

## **“Uma Abordagem Sistêmica Crítica à Implantação de um Processo de Desenvolvimento de Software”**

*Design de Sistemas de Informação e Tecnologia de Informação*

### **Omar Sacilotto Donaires**

Engenheiro Eletrônico, MBA, PMP  
Rua Rondônia, 146 – Sumarezinho – 14055-230  
Ribeirão Preto, SP  
Fone: (16) 3946-3582  
e-mail: omarsd@smar.com.br

### **Resumo:**

O processo de desenvolvimento de software de grande porte pode ser bastante complexo, principalmente no caso de desenvolvimento de inovação. Não existe uma solução pronta para o processo de software que possa ser aplicada diretamente, e então é preciso conceber o processo. Os envolvidos na concepção precisam fazer escolhas que determinarão as características do processo de desenvolvimento com conseqüências para os seus clientes, que não foram envolvidos na concepção do processo mas que são afetados positiva ou negativamente pelos seus resultados. As escolhas são inevitavelmente influenciadas pelas crenças e valores dos envolvidos. Os julgamentos deles precisam se tornar explícitos para que os afetados possam eventualmente questioná-los.

A Heurística Sistêmica Crítica (HSC) é uma metodologia sistêmica crítica que possibilita levantar os julgamentos que os envolvidos fizeram na concepção do processo de desenvolvimento de software e oferece ferramentas para questionar tais julgamentos.

Um exercício da aplicação da HSC ao processo de desenvolvimento de sistemas software para aplicações de automação de processos industriais permite ilustrar a capacidade heurística da metodologia de evidenciar as fontes de engano potenciais e sua capacidade crítica de identificar as imperfeições da concepção do processo de software.

### **Abstract:**

The software development process for large software systems can be vary complex, mainly in the case of development of innovation. There is no solution for the software process that is ready-to-use, i.e., that can be applied in a straightforward manner. Therefore it is necessary to conceive the process. Those involved in such a conception need to make choices that will determine the features of the development process with consequences for its clients, which were not involved in the conception of the process but are positively or negatively affected by its results. The choices are inevitably influenced by the beliefs and values of the involved. Their judgments need to be made explicit so that the affected can eventually question them.

The Critical Systems Heuristics (CSH) is a critical systems methodology that makes possible to survey the judgments made by the involved during the conception of the software development process and offers tools to question such judgments.

The application of the CSH to the development process of software systems for industrial process automation applications helps to illustrate the heuristics ability of the methodology to make evident the potential sources of deception and its critical ability to identify imperfections in the conception of the software process.

### **Palavras chave:**

Crítica, Sistema, Software

## **Introdução**

O processo de desenvolvimento de software de grande porte é geralmente bastante complexo. O crescimento da literatura de engenharia de software aponta para a importância do assunto e a necessidade de uma grande variedade de processos e inúmeras práticas para fazer jus a essa complexidade. Para dar ao assunto uma abordagem didática, a literatura normalmente o apresenta como um corpo de conhecimento estruturado, articulado e organizado, que aos neófitos ou incautos pode causar impressão de objetividade. Porém, a aplicação dos processos e práticas como descritos na literatura raramente é direta. Adaptações e combinações precisam ser feitas na prática para adequá-las às condições particulares de cada organização.

Por causa da urgência por soluções, por causa das pressões a que as pessoas estão sujeitas nas organizações, há uma tendência de se adotar algumas práticas e processos em detrimento de outros, adaptá-los e combiná-los de forma simplesmente pragmática, isto é, baseada no princípio que diz que “verdade é o que funciona,” sem maiores considerações críticas. Práticas e processos são selecionados com base apenas naquilo que se conhece, sendo que o que se conhece é parcial, sem profundidade conceitual ou teórica, aprendido pelo processo de tentativa e erro com incorporação de muitos vícios que acabam distorcendo as metodologias conforme originalmente propostas, e acabam resultando na criação de mitos quanto ao que funciona e o que não funciona “na prática”.

Esse uso não crítico de soluções prontas em situações que envolvem sistemas de atividade humana que precisam coordenar interesses de diferentes grupos, em outras palavras, esta abordagem funcionalista, resulta numa ilusão objetivista: o uso não crítico produz a impressão de que se está empregando soluções objetivas que levam a resultados concretos comprovados, mas acabam na verdade conduzindo à desilusão quando tais soluções “objetivas” falham em face da subjetividade do elemento humano que é indispensável no desenvolvimento de software. Portanto, uma solução crítica para dirigir escolhas das quais não se pode fugir é essencial para o problema.

