempenho da atividade.

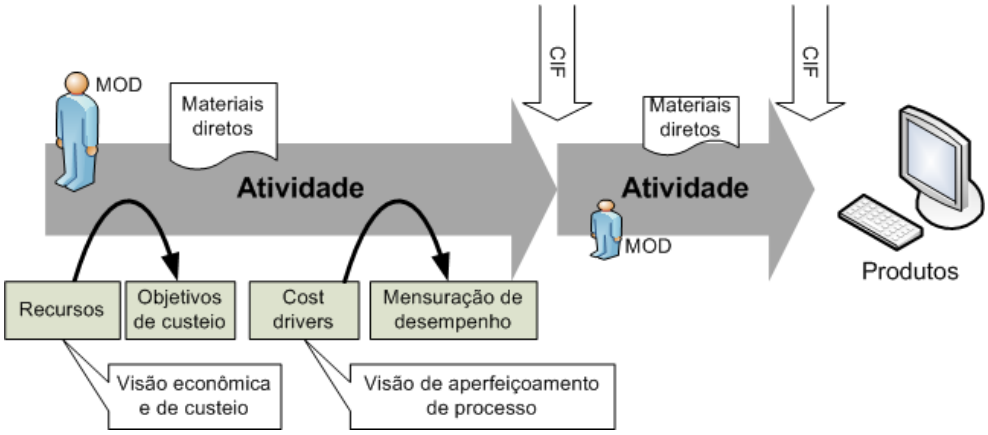

Figura 2 - Custeio Baseado em Atividades

4. O método ABC para o custeio do DDS

O método de custeio ABC, segundo Han-Wen(2006) é empregado comumente quando o objetivo é aperfeiçoar a produção. Segundo este mesmo autor a combinação deste método de custeio combinado com outros modelos como o CMMI (para garantir a qualidade de software), o BSC (para mensurar o desempenho da organização), pode aumentar a eficácia organizacional e conduzir os gerentes na melhoria de processos.

4.1 Atividades do DDS

Segundo Sutton (2001) e Iltuzer (2007) a primeira fase na implantação do método de custeio ABC é determinar quais atividades essenciais do sistema. Neste trabalho o M3DS serviu de parâmetro para que algumas atividades do DDS fossem levantadas. A segunda etapa

consiste em atribuir custo para cada uma das atividades levantadas e a próxima é a identificação dos direcionadores de custo.

A partir da figura 1, foi possível enumerar uma série de conjuntos de atividades que são efetuadas no desenvolvimento distribuído de software. Este conjunto de atividades está ilustrado pela figura 3. O M3DS é multidimensional, por esta razão, foi elaborado um modelo mais simples (indicado na figura 3) cujo objetivo é indicar somente as atividades que os desenvolvedores do projeto executam. Para cada conjunto de atividades, uma tabela indicando quais as sub-atividades, quais os insumos de entrada e saída e qual a unidade de medida foi elaborada.

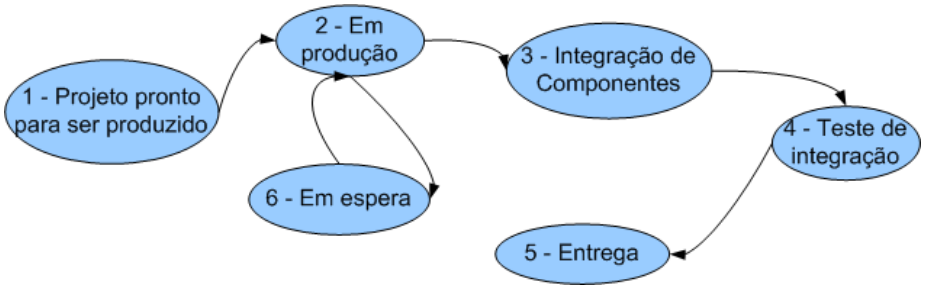

Figura 3 - Conjunto de atividades essenciais do DDS

A soma de todos os custos, referente a figura 3 representa o custo do desenvolvimento distribuído. Desta maneira tem-se:

$$C = Cr + \sum_{i=1}^{n} Ci + Cg + Ct + Ce + To$$

Onde:

