

## **UMA ABORDAGEM SISTÊMICA AO MAPEAMENTO E MELHORIA DO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE**

A SYSTEMIC APPROACH TO MAP AND IMPROVE THE SOFTWARE DEVELOPMENT PROCESS

Omar Sacilotto **DONAIRES**  
Engenheiro Eletrônico, MBA, PMP  
omarsd@smar.com.br

### **Resumo**

O processo de desenvolvimento de software de grande porte, especialmente quando envolve inovação, pode ser entendido com um sistema complexo que precisa apresentar a capacidade de auto-organização para se adaptar às mudanças constantes no seu ambiente. O artigo descreve a aplicação da Soft Systems Methodology (SSM) juntamente com o Rational Unified Process (RUP) para mapear e adaptar o processo de desenvolvimento de software num ambiente que apresenta mudanças frequentes. A SSM contribui para a discussão contínua do processo de desenvolvimento. Ela facilita a aprendizagem acerca do processo, promove o debate acerca dos problemas existentes e estimula a ação para melhoria contínua do processo. O RUP oferece uma notação intuitiva para capturar num modelo conceitual o resultado da aprendizagem. A combinação de ambos permite estabelecer um processo de desenvolvimento de software capaz de se auto-organizar para lidar com as mudanças num ambiente turbulento.

**Palavras-chave:** Melhoria de Processo – Pensamento Sistêmico – Desenvolvimento de Software.

### **Abstract**

The development process for large-scale software, particularly when innovation is an issue, can be understood as a complex system that needs to exhibit the ability of self-organization in order to adapt to the continuous changes in its environment. The paper describes the application of the Soft Systems Methodology (SSM) together with the Rational Unified Process (RUP) in order to map and adapt the software development process in an environment that presents frequent changes. The SSM contributes to the continuous discussion of the development process. It facilitates learning about the process, fosters the debate about the existing problems and encourages the action to continuously improve the process. The RUP offers an intuitive notation to help to capture in a conceptual model the result of the learning. The combination of both allows to establish a software development process that is able to self-organize itself in order to cope with changes in a turbulent environment.

**Keywords:** Process Improvement – Systemic Thinking – Software Development.

## Introdução

O desenvolvimento de *software* de grande porte, especialmente no caso em que a inovação é um elemento essencial do processo de desenvolvimento, requer a coordenação de inúmeras atividades pertencentes a várias disciplinas diferentes. Além disso, existe a necessidade de adaptação contínua a um ambiente em constante mudança, principalmente quando se emprega ou se desenvolve tecnologia de ponta. Para lidar com todos os aspectos do desenvolvimento e da inovação, o processo de *software* pode precisar ser bastante complexo (Booch, 1994). E, para lidar com as frequentes mudanças no ambiente de desenvolvimento o processo, é possível que precise de adaptação contínua.

Não existe uma solução pronta para o processo de *software* que possa ser aplicada diretamente nesses casos. A literatura da engenharia de *software* descreve alguns ciclos de vida para desenvolvimento de *software* (Boehm, 1988, 2000; Booch, 1996; Pressman, 1995); porém, são frequentemente descritos de forma idealizada, simplificada para fins didáticos. Consequentemente, é transmitido um senso de ordem que não existe na realidade quando se trata de sistemas complexos. Observa-se, então, que tais ciclos de vida precisam ser expandidos para fins de aplicação prática.

Outras soluções, na tentativa de cobrirem ao máximo todos os cenários do desenvolvimento de *software*, descrevem um grande número de elementos em detalhes (Kruchten, 2000; Rational Software Corporation, 2003). Essas soluções também não são direcionadas à aplicação direta, e requerem um esforço de configuração do processo para adequá-lo à cultura e às necessidades particulares da organização e da equipe de desenvolvimento.

Em qualquer dos casos existe o esforço de implantação ou a adequação das práticas de desenvolvimento, e esse esforço raramente é trivial.

Infelizmente, em função da ênfase positivista e cartesiana na educação, em geral, e na formação dos profissionais de engenharia e de *software*, em particular, existe a tendência da aplicação dos modelos de processo segundo uma concepção mecanicista, isto é, tenta-se aplicá-los tal como descritos na literatura. Em muitos casos acredita-se que seja possível sair do modelo artesanal de desenvolvimento para o modelo da fábrica, em que há execução rotineira de atividades repetitivas. O processo mecânico é atraente pela facilidade de previsão e controle que ele oferece. Em alguns casos especiais ele, é de fato possível. Porém, nos ambientes turbulentos anteriormente descritos, a busca ansiosa pelo controle mecanicista se torna uma armadilha. As pessoas podem ser levadas a pensar que a organização é caótica quando na verdade o processo mecânico repetitivo caracteriza-se como inadequado à realidade conturbada da organização.

Uma das alternativas à abordagem mecânica é a dos sistemas de atividade humana (Checkland, 1981). A característica distintiva dos sistemas de atividade humana está relacionada à autoconsciência do homem que tem como consequência a liberdade de escolha para selecionar suas ações. Essa característica é muito importante para o entendimento da natureza dos sistemas de atividade humana. A racionalização de sistemas de atividade humana justifica métodos especiais para lidar com as consequências sistêmicas das idiosincrasias do comportamento humano. Os problemas dos sistemas de atividade humana são, em geral, mal-estruturados. Um problema mal-estruturado é aquele em que a sua formulação é, ela própria, um problema.

A abordagem dos sistemas de atividade humana baseia-se em conceitos da Sociologia e propõe o debate para conciliar os pontos de vista divergentes, assim como a aprendizagem para lidar com as situações problemáticas. Ela é mais

realista e mais adequada para fugir do modelo artesanal do que a abordagem mecanicista quando as condições de desenvolvimento são variáveis e instáveis. As pessoas acostumadas com o processo mecânico repetitivo tendem a rejeitar esse tipo de abordagem à primeira vista porque ele parece mais desorganizado e mais difícil de controlar. A abordagem sociológica é, de fato, mais complexa. Essa complexidade adicional, porém, é benéfica e necessária. Ela permite acomodar a variedade inerente a um processo de desenvolvimento complexo. É isso que ensina a máxima da máxima da cibernética quando declara que “somente variedade absorve variedade” (DONAIRES, 2006a).

### **1 O processo de desenvolvimento de *software* como um sistema de atividade humana**

O processo de desenvolvimento segundo uma abordagem sociológica não pode ser preconcebido e projetado como uma máquina. Ele acontece à medida que as pessoas trabalham no processo. Ele pode ser entendido como um sistema de atividade humana, conforme proposto por Checkland (1981) na *Soft Systems Methodology* (SSM). Os sistemas de atividade humana explorados por Checkland na SSM são menos tangíveis que os sistemas naturais e projetados, mas são claramente observáveis. Conforme exemplifica o autor, esses sistemas são mais ou menos conscientes e organizados em torno de algum propósito ou missão subjacente, como um homem empunhando um martelo ou os sistemas políticos internacionais. Eles consistem em um número de atividade conectadas segundo algum princípio de coerência.