Ser crítico nesse contexto não se refere simplesmente àquela atitude voluntária honesta de julgar as próprias decisões, porém segundo critérios pessoais arbitrários. Ser crítico nesse contexto refere-se a um processo sistemático de busca das fontes de engano que as decisões subjetivas ocultam. E é isso que a HSC propõe: uma abordagem sistemática, ou melhor ainda, uma abordagem sistêmica ao processo crítico.

A Heurística Sistêmica Crítica (HSC) foi desenvolvida por Werner Ulrich no seu trabalho seminal (1983) que lançou as bases para uma abordagem crítica no pensamento sistêmico. Seu trabalho é um dos trabalhos mais relevantes, senão o mais relevante, para o pensamento sistêmico crítico.

### **Heurística Sistêmica Crítica (HSC)**

Ulrich usa os três termos – “heurística”, “sistema”, “crítica” – no sentido dado a eles por Immanuel Kant, o pai da filosofia crítica (1983, p19ss):

1. Heurística: Heurística é o que ajuda a descobrir perguntas relevantes para o problema e conhecimento relevante para o problema. Ela ajuda o investigador na tarefa de descobrir e desdobrar problemas. Não há nenhuma garantia contra o engano na descoberta. Portanto, de um ponto de vista crítico, é preciso assumir o engano, ou seja, promover a reflexão do planejador nas fontes de engano nas suas “descobertas.”
2. Sistema: Kant entende o conceito de sistemas como se referindo à totalidade de condições relevantes das quais julgamentos teóricos e práticos dependem, incluindo julgamentos a priori metafísicos, éticos, políticos, e ideológicos básicos. A idéia de sistema como a HSC a entende não pressupõe que se possa conhecer “o sistema todo”, mas somente que se pode empreender um esforço crítico de refletir na inevitável falta de compreensibilidade no nosso entendimento e projeto de sistemas (sociais).
3. Crítica: Ser crítico significa discernir ou julgar cuidadosamente com a intenção de precaver-se contra o erro. Tal julgamento pressupõe padrões ou normas implícitas ou explícitas contra as quais julgar algo. Em tempos modernos “ser crítico” veio a significar o questionamento das próprias normas. Com Kant em particular, o próprio criticismo se torna uma norma quase absoluta “a que tudo deve estar sujeito.” “Ser crítico” então significa acima de tudo se tornar auto-reflexivo com respeito às pressuposições que fluem nos próprios julgamentos, tanto na busca do conhecimento verdadeiro quanto da ação racional.

### **O sentido crítico da idéia de sistemas**

Ulrich (1983, p. 223ss) reclama que o conceito de sistemas como proposto por Kant é freqüentemente depreciado pelos cientistas sistêmicos da tradição funcionalista que o rotulam de “holístico” e desdenham do fato de que seja impossível “conhecer o sistema todo”. Porém, explica ele, a idéia de sistemas, entendida como uma idéia crítica “inevitável” da razão, *não* pressupõe que se possa conhecer “o sistema total” mas somente que se empreenda um esforço conceitual crítico para refletir na

inevitável falta de compreensibilidade nos nossos mapas, tendo em mente a totalidade de condições desconhecida que pode distorcê-los.

Sempre que se aplica o conceito de sistemas a uma seção do “mundo real,” tem-se que fazer suposições a priori muito fortes acerca do que pertence ao sistema em questão e o que pertence ao seu “ambiente.” Ulrich (ibid., p. 225ss) chama tais julgamentos de *julgamentos de fronteira*. Não é a realidade “lá fora” que determina a fronteira entre o sistema e o ambiente, mas antes o ponto de vista do investigador, o propósito do seu esforço de mapeamento, suas concepções pessoais da realidade a ser mapeada e os valores que ele associa a ela. Isso significa que *todos os julgamentos de fronteira, e os mapas ou projetos dos sistemas dos quais eles são constitutivos, possuem um conteúdo normativo carente de reflexão crítica*.