$C = custo\ final$

$Cr = custo\ da\ ruptura$

$Ci = custo\ individual\ do\ desenvolvimento\ no\ site$

$Cg = custo\ da\ integração$

$Ct = custo\ do\ teste$

$Ce = custo\ da\ entrega$

$To = custo\ do\ tempo\ ocioso$

A tabela 1, tabela 2, tabela 3, tabela 4, tabela 5 e

tabela 6 representam a atividades que são realizadas em cada um das macro-atividades indicados na figura 3. Estas atividades são consideradas críticas para o DDS. Demais atividades do desenvolvimento de software como levantamento de requisitos e implantação não são escopos deste estudo.

Tabela 1 – Macro-atividade 1: projeto pronto para ser produzido

ID	Atividades	Entrada	Saída	Unidade de medida
1.1	Elaborar solução do projeto	Requisitos	Arquitetura do projeto	Projeto
1.2	Efetuar ruptura do projeto	Arquitetura	Arquitetura e Componentes	Pontos por função
1.3	Criar mecanismo de controle de versões (gestão de configuração)	Componentes, documentação	Atualizar/Criar novos dados nos servidores de controle de versão	Bits

Tabela 2 – Macro-atividade 2: em produção

ID	Atividades	Entrada	Saída	Unidade de medida
2.1	Efetuar atualização a	Arquitetura,	Atualização dos computadores de	Bits (Tráfego na rede)

	partir dos servidores de controle de versão	documentação de uso do controle de versões	trabalho (estação de trabalho na qual os desenvolvedores utilizam para codificar)	
2.2	Codificação	Especificação dos Requisitos, dados do sistema de controle de versões	Componentes de software	Pontos por função
2.3	Efetuar atualização dos servidores de controle de versão	Componentes de software	Atualizar/Criar novos dados nos servidores de controle de versão	Bits

Tabela 3 – Macro-atividade 3: Integração de componentes

ID	Atividades	Entrada	Saída	Direcionador de custo
3.1	Efetuar atualização a partir dos servidores de controle de versão	Arquitetura, documentação de uso do controle de versões	Atualização dos computadores de trabalho (estação de trabalho na qual os desenvolvedores utilizam para codificar)	Bits (soma total ou parcial do tráfego gerado pela atividade 2)
3.2	Integração	Especificação dos Requisitos, dados do sistema de controle de versões	Componentes de software	Pontos por função
3.3	Efetuar atualização dos servidores de controle de versão	Componentes de software	Atualizar/Criar novos dados nos servidores de controle de versão	Bits

Tabela 4 – Macro-atividade 4: Teste de integração

ID	Atividades	Entrada	Saída	Direcionador de custo
4.1	Efetuar atualização a partir dos servidores de controle de versão	Arquitetura, documentação de uso do controle de versões	Atualização dos computadores de trabalho (estação de trabalho na qual os desenvolvedores utilizam para codificar)	Bits (soma total ou parcial do tráfego gerado pela atividade 2)
4.2	Teste	Especificação dos Requisitos, dados do sistema de controle de versões	Componentes de software	Pontos por função
4.3	Efetuar atualização dos servidores de controle de versão	Componentes de software	Atualizar/Criar novos dados nos servidores de controle de versão	Bits

Tabela 5 – Macro-atividade 5: entrega

ID	Atividades	Entrada	Saída	Direcionador de custo
5.1	Efetuar atualização a partir dos servidores de controle de versão	Arquitetura, documentação de uso do controle de versões	Atualização dos computadores de trabalho (estação de trabalho na qual os desenvolvedores utilizam para codificar)	Bits (soma total ou parcial do tráfego gerado pela atividade 2)
5.2	Efetuar procedimento de deploy (compilação final do software)	Versão *branch* do software	Software funcional e pronto para ser instalado	Pontos por função
5.3	Efetuar atualização dos servidores de controle de versão	Componentes de software	*Branch* de versão do software	Bits

Tabela 6 – Macro-atividade 6: em espera

ID	Atividades	Entrada	Saída	Direcionador de custo
6.1	Aguardar por componente	Componente de software	Retomada das atividades	Tempo