Os sistemas de atividade humana podem ser representados por meio de modelos conceituais. O modelo sistêmico nesse caso não tem a pretensão de ser um retrato exato da realidade, mas uma representação consensual das visões que os participantes têm do processo de

desenvolvimento. O modelo conceitual do sistema de atividade humana está no cerne da SSM, e captura o resultado da aprendizagem dos participantes sobre o processo de desenvolvimento de *software*.

É isso que caracteriza a natureza “*soft*” da metodologia, em contraste com o pensamento sistêmico “*hard*”. O pensamento sistêmico *hard* propõe que os sistemas sociais existem na prática e determinam o comportamento dos atores participantes do sistema. Eles podem ser rotulados, descritos e manipulados para otimizar seu desempenho em busca de eficiência. O pensamento sistêmico *soft*, ao contrário, não identifica o modelo sistêmico com a realidade. Ou seja, o sistema não é a realidade. O sistema é apenas uma representação da abstração mental da realidade segundo a interpretação dos atores participantes dessa mesma realidade. O pensamento sistêmico *soft*, então, não trata de um tipo específico de sistema, mas da forma como se interpreta qualquer modelo de sistema em relação à realidade que ele representa.

A consequência disso é que, na aplicação da SSM, pode-se usar como modelo conceitual qualquer modelo sistêmico que satisfaça as condições dos sistemas formais propostas por Checkland. É importante, porém, que se entenda que tal modelo não é a realidade, mas uma abstração que captura a visão dos participantes acerca da realidade.

### **2 O processo de desenvolvimento de *software* como um processo racional**

Conforme a visão geral descrita pela própria *Rational Software Corporation* (2003), o *Rational Unified Process®* (RUP®) é um processo de engenharia de *software*. Ele provê uma abordagem disciplinada para atribuir tarefas e responsabilidades dentro de uma organização de desenvolvimento. Seu objetivo

é garantir a produção de *software* de alta qualidade que satisfaça as necessidades dos seus usuários dentro de um cronograma e orçamento previsíveis.

Embora essa descrição pareça sugerir uma aplicação mecanicista do RUP, ele pode ser visto como um sistema de atividade humana, e aplicado segundo a perspectiva do pensamento sistêmico “*soft*”. De fato, o RUP é descrito por meio de fluxos de trabalho que mostram atores desempenhando papéis específicos à medida que executam as várias atividades necessárias ao desenvolvimento de *software*.

O RUP precisa ser customizado para o contexto específico de uma organização e um projeto de *software*. Conforme especificado pelo próprio RUP, é crucial para o sucesso do projeto que o processo de desenvolvimento seja relevante para o projeto em consideração, e para a dimensão e requisitos de formalidade do projeto. Processo demais tende a atrapalhar a criatividade e a eficiência. Processo de menos pode levar a um ambiente caótico, que tipicamente leva os membros individuais do projeto a tomar decisões locais que podem resultar em resultados ineficientes, inconsistentes e imprevisíveis.

No contexto descrito inicialmente, entretanto, essa customização não é um processo simples, tampouco estático. Ela requer a coordenação de inúmeros elementos e a adaptação contínua a mudanças. Nesse caso, a abordagem de aprendizagem da SSM se apresenta adequada para mapear e manter o modelo do processo de desenvolvimento.

As seções a seguir descrevem em maiores detalhes a SSM e o RUP. Descreve-se, em seguida, o um caso real de aplicação da SSM por uma equipe de desenvolvimento de uma empresa brasileira em que o RUP é o modelo conceitual que captura a aprendizagem acerca do processo de desenvolvimento de *software*. No caso descrito, o RUP é o foco do mapeamento,

melhoria e adaptação. Seguem-se então as considerações finais.

### 3 *Soft Systems Methodology* (SSM)

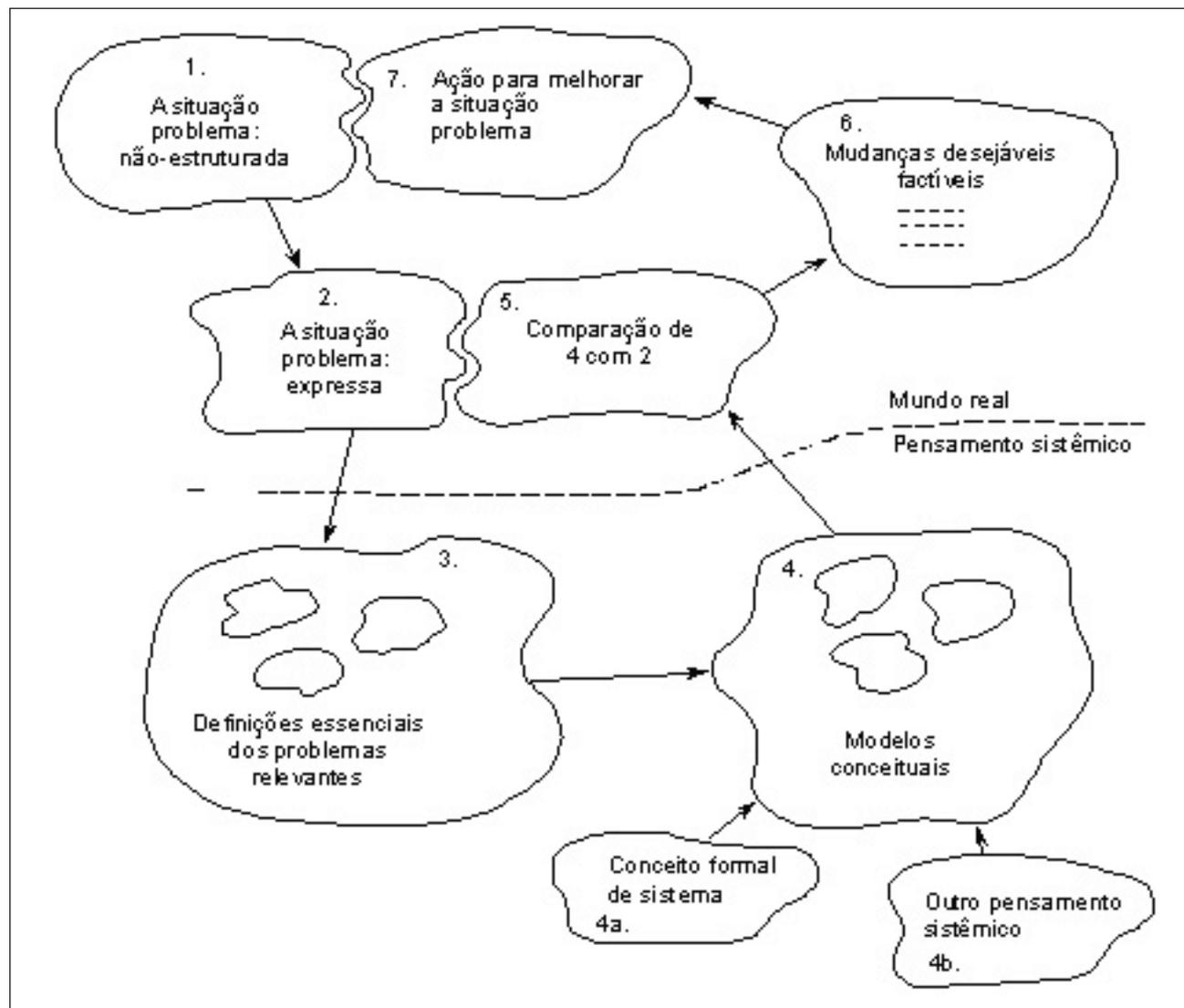
A SSM, representada na Figura 1, foi proposta por Peter Checkland (1981). Ela é um sistema de sete estágios de aprendizagem e intervenção na realidade. É um exemplo de pensamento sistêmico “*soft*”, fundamentado na sociologia interpretativista e na abordagem filosófica fenomenológica (DONAIRES, 2006a).