O estudo de fronteira não deve se restringir ao “é” mas sempre incluir o “deveria.” Se um julgamento de fronteira é racional ou não depende menos de quais fronteiras sejam presentemente estabelecidas do que de quais deveriam ser as fronteiras, dado o propósito do modelo. O conteúdo normativo da resposta à pergunta de quais deveriam ser as fronteiras não pode ser justificado pela referência à disponibilidade de dados, ou fronteiras atualmente aceitas, ou o sucesso da ação instrumental. O conteúdo normativo só pode ser justificado através do consentimento voluntário daqueles que possam ser afetados pelas conseqüências.

#### A estrutura conceitual básica da Heurística Crítica

Segundo Ulrich (1983, p. 241s) a expressão fenomenal da intencionalidade humana que é constitutiva da realidade social não pode ser adequadamente capturada no nível sintático, nem mesmo no semântico, mas somente no pragmático (no sentido semiótico da teoria dos sinais). Por isso, ele introduz a dimensão de mapeamento pragmático e, com ela, doze categorias de mapeamento pragmático conforme a Figura 1.

<i>Categorias</i>	<i>Questões centrais cobertas</i>
1. Cliente	Fontes de <i>motivação</i> (de S)
2. Propósito	
3. Medida de melhoramento	
4. Tomador de decisão	Fontes de <i>controle</i> (de S)
5. Componentes	
6. Ambiente de decisão	
7. Planejador	Fontes de <i>especialização e implementação</i> (de S)
8. Especialização	
9. Garantidor	
10. Testemunha	Fontes de <i>legitimação</i> (de S)
11. Emancipação	
12. Visão de mundo	

Os envolvidos } O sistema social S a ser delimitado.  
Os afetados }

**Figura 1: Categorias heurísticas de mapeamento pragmático. Fonte: Ulrich (1983, p. 258)**

Para suportar um processo sistemático de crítica de fronteira as doze categorias são traduzidas em doze perguntas críticas de fronteira no modo “é” e no modo “deveria”. (Ulrich, 2002)

<i>“É”</i>		<i>“Deveria ser”</i>	
1.	Quem é o verdadeiro cliente do projeto de S?	1.	Quem deveria ser o cliente (beneficiário) do sistema S a ser projetado ou melhorado?
2.	Qual é o verdadeiro propósito do projeto de S?	2.	Qual deveria ser o propósito de S, isto é, que metas S deveria ser capaz de alcançar para servir ao cliente?
3.	Qual é, a julgar pelas conseqüências do projeto, sua medida interna de sucesso?	3.	Qual deveria ser a medida de sucesso (ou aperfeiçoamento) de S?
4.	Quem é verdadeiramente o tomador de decisão, ou seja, quem pode de fato mudar a medida de sucesso?	4.	Quem deveria ser o tomador de decisão, ou seja, ter o poder de mudar a medida de aperfeiçoamento de S?
5.	Que condições de planejamento e implementação bem sucedidas de S são verdadeiramente controladas pelo tomador de decisão?	5.	Que componentes (recursos e restrições) de S deveriam ser controladas pelo tomador de decisão?
6.	Que condições não são controladas pelo tomador de decisão, isto é, o que representa “ambiente” para ele?	6.	Que recursos e condições deveriam ser parte do ambiente de S, isto é, não deveriam ser controlados pelo tomador de decisão?

7.	Quem está verdadeiramente envolvido como planejador?	7.	Quem deveria estar envolvido como projetista de S?
8.	Quem está envolvido como “especialista”, de que tipo é sua especialidade, que papel ele verdadeiramente desempenha?	8.	Que tipo de especialidade deveria fluir no projeto de S, isto é, que deveria ser considerado um “especialista” e qual deveria ser o seu papel?
9.	Onde os envolvidos vêm a garantia de que seu planejamento será bem sucedido?	9.	Quem deveria ser o garantidor de S, isto é, onde o projetista deveria procurar a garantia de que seu planejamento será implementado e se mostrar bem sucedido, a julgar pela medida de sucesso (ou aperfeiçoamento) de S?
10.	Quem dentre as testemunhas envolvidas representa as preocupações dos afetados? Quem é ou pode ser afetado sem estar envolvido?	10.	Quem deveria estar entre as testemunhas representando as preocupações dos cidadãos que serão ou poderiam ser afetados pelo projeto de S? Quer dizer, quem dentre os afetados deveria ser envolvido?
11.	É dada aos afetados uma oportunidade de se emanciparem dos especialistas e tomar seu destino nas suas próprias mãos?	11.	Em que grau e de que maneira deveria ser dada aos afetados a chance de emancipação das premissas e promessas dos envolvidos?
12.	Que visão de mundo é verdadeiramente subjacente ao projeto de S? É a visão de mundo de um (alguns) dos envolvidos ou de um (alguns) dos afetados?	12.	Sobre que visões de mundo dos envolvidos ou afetados o projeto de S deveria se basear?