Vários direcionadores custo foram identificados em cada atividade. Basicamente foram três tipos: bits transmitidos/armazenados, pontos por função que é uma teoria apresentada por Matson (1994) e tempo. Para cada inicio/fim de atividade, há um custo na

transmissão e armazenamento dos dados gerados. Caso ocorra algum impasse no desenvolvimento há um custo gerado em função de algum desenvolvedor ocioso. O ponto por função representa a métrica do software, indicando qual o tamanho do projeto. Durante todo o projeto, um sistema de controle de versões deve manter-se ativo, consumindo uma estrutura de armazenamento e rede (internet).

O custo por bit transmitido e armazenado deve ser calculado de acordo com o custo local de cada site de acesso a internet. O servidor de configurações, geralmente é uma estrutura centralizada e os demais sites de desenvolvimento transmitem/armazenam seus resultados neste.

5. Exemplo

Este seção demonstra, por meio de um exemplo fictício, como o método ABC pode ser utilizado como parâmetro de avaliação de viabilidade para produção distribuída.

Uma determinada organização de desenvolvimento global de software pretende avaliar o custo da produção distribuída e verificar se é viável a distribuição da produção de um projeto entre seus sites. Esta organização possui uma matriz, que desenvolve, e também outros 4 sites, dispersos pelo globo terrestre. A matriz está com um novo projeto de software e pretende verificar qual o custo, caso ela distribua a produção pelos demais sites.

Uma análise inicial do novo projeto revelou que seu tamanho é de 1000 pontos por função (1000ppf). De acordo com a capacidade e conhecimento em determinados processos/tecnologias, a matriz conseguiu mapear uma alocação de tarefas entre os sites. Cada site tem uma determinada capacidade de produção por dia, já mensurada em projetos anteriores. Também há um custo por hora de cada site, que pode variar de acordo com a quantidade e qualificação de funcionários e também de acordo com o custo salarial onde o site está instalado.

A tabela 7 indica os valores relacionados à produção da organização. O custo e estimativas do projeto, que estão indicadas nas duas últimas colunas desta tabela. Todos os sites operam igualmente em um turno de 8 horas diárias. Nesta primeira previsão, os dados transmitidos e armazenados não foram considerados. Também não foram considerados os possíveis impasses que ocorrem em um desenvolvimento distribuído. O custo/hora inclui o uso dos computadores pelos desenvolvedores.

Tabela 7 - Relação de capacidade produtiva de site / alocação de tarefas do projeto

Unidade de produção	Capacidade de produção/dia	Alocação de tarefas	Custo/hora	Previsão dias	Previsão horas
Matriz	20 ppf	100 ppf	$ 100,00	5	40
Site A	10 ppf	200 ppf	$ 30,00	20	160
Site B	25 ppf	200 ppf	$ 70,00	8	64
Site C	30 ppf	400 ppf	$ 25,00	13	107
Site D	15 ppf	100 ppf	$ 50,00	7	53
TOTAL	**100 ppf**	**1000 ppf**	**$ 275,00**	**53**	**424**

O sistema de controle de versões é centralizado e está instalado na matriz. Por esta razão o custo dos bits para a matriz é o custo de armazenamento que envolve além de guardar o bit, restaurar, efetuar cópias de segurança, e demais atividades referentes a gerenciamento de servidores. Os demais sites são remotos e possuem o custo de transmissão de bits. Este custo depende o serviço de Internet instalado nos sites locais. A tabela 8 sintetiza o custo dos bits transmitidos e bits armazenados no projeto.

Tabela 8 - Custo de bits transmitidos e armazenados

Unidade de produção	Custo / bit	bits armazenados	bits transmitidos	Custo total
Matriz	$ 0,0590	350 Mb		$ 20,65
Site A	$ 0,0120		850 Mb	$ 10,20
Site B	$ 0,0890		890 Mb	$ 79,21
Site C	$ 0,2500		440 Mb	$ 110,00
Site D	$ 0,5500		680 Mb	$ 374,00
TOTAL	**$ 0,9600**	**350 Mb**	**2860 Mb**	**$ 594,06**

A figura 4 representa o roteiro de produção para o projeto exemplificado. As setas representam a transmissão dos produtos de trabalho. Há um impasse, na qual o Site C fica ocioso, representando custo de tempo de programação.