A ideia da metodologia é, dado um problema mal-estruturado típico de um sistema de atividade humana, promover um debate estruturado entre os participantes, estimulando-os a contribuir com suas respectivas visões do problema, até que se chegue a um consenso acerca de quais intervenções no sistema poderiam melhorá-lo, atenuando a condição problemática. Os sete estágios não precisam ser seguidos em sequência e, geralmente, o aprendizado ocorre iterativamente, pela execução do processo repetidamente.

O primeiro estágio trata da situação-problema não-estruturada. Os participantes do debate contribuem com suas respectivas *Weltanshauung*, isto é, suas visões do problema. O objetivo das contribuições individuais é construir um quadro tão rico quanto possível não do ‘problema’, mas da situação na qual se percebe o problema sem impor-lhe uma estrutura particular.

Segue-se o segundo estágio, que tem o propósito de expressar a situação-problema de maneira um pouco mais formal. Para tanto, procura-se identificar e registrar: (1) os elementos de estrutura (aquilo que muda lentamente); (2) os elementos de processo (aquilo que está continuamente mudando); (3) a relação entre estrutura e processo na situação objeto de investigação; e (4) as *Weltanshauungen* dos participantes.

Figura 1 – A SSM em resumo



Fonte: Checkland, 1981.

No terceiro estágio, são elaboradas as definições essenciais dos sistemas relevantes. As definições essenciais procuram capturar de forma concisa, numa única frase, a essência dos sistemas mais importantes relacionados à melhoria da situação-problema. Uma definição essencial bem formada deve tornar explícitos o cliente, os atores e o proprietário do sistema, o processo de transformação que ocorre, as *Weltanschauungen* e as restrições ambientais.

As definições essenciais dirigem a criação do modelo conceitual no quarto estágio da metodologia.

O modelo conceitual deve seguir as regras dos sistemas formais. Esse estágio admite o uso de outras ferramentas do pensamento sistêmico.

No quinto estágio, o modelo conceitual é comparado com a situação-problema expressa de forma estruturada e, a partir das diferenças encontradas, são sugeridas as mudanças necessárias para melhorar o sistema.

As mudanças levantadas no quinto estágio são avaliadas no sexto estágio e classificadas como desejáveis e factíveis.

O estágio sete é o da ação para melhorar a situação-problema pela implementação das mudanças que se revelaram desejáveis e factíveis.

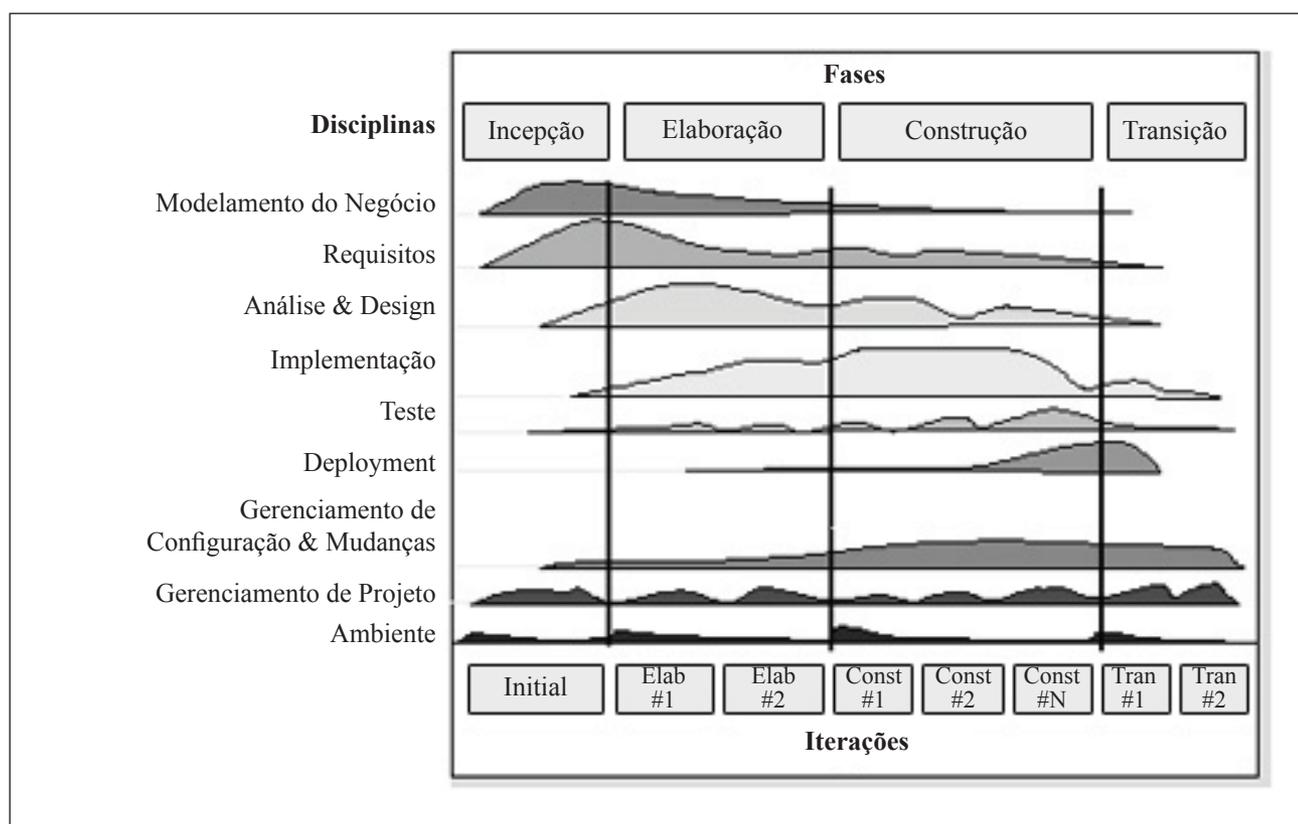
Após a intervenção realizada no estágio sete, espera-se uma melhoria na condição problema, mas não há garantias de que o sistema vá melhorar. Uma vez que se está lidando com problemas mal-estruturados de sistemas complexos, a situação-problema pode, ao invés, segundo a percepção dos participantes, piorar. A metodologia admite a possibilidade de fracasso numa intervenção e propõe que se repita todo o processo para construir um