### Os três padrões críticos

Ulrich (1983, p. 259ss) introduz também três idéias que servem como padrões críticos de reflexão e argumentação acerca das fontes de engano em mapas ou projetos alternativos: a idéia de sistemas, a idéia de moral, e a idéia de garantidor.

A idéia de sistemas representa o ideal de compreensibilidade do ponto de vista das condições, e porque tal compreensibilidade é *somente* um ideal, ela sugere uma falta de compreensibilidade em todos os nossos mapas. Ela sugere a necessidade de reflexão crítica nos julgamentos de sistema total subjacentes; a intenção da lista de categorias é ajudar o planejador a conceber todos os julgamentos de sistema total em questão.

A idéia de moral requer de um agente que ele reflita no grupo total de indivíduos que podem ser afetados pelas suas ações. Ela representa o ideal de perfeição moral e, por ser um ideal, sugere a imperfeição moral de todos os nossos projetos, e a necessidade de refletir nas fontes e potenciais conseqüências de tal imperfeição.

A idéia de garantidor faz lembrar que não há nenhuma garantia de melhoramento através de planejamento. É exatamente por isso que o planejador deve esforçar-se para incorporar no seu esforço de projeto tantas fontes de garantia (imperfeitas) quantas for possível.

O planejador as empregará essas três idéias antes de tudo “monologicamente,” ou seja, ele as usará para melhorar seu entendimento dos seus mapas e projetos e para eliminar deficiências antes de submetê-las ao escrutínio público. Mas as idéias podem ser usadas também discursivamente, como pontos de referência para argumentação cogente num discurso prático para desdobrar as implicações críticas das três idéias.

### O processo desdobramento

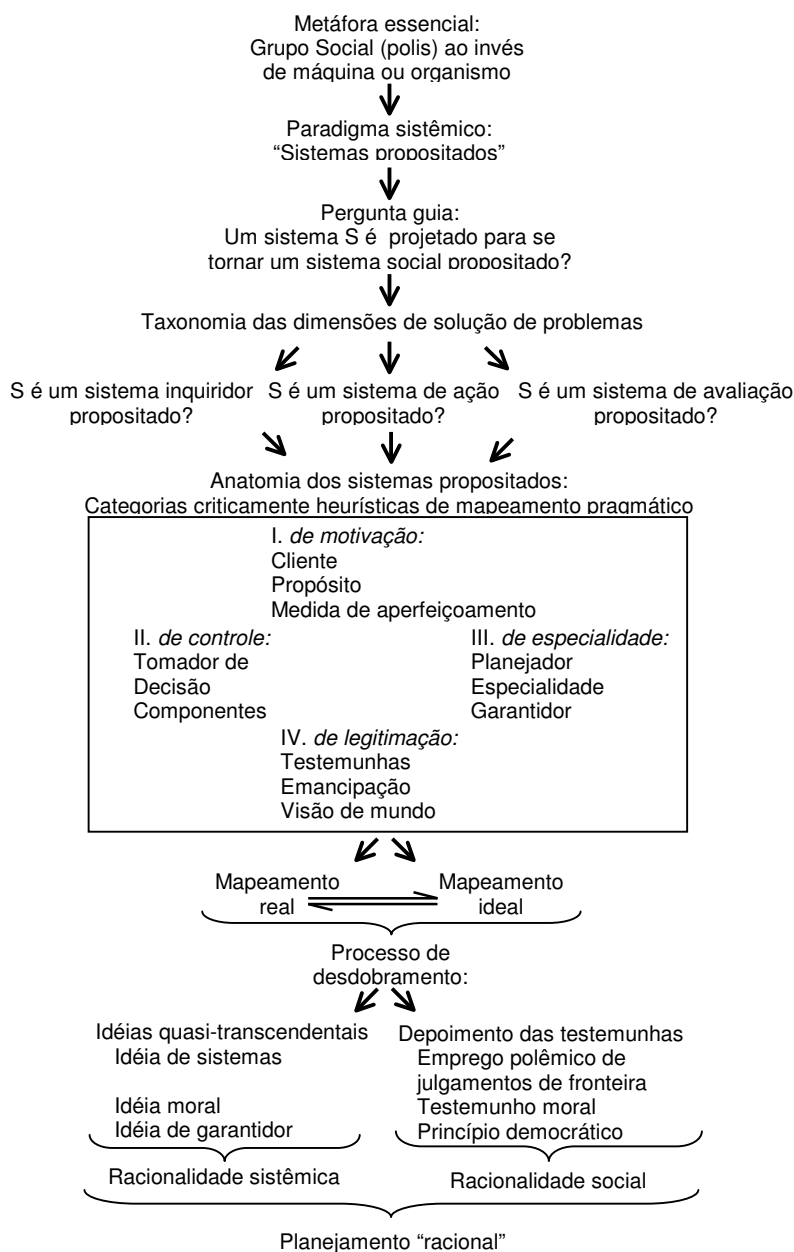
Ulrich chama de *processo de desdobramento* à interação entre planejadores (“racionalidade sistêmica”) e testemunhas (prática social vivida). Ele propõe focalizar esse processo em três princípios fundamentais: o princípio da “dialética”, “o emprego polêmico da razão” e o princípio democrático.