Estas atividades são exclusivamente deste projeto. Para cada projeto, a organização deve elaborar um diagrama de roteiro, tentando identificar previamente os impasses e minimizando-os. Para cada conjunto de tarefas executados por cada site, além do custo do software, deve ser adicionado o custo ocasionado pela distância. Neste caso, é tratado apenas o custo ocasionado pela distância.

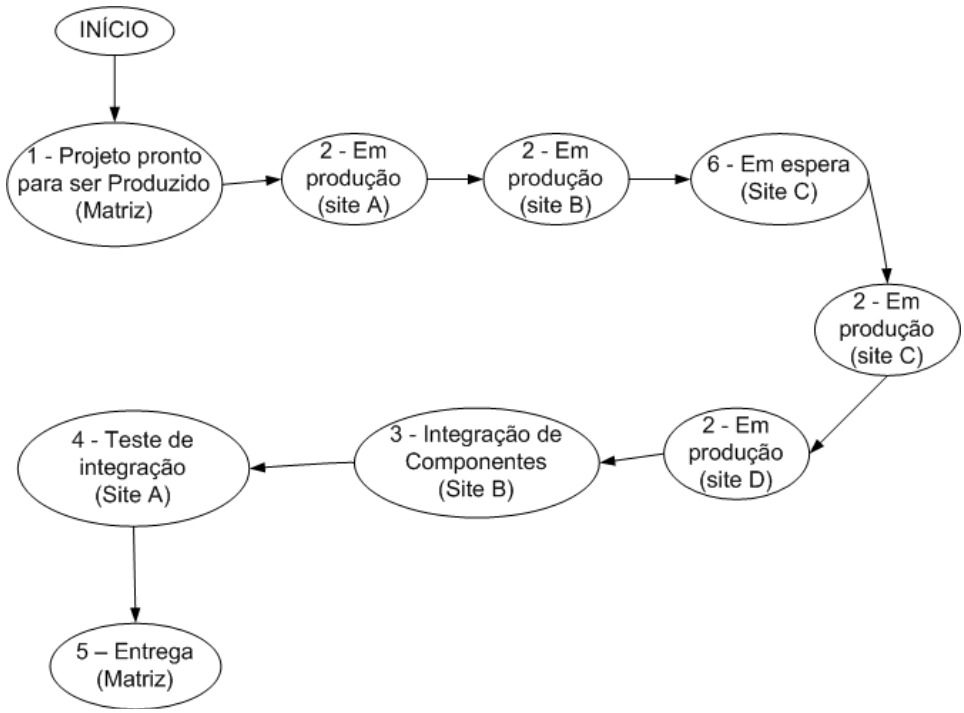

Figura 4 - Roteiro de produção

A figura 4 quando relacionada com custos, resulta em uma tabela identificando a atividade e o custo de cada atividade. Este resultado, justificando inclusive os valores apresentados na tabela 8 é exibido pela tabela 9.