modelo conceitual cada vez melhor do sistema de atividade humana que permita aprender sobre o sistema a cada intervenção e aumentar as chances de se realizar intervenções mais bem sucedidas a cada tentativa. A aprendizagem acerca da realidade se concretiza, então, à medida que esse processo se desdobra. O que se aprende sobre a realidade é capturado no modelo conceitual.

#### 4 *Rational Unified Process (RUP)*

A Figura 2 ilustra a estrutura geral do RUP em duas dimensões.

**Figura 2** – Estrutura do RUP – duas dimensões



Fonte: Adaptado de Kruchten (2000: p. 23)

O eixo horizontal representa o tempo e mostra os aspectos do ciclo de vida do processo à medida que ele se desenrola. Essa primeira dimensão ilustra os aspectos dinâmicos do processo e é expresso em termos de fases, iterações e marcos.

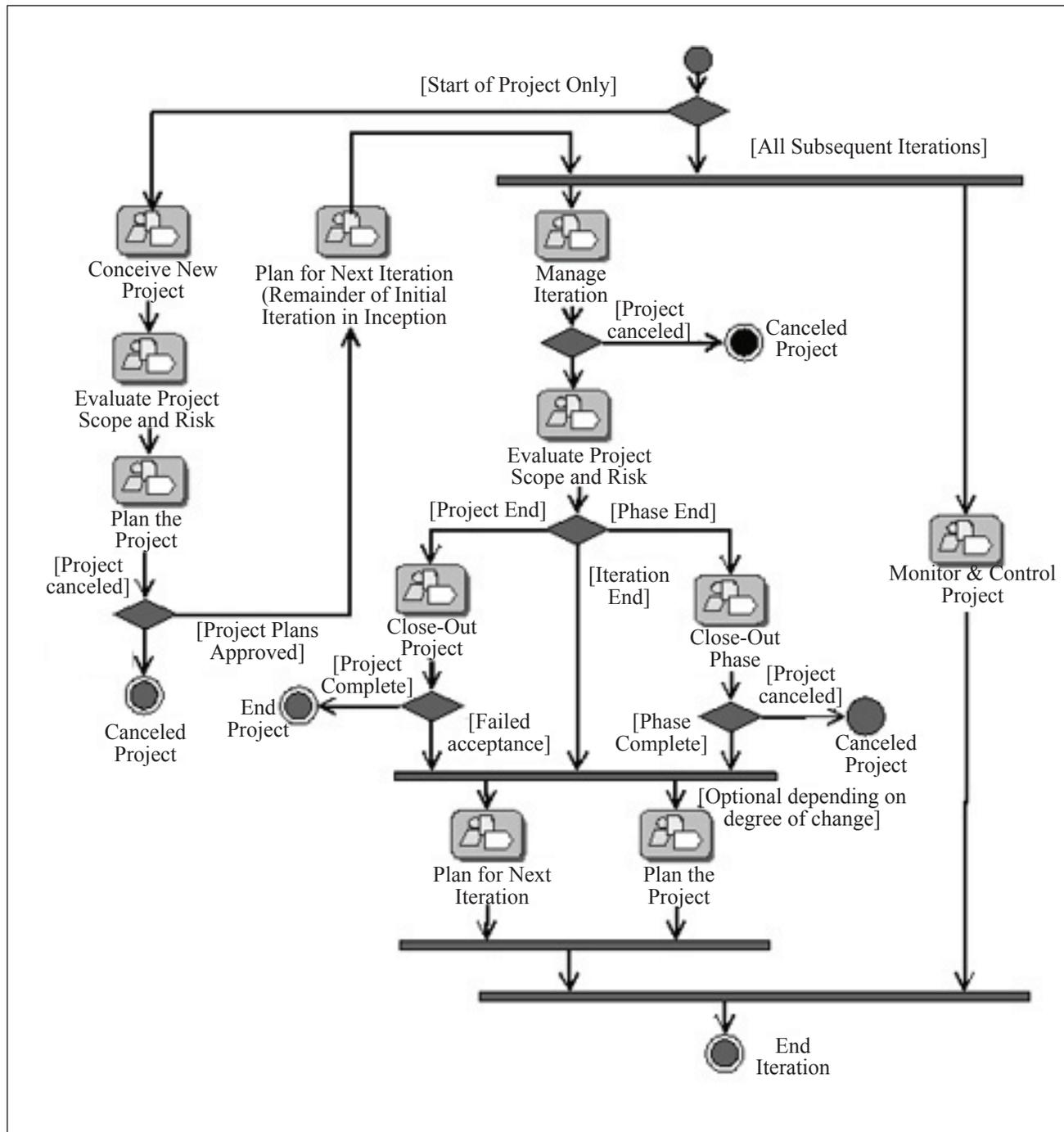
O eixo vertical representa disciplinas que agrupam logicamente as atividades por natureza. Essa segunda dimensão retrata os aspectos estáticos do processo que são descritos em termos de componentes de processo, disciplinas, atividades, fluxos de trabalho, artefatos e

papéis. Conforme revela o gráfico, a ênfase nas disciplinas varia com o tempo.

A cada uma das disciplinas corresponde um fluxo de trabalho de visão geral. Os fluxos

de trabalho de visão geral, conforme mostra o exemplo da Figura 3, identificam grupos de atividades, definem sua ordem relativa e mostram as decisões mais importantes.