Quando fala da relação dialética entre racionalidade sistêmica e racionalidade social, os envolvidos e os afetados, etc., Ulrich (1983, p. 277s) se refere a duas posições que são complementares: um lado serve como a fonte de conceitos teóricos para o outro, enquanto o último serve como a fonte de conceitos práticos para o primeiro. Idealmente a racionalidade sistêmica e a racionalidade social convergem. Na prática, mais freqüentemente elas conflitam. Mas talvez isso seja bom, na medida em que impedem uma à outra de sucumbir a, ou propositalmente criar, uma ilusão de compreensibilidade e objetividade (id., *ibid.*, p. 298).

Através do emprego polêmico da razão as testemunhas podem expor o caráter dogmático das “necessidades objetivas” dos especialistas através dos seus próprios argumentos subjetivos (id., *ibid.*, p. 305). Uma solução crítica pode somente visar o desdobramento da dialética entre os dois lados de tal maneira que um lado desafie o outro a tornar transparentes suas pressuposições normativas. Somente tal transparência de valores pode impedir qualquer lado de reivindicar racionalidade, objetividade ou perfcia quando na realidade ele é dogmático, amarrado a interesses ou incompetente (id., *ibid.*, p. 301).

Argumentos polêmicos não precisam demonstrar a sofisticação do interlocutor mas só o embaraço do seu oponente. A importância de tal emprego polêmico da razão é que ele é acessível a todo cidadão afetado e às testemunhas que o representam vis-à-vis os envolvidos, dando-lhe uma vantagem sobre os especialistas desde que usado para propósitos críticos somente. O emprego polêmico da razão é inerentemente democrático porque seu poder crítico não depende de qualquer conhecimento teórico além do bom senso. Assim ele ajuda o cidadão a se emancipar dos especialistas que com suas “necessidades objetivas” minam seu status democrático de soberania (id., ibid., p. 308).

O princípio democrático é o princípio da soberania e igualdade dos cidadãos, sejam eles envolvidos ou “meramente” afetados pelo planejamento.



**Figura 2: Abordagem criticamente heurística ao projeto e avaliação de sistemas propositados.**

**Fonte: Ulrich (1983, p. 341s)**

### O paradigma dos sistemas propositados

A Figura 2 mostra a estrutura geral da abordagem criticamente heurística ao projeto e avaliação de sistemas propositados. A HSC associa o conceito de sistema a uma metáfora essencial de grupo social, e adere ao paradigma sistêmico dos sistemas propositados conforme C. West Churchman.

Um sistema propositado é um sistema que é auto-reflexivo com respeito às suas implicações normativas, visto do ponto de vista não somente dos envolvidos mas também dos afetados, e que tem autonomia pelo menos parcial para determinar seu cliente, seus propósitos etc. “Autonomia parcial” significa que o sistema pode exercer sua própria vontade na escolha das suas metas (id., *ibid.*, p. 334).