Tabela 9 - Relação de custos por atividade

Local	ID	Atividades	Unidade de medida	custo
Matriz	1.1	Elaborar solução do projeto	Projeto	1
	1.2	Efetuar ruptura do projeto	Pontos por função	70 ppf
	1.3	Criar mecanismo de controle de versões (gestão de configuração)	Bits armazenados	50 Mb
		Mão de obra do arquiteteto de software	Horas	28
Site A	2.1	Efetuar atualização a partir dos servidores de controle de versão	Bits (Tráfego na rede)	50 Mb
	2.2	Codificação	Pontos por função	200 ppf
	2.3	Efetuar atualização dos servidores de controle de versão	Bits	100 Mb
		Mão de obra (desenvolvedor a custo local)	Horas	160
Site B	2.1	Efetuar atualização a partir dos servidores de controle de versão	Bits (Tráfego na rede)	100 Mb
	2.2	Codificação	Pontos por função	200 ppf
	2.3	Efetuar atualização dos servidores de controle de versão	Bits	130 Mb
		Mão de obra (desenvolvedor a custo local)	Horas	64
Site C	2.1	Efetuar atualização a partir dos servidores de controle de versão	Bits (Tráfego na rede)	130 Mb
	2.2	Codificação	Pontos por função	400 ppf
	2.3	Efetuar atualização dos servidores de controle de versão	Bits	310 Mb
		Mão de obra (desenvolvedor a custo local)	Horas	107
Site D	2.1	Efetuar atualização a partir dos servidores de controle de versão	Bits (Tráfego na rede)	310 Mb
	2.2	Codificação	Pontos por função	100 ppf
	2.3	Efetuar atualização dos servidores de controle de versão	Bits	370 Mb
		Mão de obra (desenvolvedor a custo local)	Horas	53
Site B	3.1	Efetuar atualização a partir dos servidores de controle de versão	Bits (soma total ou parcial do tráfego gerado pela atividade 2)	310 Mb
	3.2	Integração	Pontos por função	30 ppf
	3.3	Efetuar atualização dos servidores de controle de versão	Bits	350 Mb
		Mão de obra (desenvolvedor a custo local)	Horas	9,6
Site A	4.1	Efetuar atualização a partir dos servidores de controle de versão	Bits (soma total ou parcial do tráfego gerado pela atividade 2)	350 Mb
	4.2	Teste	Horas	10
	4.3	Efetuar atualização dos servidores de controle de versão	Bits	350 Mb
Matriz	5.1	Efetuar atualização a partir dos servidores de controle de versão	Bits (soma total ou parcial do tráfego gerado pela atividade 2)	350 Mb
	5.2	Efetuar procedimento de deploy (compilação final do software)	Pontos por função	350 Mb
	5.3	Efetuar atualização dos servidores de controle de	Bits	350 Mb

		versão		
Site C	6.1	Aguardar por componente	Horas	16

Totalizando o cálculo e agregando os valores, a relação custo do DDS, para este projeto, utilizando este conjunto de alocação de atividades é representado pela tabela 10.

Tabela 10 - Valor total do DDS

Elemento de custo	Valor total
Total em $	$ 18.785,33
Total de Bits armazenados	350 Mb
Total de Bits transmitidos	2869 Mb
Total em Horas	448

6. Conclusões

Este artigo apresentou um método para calcular o custo do desenvolvimento distribuído de software. Embora um direcionador de custo adotado neste trabalho foi o ponto por função que é uma métrica adotada por muitas empresas para estimar o software, não fez parte do escopo deste trabalho o uso do ponto por função na métrica do software.

O método apresentado neste artigo possibilita os gestores de produção em um ambiente de desenvolvimento distribuído de software a decidirem quanto a viabilidade de se distribuir a produção.

A viabilidade da produção distribuída está relacionada com o custo e também com outros fatores indicados por L'Erario (2007). Como por exemplo, organizações efetuam operações de *nearshore* com o objetivo de atingir um determinado mercado local.

Uma limitação deste trabalho é o fato do cálculo de custo efetuado neste trabalho considerar que todos os sites estão prontos para produzir. Esta configuração nem sempre está presente, pois, toda vez que uma organização ingressa em um modelo distribuído de produção de software, recebem um treinamento de ambientação deste sistema produtivo.

Como trabalho futuro, a aderência deste método pode ser validada nas organizações. Embora a extração das atividades fosse a partir de um modelo já validado (M3DS), seria necessária a validação em campo, pois, muitas variáveis que possivelmente surgiriam em um estudo de caso podem agregar a este método, dando-o mais aplicabilidade.

Referências

ARAUJO, J. A. R. . **Operations strategy and cost management**. Revista da Gestão da Tecnologia e Sistemas de Informação - Journal of Informations Systems and Tecnology Managment, São Paulo, v. 02, n. 3, p. 291-303, 2005.