**Figura 3** – Exemplo de fluxo de trabalho de visão geral: gerenciamento de projetos



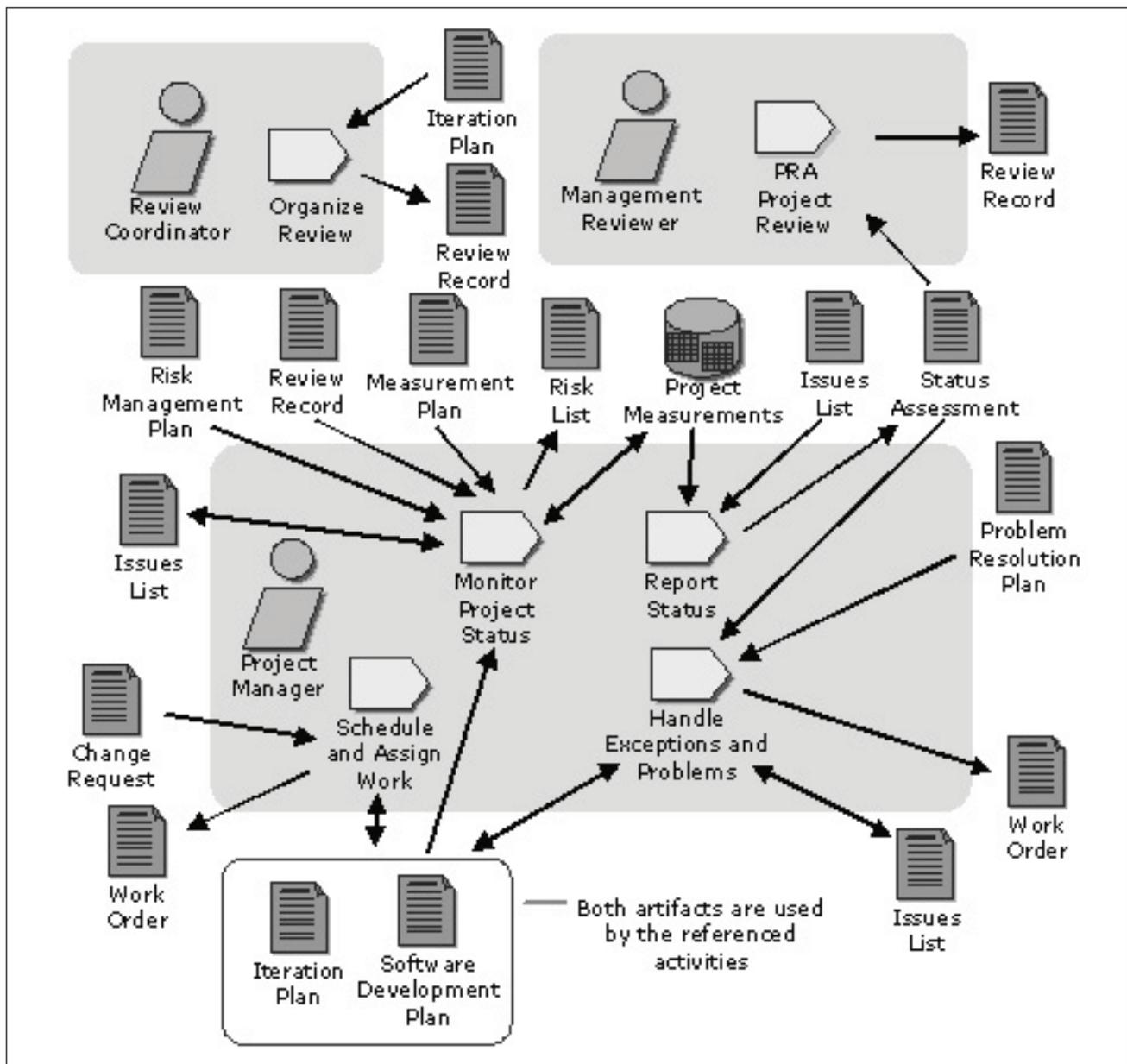
Fonte: Traduzido de RUP (2003: process\workflow\ovu\_mgm.htm)

A Figura 3 mostra o fluxo de trabalho de visão geral do gerenciamento do projeto de desenvolvimento de *software*. Ele mostra a ordem natural das atividades, tais como concepção inicial do projeto, avaliação do escopo e risco do projeto, planejamento, gerenciamento, até o cancelamento ou a conclusão do projeto. As barras horizontais representam pontos de sincronismo das atividades. O monitoramento e o controle do projeto acontecem o tempo todo. O diagrama mostra também decisões e eventos como a conclusão ou o cancelamento de

uma fase ou do projeto com um todo.

Cada uma das caixas no fluxo de trabalho de visão geral da Figura 3 representa um grupo de atividades, e são descritas através de fluxos de trabalho detalhados como os da Figura 4. Os fluxos de trabalho detalhados mostram como os atores, desempenhando papéis específicos, executam as várias atividades do processo e, como resultado do seu trabalho, produzem novos artefatos ou transformam artefatos de entrada em artefatos de saída.

**Figura 4** – Exemplo de fluxo de trabalho detalhado: monitorar & controlar projeto



Fonte: Traduzido de RUP (2003: process\workflow\managemen\wfs\_moncon.htm)

Um papel define o comportamento e responsabilidades de um indivíduo, ou de um grupo de indivíduos trabalhando em equipe, dentro do contexto de uma organização de engenharia de *software*. Papéis não são indivíduos; ao invés, descrevem como os indivíduos devem se comportar na ocupação e nas responsabilidades de cada um. Membros individuais da organização de desenvolvimento de *software* vestirão diferentes chapéus, ou desempenharão diferentes papéis, os quais têm atividades que definem o trabalho que eles realizam. Uma atividade é algo que um papel faz o qual provê um resultado significativo no contexto do projeto.

Uma atividade é uma unidade de trabalho que um indivíduo, desempenhando o papel descrito, pode ser solicitado a realizar. Ela tem um propósito claro, normalmente expresso em termos de criar ou atualizar alguns artefatos, tal como um modelo, uma classe, ou um plano, além de ser designada a um papel específico. A granularidade de uma atividade geralmente varia de algumas horas a alguns dias, normalmente envolve um papel, e afeta um ou somente um pequeno número de artefatos. Uma atividade deve ser utilizável como um elemento de planejamento e progresso; se for muito pequena, será negligenciada, e, se for muito grande, o progresso teria que ser expresso em termos de partes de uma atividade.

Atividades têm artefatos de entrada e saída. Um artefato é um produto de trabalho do processo: papéis usam artefatos para realizar atividades, e produzir artefatos ao realizar as atividades. Atividades estão sob a responsabilidade de um único papel, tornando as responsabilidades algo fácil de se identificar e entender, e promovendo a ideia de que toda informação produzida no processo requer o conjunto apropriado de habilidades. Muito embora um papel possa ser “dono” de um artefato, outros papéis o usarão, e até o atualizarão se o papel tiver permissão para tal.

A Figura 4 anterior mostra o fluxo detalhado

da monitoração e controle do projeto do desenvolvimento de software. Demonstra-se que o gerente de projetos é responsável por monitorar o andamento do projeto, reportar o status, tratar problemas e exceções, programar e distribuir o trabalho. O diagrama inclui as atividades de revisão e os papéis responsáveis por conduzi-las. Os artefatos utilizados e produzidos em cada atividade também são indicados no diagrama.

## 5 A metodologia combinada: SSM + RUP

### 5.1 Adequação do RUP à utilização como modelo conceitual na SSM

Conforme se pode inferir dos exemplos, a notação do RUP é bastante intuitiva e muito apropriada para se descrever sistemas de atividade humana para os propósitos de aplicação da SSM. Porém, conforme requer o estágio 4a da SSM, é preciso certificar-se de que o RUP – ou qualquer adaptação sua – satisfaça as condições do conceito formal de sistema. De fato, o RUP satisfaz cada uma delas conforme se analisa a seguir.

1. O RUP tem um propósito ou missão. Conforme a documentação do produto, esse propósito é garantir a produção de *software* de alta qualidade que satisfaça as necessidades dos seus usuários dentro de um cronograma e orçamento previsíveis. Ele é legítimo como propósito geral de qualquer esforço de desenvolvimento de software, e pode ainda ser ajustado para as necessidades mais específicas de uma organização.
2. O RUP possui uma medida de desempenho. Embora indicadores numéricos possam ser introduzidos como parte do processo de software, essa medida de desempenho não precisa ser sempre uma métrica numérica. Atividades de revisão são incluídas em vários fluxos de trabalho para avaliar o desempenho do processo de desenvolvimento de *software*

- em contraste com metas, planos e padrões estabelecidos, de forma a assegurar a qualidade dos resultados do processo.
3. O RUP possui um processo de tomada de decisão que garante uma ação reguladora à luz de (1) e (2). Esse processo de tomada de decisão assume várias formas. Num escopo amplo, a tomada de decisão está presente em disciplinas como o gerenciamento de projetos e o gerenciamento de configuração e de mudanças. Os diagramas de fluxo de trabalho de visão geral mostram os pontos-chave de tomada de decisão e as ações alternativas. Além disso, o processo de tomada de decisão está presente de forma mais localizada em atividades mais dedicadas previstas em várias disciplinas, tais como planejamento, monitoramento, controle e revisão.
  4. O RUP possui componentes que são, eles mesmos, sistemas com todas as propriedades do RUP. Pode-se entender que os fluxos de trabalho detalhados representam componentes dos fluxos de trabalho de visão geral, que, por sua vez, são componentes do processo como um todo. Cada qual tem seu propósito, sua abordagem para a tomada de decisão, e, para cada um, pode-se estabelecer uma medida de desempenho.
  5. O RUP possui componentes que exibem um grau de conectividade tal que efeitos e ações podem ser transmitidos pelo sistema. Os fluxos de trabalho mostram como os elementos se relacionam, de forma que os artefatos são transformados à medida que os atores, ao assumirem vários papéis, desempenham as atividades do processo de desenvolvimento de *software*.
  6. O RUP existe em sistemas e/ou ambientes mais amplos com os quais interage. Esse ambiente é o de desenvolvimento, que pode estar, conforme descrito inicialmente, em constante mudança.
  7. O RUP possui uma fronteira definida pelos limites do alcance do processo de decisão.
- Ela é definida principalmente pelo escopo das disciplinas e pelos os papéis descritos. Os fatores de mudança que se encontram fora do controle do processo de desenvolvimento de *software* estão além das fronteiras, ou seja, no ambiente.
8. O RUP possui recursos físicos e, por intermédio de participantes humanos, abstratos à disposição do processo de decisão. Os artefatos e papéis, bem como toda a infraestrutura necessária para execução das atividades, definem o conjunto de recursos físicos e abstratos necessários para condução do processo de desenvolvimento de *software*.
  9. O RUP possui garantia de continuidade, não é efêmero, possui estabilidade de longo prazo, recuperará a estabilidade após algum grau de perturbação. Um dos objetivos mais importantes da SSM é permitir a auto-organização do processo de desenvolvimento de *software* e sua adaptação em face de mudanças. Isso garante a continuidade do processo em ambientes turbulentos.
- Portanto, uma vez que o RUP pode ser configurado para satisfazer as condições do conceito formal de sistema, ele pode ser usado como modelo conceitual no contexto da SSM.

## 5.2 Aplicação ao Processo de Desenvolvimento de *Software*

A metodologia que combina SSM e RUP está sendo aplicada do Departamento de Desenvolvimento Eletrônico (DDE) da Smar Equipamentos Industriais Ltda. pela equipe de desenvolvimento do projeto *Syscon*.

A Smar desenvolve, fabrica e comercializa soluções para automação de processos industriais. O DDE é o departamento de pesquisa e desenvolvimento que abriga projetos de *hardware* e *software* dos produtos da empresa. Entre os produtos mais importantes, estão os equipamentos de campo, concebidos para

instalação no chão de fábrica para monitorar e controlar variáveis de processos industriais tais como pressão, vazão, nível, temperatura, densidade etc.

Os equipamentos de campo, até bem pouco tempo atrás, costumavam ser pneumáticos. Atualmente, podem ser comparados a computadores digitais interconectadas em rede. Realizam funções avançadas tais como medição, controle distribuído, geração de alarmes e eventos e suporte a manutenção da planta industrial. A tecnologia na área de controle de processos industriais está em franco processo de evolução, e vários protocolos de comunicação digital entre equipamentos vêm surgindo e se desenvolvendo. Dependendo do porte da aplicação industrial, centenas ou milhares desses equipamentos são necessários para atendê-la. Configurar e manter esses equipamentos é um processo complexo e requer o suporte de ferramentas de software dedicadas muito sofisticadas.

O *Syscon* é a ferramenta de *software* usada na configuração dos equipamentos para aplicações específicas. É um *software* bastante elaborado que vem continuamente evoluindo ao longo de quase duas décadas. O processo de desenvolvimento desse produto precisa ser muito flexível para acomodar as turbulências do meio externo, que incluem, de modo geral, a evolução da tecnologia de automação industrial, do mercado e da engenharia de *software*. De modo mais específico, a evolução inclui, por exemplo, os protocolos de comunicação digital, as normas de segurança para equipamentos industriais, os clientes, as áreas de aplicação, as linguagens, as técnicas e as ferramentas de programação usadas em seu desenvolvimento.

A metodologia sugerida foi aplicada pela equipe de desenvolvimento do *Syscon*, que é composta de cinco pessoas: o coordenador do projeto e quatro desenvolvedores. Esse trabalho ainda não foi finalizado, portanto encontrando-se, atualmente, em andamento. A experiência de

aplicação até o momento é descrita a seguir. Para efeito didático, a descrição está organizada de acordo com os estágios da SSM.