BORDEN, Alexander, NETT, Bernhard e WULF Volker. **Coordination Practices in Distributed Software Development of Small Enterprises**, icgse,pp.235-246, International Conference on Global Software Engineering (ICGSE 2007), 2007

BURTON R.M. e OBEL B. **Strategic Organizational Diagnosis and Design**. Kluwer Academic Publishers, 1998.

CARMEL, Erran; AGARWAL, Ritu. **Tactical Approaches for Alleviating Distance in Global Software Development**. IEEE Software, v. 18, n. 2. março/abril 2001. p. 22-29.

CATALDO, M., BASS, M., HERBSLEB, J. D., and Bass, L. 2007. **On Coordination Mechanisms in Global Software Development.** In Proceedings of the international Conference on Global Software Engineering, ICGSE. IEEE Computer Society, 2007

FABRI, José ; TRINDADE, André L P ; BEGOSSO, Luis R ; A. L'ERARIO, Alexandre ; SILVEIRA, Fábio L F ; PESSOA, Marcelo S P . **Techniques for the Development of a Software Factory: case CEPEIN-FEMA**. In: 17th International Conference Software & Systems Engineering an their Applications, 2004, Paris. Annais of 17th International Conference Software & Systems Engineering an their Applications, 2004. v. 3.

HAN-WEN Tuan; CHIA-YI Liu; CHIOU-MEI Chen, **Using ABC Model for Software Process Improvement: A Balanced Perspective**, System Sciences, 2006. HICSS '06. Proceedings of the 39th Annual Hawaii International Conference on , vol.9, no., pp. 229c-229c, 04-07 Jan. 2006

HERBSLEB, James D.; MOCKUS, Audris.. **An Empirical Study of Speed and Communication in Globally-Distributed Software Development**. IEEE Transactions on Software Engineering, v. 29, n. 3, p.1-14, 2003.

ILTUZER, Z.; TAS, O., **Implementation of Activity-Based Costing in e-Businesses**, Management of Engineering and Technology, Portland International Center for , vol., no., pp.1119-1125, 5-9 Aug. 2007

L'ERARIO, Alexandre et al. **Desenvolvimento distribuído de software para sistemas pervasivos: um estudo de caso**. In: SIMPÓSIO BRASILEIRO DE SISTEMA DE INFORMAÇÃO, 1., 2004, Porto Alegre. Simpósio. Porto Alegre: SBC, 2004. p. 163 - 170.

L'ERARIO, Alexandre, PESSÔA, Marcelo Schneck de Paula. **An Analysis of the Dynamics and Properties of the Distributed Development of Software Environments: A Case Study**. Software Engineering Research and Practice 2007: 471-477

MAGALHAES, Ivan Luizio; PINHEIRO, Walfrido Brito. **Gerenciamento De Serviços De Ti Na Pratica: Uma Abordagem Com Base Na Itil**. São Paulo: Novatec, 2007.

MATSON, J.E.; BARRETT, B.E.; MELLICHAMP, J.M., **Software development cost estimation using function points**, Software Engineering, IEEE Transactions on , vol.20, no.4, pp.275-287, Apr 1994

MINTZBERG, Henry. **Criando organizações eficazes: Estruturas em cinco configurações**. 2. ed. São Paulo: Atlas, 2003. 334 p.

PRIKLADNICKI, Rafael; AUDY, Jorge Luis Nicolas. **Towards a Model of Software Development Process for a Physically Distributed Environment - Minimizing communication difficulties and adding planning and evaluation view to the software development life cycle**. In: CACIC 2002 - VIII CONGRESSO ARGENTINO DE CIENCIAS DE LA COMPUTACION, 2002, Buenos Aires. Anales del VIII Congresso Argentino de Ciencias de la Computacion. 2002. v. I, p. 798-809.

SINHA, V.S.; SENGUPTA, B.; GHOSAL, S., **An Adaptive Tool Integration Framework to Enable Coordination in Distributed Software Development**, Global Software Engineering, 2007. ICGSE 2007. Second IEEE International Conference on , vol., no., pp.151-155, 27-30 Aug. 2007

SUTTON, Sharon, **A new age of accounting, Production and Inventory Management**. Journal, 32(1), First quarter 1991, pp. 72-74