1. A situação-problema não-estruturada: em um primeiro momento, a equipe se reuniu para discutir os problemas processo de desenvolvimento de *software*. Gerou-se uma lista inicial com cerca de vinte situações-problema identificadas pelos integrantes no processo de desenvolvimento de *software*.
2. A situação-problema expressa: em seguida, as situações-problema foram organizadas por afinidade com as disciplinas conhecidas na literatura de engenharia e adotadas pelo RUP. Procurou-se analisar também, conforme sugere a SSM, se as situações-problema apresentavam elementos de estrutura, elementos de processo, relação entre estrutura e processo, ou se elas tinham algo em comum com *Weltanschauung*, isto é, a cultura vigente no departamento ou na equipe de desenvolvimento.
3. Definições essenciais dos problemas relevantes: em seguida, a equipe trabalhou na definição essencial do processo de desenvolvimento de *software* desejado. Procurou-se elaborá-la cuidadosamente de forma que apresentasse todos os elementos de uma definição essencial bem formada, como pede a SSM. A partir de contribuições dos vários participantes, chegou-se a um consenso. O que se deseja do processo de desenvolvimento de *software* pode ser expresso na seguinte definição essencial:

É um sistema de desenvolvimento que concilia as necessidades e expectativas dos clientes e usuários do mercado de automação de processos industriais, os interesses estratégicos da Smar, e o aperfeiçoamento da competência dos desenvolvedores, num processo de desenvolvimento de *software* criado e mantido pela equipe de desenvolvimento, alinhado com o processo de desenvolvimento da Smar, flexível

e capaz de produzir uma arquitetura flexível para acompanhar as mudanças frequentes no mercado, na tecnologia de desenvolvimento e de automação, nas necessidades dos clientes, dos usuários e do mercado e nos interesses estratégicos da Smar.

4. Modelos conceituais: para atender à definição essencial, a criação do modelo conceitual se baseou em duas ferramentas sistêmicas e envolveu, portanto, um duplo esforço de modelamento: o diagnóstico do processo de desenvolvimento através do VSM (Beer, 1972, 1979, 1985) e o mapeamento do processo conforme o RUP. Os resultados do diagnóstico pelo VSM foram apresentados em um outro trabalho (DONAIRES, 2006) e não foram contemplados no escopo deste artigo. O mapeamento do processo no RUP consistiu no trabalho de representação do processo de desenvolvimento atual por intermédio dos diagramas de fluxo de trabalho do RUP. Além de refletir o processo atual de desenvolvimento de *software*, o modelo resultante introduz papéis, atividades e artefatos que faltam no processo atual e estão de alguma forma relacionados com as situações-problema identificadas. O modelo foi implementado em páginas do *MS PowerPoint* como diagramas de fluxo de trabalho navegáveis através de *hiperlinks*.
5. Comparação de 4 com 2: nos diagramas do modelo conceitual, os papéis, as atividades e os artefatos faltantes no processo atual foram diferenciados por uma convenção para identificação visual desses elementos.
6. Mudanças desejáveis e factíveis: nesse estágio percebeu-se que muitas mudanças seriam necessárias para aproximar a realidade do modelo conceitual gerado. A lista de situações-problema é longa e algumas delas, por si só, são suficientemente complexas para merecer atenção dedicada. Isso foi identificado como um risco. A equipe se viu diante de uma situação que a SSM denomina

de hiato intransponível.

7. Ação para melhor a situação-problema: a fim de mitigar o risco da mudança, decidiu-se escolher um problema individual. Escolheu-se uma das situações-problema que se julgou mais urgente e, ao mesmo tempo, positiva para exercitar a SSM de forma didática para a equipe de desenvolvimento. Esse trabalho encontra-se atualmente em andamento.

O trabalho de modelamento feito até então não foi descartado, tornando-se uma referência. Ele define o contexto para mudanças mais localizadas. O fato de endereçar os problemas individualmente permitirá exercitar a SSM para refinar o modelo conceitual do processo de desenvolvimento complexo. Como consequência, o modelo evoluirá, mantendo-se atualizado com as mudanças e alinhado com as visões dos membros da equipe.

## 6 Conclusão

O artigo descreveu uma abordagem de aplicação da *Soft Systems Methodology* (SSM) em conjunto com o *Rational Unified Process* (RUP) ao processo de desenvolvimento de software quando ele precisa ser dinâmico, capaz de se auto-organizar para lidar com as mudanças num ambiente turbulento. A SSM contribui para a discussão contínua do processo de desenvolvimento, uma vez que facilita o aprendizado acerca do processo, promove o debate acerca dos problemas e estimula a ação para melhoria contínua do processo. O RUP oferece uma notação intuitiva para capturar no modelo conceitual da SSM o resultado da aprendizagem acerca do processo de desenvolvimento de *software*.

Até o presente momento, a experiência de aplicação dessa abordagem por uma das equipes de desenvolvimento de *software* de um departamento de pesquisa e desenvolvimento de uma empresa brasileira permite tecer as considerações a seguir.

O pensamento sistêmico *soft* que permeia a SSM liberta os envolvidos no debate, no caso os próprios desenvolvedores, do ônus de pensar que o modelo de processo capturado nos fluxos de trabalho do RUP seja uma fotografia estática da realidade. Segundo o paradigma sociológico subjacente à SSM, a realidade do processo de desenvolvimento não segue nenhum modelo estático retratável em papel. O processo de desenvolvimento é um fenômeno social que emerge a partir das interações dos membros da equipe entre si e com o restante da empresa ao lidar com os desafios do dia-a-dia. Em ambientes turbulentos, para lidar com as mudanças frequentes, ele acaba sendo naturalmente dinâmico; além disso, os envolvidos podem ter visões variadas e variáveis sobre o processo. Portanto, o modelo que representa o processo de desenvolvimento não pode ser reconhecido nem instanciado na prática. O modelo que representa o processo de desenvolvimento é apenas um modelo conceitual de consenso que serve como referência para ser comparado com a realidade e, identificadas as diferenças, estimular o debate.

A percepção desse fato é muito importante, porque frequentemente os desenvolvedores vivem a frustração gerada pela percepção de que vivem num ambiente caótico. Essa frustração deriva do fato de que a realidade percebida por eles não apresenta uma organização estática tão fácil de se compreender quanto os modelos de referência que se aprendem na literatura. Existe uma expectativa de que a realidade siga um modelo ordenado, de que ela apresente regularidade e repetitividade como as típicas de uma máquina. Essa expectativa é uma consequência da formação cartesiana dos profissionais e do legado da administração clássica que permeia o pensamento gerencial. Ela gera a esperança de que os problemas profissionais possam ser claramente identificados, representados em modelos e tratados de forma objetiva. No entanto, isso não é a realidade dos sistemas complexos em ambientes turbulentos.

Quanto mais dinâmico for o processo em busca de auto-organização e adaptação, mais difícil fica identificar a regularidade e a repetitividade de forma objetiva. Então, tal processo pode ser confundido com desorganização. Para sobreviver num ambiente turbulento, que constantemente apresenta mudanças, é preciso que se aprenda a diferenciar a desorganização caótica dessa aparente falta de organização do processo dinâmico de auto-organização e adaptação. O pensamento sistêmico *soft* da SSM contribui significativamente para isso. Em primeiro lugar, SSM permite mapear o processo de desenvolvimento sem a pressão do compromisso de que ele seja um retrato objetivo da realidade. Em segundo lugar, concilia o conforto da utilização de um modelo sistêmico conceitual ordenado com a realidade turbulenta típica dos sistemas de atividade humana. É assim que a SSM promove um debate estruturado, baseado no conceito de sistemas, sobre uma realidade complexa, mal-estruturada. A SSM contribui como uma forma de aprendizagem sobre a realidade complexa. Dessa forma, o pensamento sistêmico “soft” se mostra adequado para lidar com os sistemas de atividade humana.

Por outro lado, a notação dos fluxos de trabalho do RUP é intuitiva e adequada para representar o processo de desenvolvimento de software como um sistema de atividade humana. Ela oferece vários elementos como ferramentas para capturar a aprendizagem sobre o processo de desenvolvimento de software num modelo conceitual.

Por meio de várias iterações da SSM os fluxos de trabalho do RUP podem ser configurados, modificados e melhorados para refletirem a realidade da equipe e do desenvolvimento de *software* de forma cada vez mais adequada, sem a pretensão de se tornar uma representação objetiva dessa realidade. Isso oferece duas possibilidades importantes: (1) chegar a uma representação cada vez mais completa do que aprendeu sobre o processo complexo de

desenvolvimento de *software*; (2) modificar continuamente o processo de desenvolvimento para sua melhoria contínua, sua adaptação às mudanças no ambiente de desenvolvimento e sua adequação às visões dos membros da equipe de desenvolvimento. O resultado é uma forma disciplinada de se tornar objetivo o conhecimento sobre a realidade subjetiva. Pode ser usada, portanto, como uma abordagem à gestão do conhecimento.

Embora seja uma metodologia sistêmica e, portanto, holística, a SSM não obriga as situações-problema a serem tratadas todas de uma só vez. A natureza “*soft*” da SSM permite que se trate a lista de situações-problema por partes, ou seja, uma de cada vez. O fato de se trabalhar aspectos da situação-problema que são localizados em relação ao contexto geral de problema não viola a consistência sistêmica da metodologia; ao contrário, ela foi concebida para lidar com situações mal-estruturadas de sistemas complexos, nas quais não se conhece a princípio todos os detalhes da situação. Com as várias iterações de intervenção sobre a realidade, consegue-se aprender gradativamente como lidar com os problemas.

Tal fato permite reduzir o risco de se perder o controle sobre as mudanças ao se lidar com a complexidade. Além disso, as intervenções na organização podem ser conduzidas num ritmo em que a equipe e a empresa sejam capazes acomodá-las. Traduzido nos termos de Checkland, isso significa reduzir o escopo da melhoria a “um hiato transponível”. Não obstante, o exercício de se lidar com as situações-problema individualmente contribui para a compreensão do sistema total. Dessa forma, à medida que se trata as situações-problema de modo individual, captura-se a compreensão do processo de desenvolvimento como um todo no modelo conceitual.

Outra experiência com aplicação da abordagem sugerida que merece menção diz respeito à geração das definições-raiz. O esforço da

geração dessas permite explorar as visões dos participantes para capturar a essência das situações-problema e dos sistemas relevantes. O processo de definição é um exercício civilizado de expor, ouvir e conciliar as visões individuais dos participantes. Uma vez geradas, as definições-raiz expressam o consenso da equipe, resumem a discussão numa frase e consolidam a aprendizagem até o momento. Essa expressão concisa da natureza dos sistemas que deve melhorar a condição problemática encerra um certo mistério que motiva a continuidade do processo. O modelamento do sistema segue, então, de forma natural; conceitualmente, representa o sistema que deve resolver esse mistério para, enfim, melhorar a condição problemática.

A experiência de aplicação aponta para uma outra possibilidade interessante. Parece ser viável ramificar esforços paralelos de aplicação da abordagem sugerida para tratamento de situações-problema de forma individual. Pode-se, por exemplo, manter uma linha de aplicação para o sistema total e outras paralelas para aspectos localizados do sistema. O objetivo da aplicação para o sistema total é, com o debate estruturado que a SSM promove, integrar as descobertas das aplicações individuais e conciliá-las com a compreensão do todo sistêmico.

A combinação da SSM com o RUP é poderosa na medida em que permite capturar, por meio da notação intuitiva do RUP, a aprendizagem promovida pela SSM. Aplicada pela própria equipe de desenvolvimento, a abordagem descrita promove a auto-organização do processo de desenvolvimento diante de novas condições emergentes, o que caracteriza um processo de desenvolvimento de *software* adaptativo.

## Referências

- BEER, S. **Brain of the Firm**. Londres: Allen Lane, 1972.
- \_\_\_\_\_. **The Heart of Enterprise**. Chichester: Wiley, 1979.

\_\_\_\_\_. **Diagnosing the System for Organizations**. Chichester: Wiley, 1985.

BOEHM, B. **A Spiral Model of Software Development and Enhancement**. IEEE Computer, v.21, iss. 5, p.61-72, may 1988.

\_\_\_\_\_. **Spiral Development: Experience, Principles, and Refinements**. Spiral Development Workshop. February 9, 2000. Edited by Wilfred J. Hansen. Special Report CMU/SEI-2000-SR-008. Pittsburgh: CMU/SEI, 2000.

BOOCH, G. **Object-Oriented Analysis and Design with Applications**. California: Benjamin/Cummings, 1994.

\_\_\_\_\_. **Object Solutions: Managing the Object-Oriented Project**. California: Addison-Wesley, 1996.

CHECKLAND, P. B. **Systems Thinking, Systems Practice**. Chichester: Wiley, 1981.

DONAIRES, O. S. Teoria Geral de Sistemas II, in: MARTINELLI, D. P., VENTURA, C. A. A. **Visão Sistêmica e Administração: Conceitos, Metodologias e Aplicações**. São Paulo: Saraiva, 2006.

\_\_\_\_\_. Programando na Complexidade: Um Modelo Sistêmico-Cibernético de Desenvolvimento e Melhoria de Software. **Anais do II Congresso Brasileiro de Sistemas**. Ribeirão Preto: FEARP/USP, 2006b.

KRUCHTEN, P. **The Rational Unified Process: An Introduction**. Second Edition. Addison-Wesley, 2000.

PRESSMAN, R.S. **Engenharia de Software**. MAKRON, 1995.

**Rational Software Corporation**. Rational Unified Process®. Version 2003.06.12. Copyright © 1987 – 2003.