A tarefa fundamental de projeto do planejador social é projetar para o desenvolvimento de motivação intrínseca e reflexão crítica por parte daqueles que terão que trabalhar e viver com seus projetos. Esta tarefa se aplica a cada um de três tipos básicos de processos complementares de solução de problemas que um sistema propositado deve inevitavelmente realizar: inquirição (produzir conhecimento significativo com respeito ao seu propósito), ação (garantir o uso propositado desse conhecimento) e avaliação (avaliar sua produção e uso de conhecimento do ponto de vista tanto do cliente quanto daqueles que podem ser negativamente afetados).

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

O estabelecimento de um bom processo de desenvolvimento de software requer que se escolha dentre inúmeras práticas e processos disponíveis quais sejam relevantes para uma determinada situação. Isso põe os envolvidos nas escolhas diante de julgamentos de fronteira. A HSC permite desafiar as decisões dos envolvidos em busca de um processo de desenvolvimento mais justo no atendimento das necessidades da comunidade de usuários que são positiva ou negativamente afetados pelo software que ele produz, mas que não foram envolvidos na concepção do processo de desenvolvimento de software a não ser através das suas “testemunhas”.

A análise a seguir não consiste num estudo de caso específico, mas trata-se de um ensaio de aplicação da HSC à concepção de um processo de desenvolvimento de sistemas de software de grande porte para aplicações de automação industrial, de acordo com a experiência do autor. A intenção é ilustrar a aplicação da metodologia.

#### **O processo de desenvolvimento de software como um sistema propositado**

Para justificar a aplicação da HSC ao processo de desenvolvimento de software é preciso entendê-lo como um grupo social e um sistema propositado. De fato o processo de desenvolvimento de software é um sistema de atividade humana que precisa conciliar interesses divergentes. Por isso, ele precisa ser concebido de forma a produzir motivação intrínseca e reflexão crítica por parte dos desenvolvedores que terão que trabalhar no processo de produção do software, e dos usuários que terão que conviver com o sistema de software produzido. Essa motivação e essa reflexão devem guiá-los nas atividades das três dimensões de solução de problema de um sistema propositado: inquirição, ação e avaliação.

O processo de desenvolvimento de software é um sistema inquiridor propositado na medida em que ele precisa facilitar aos desenvolvedores a descoberta das necessidades objetivas e subjetivas dos usuários do software. É justo interpretar a tarefa de levantamento dos requisitos de um sistema de software de grande porte como um processo de investigação desafiante. Os requisitos do sistema de software nunca serão plenamente conhecidos. Geralmente os clientes e usuários não sabem exatamente o que querem. Quando sabem o que querem não conseguem expressá-lo na linguagem dos desenvolvedores. Quando conseguem expressá-lo, percebem que suas necessidades mudam com o tempo, mesmo durante o próprio tempo de desenvolvimento do produto. Daí a importância da motivação para descobrir os requisitos e acompanhar a sua evolução, e a importância da reflexão crítica para o sucesso do processo.

O processo de desenvolvimento de software também é um sistema de ação propositado na medida em que ele precisa se capaz de transformar as especificações das necessidades em um produto concreto. Ele será tanto mais bem sucedido quanto maior for sua motivação em realizar as expectativas e sonhos dos seus usuários, sempre considerando essas necessidades de forma crítica.

O processo de desenvolvimento de software também é um sistema de avaliação propositado na medida em que ele seja capaz de validar o software contra os requisitos, verificar sua correção, avaliar a extensão na qual a satisfação do cliente foi alcançada, e o quanto as expectativas dos usuários foram satisfeitas. Será um sistema de avaliação moralmente responsável se incluir entre as suas preocupações os eventuais impactos negativos da automatização introduzida pelo sistema de software – por exemplo, demissões que possam ocorrer por causa da implantação do sistema.

Uma vez entendida a sua natureza de sistema propositado, pode-se explorar o conteúdo normativo do processo de desenvolvimento de software através da aplicação das doze perguntas correspondentes às categorias criticamente heurísticas, tanto no modo “é” quanto no modo “deveria ser”.

#### **Como “é” processo de desenvolvimento de software**

As doze perguntas no modo “é” aplicadas ao processo de desenvolvimento de software permitem expor os julgamentos de fronteira dos projetistas na concepção do processo. Como não se trata de um estudo de caso específico, não é possível responder definitivamente às perguntas no modo “é”, senão explorar algumas possibilidades.

### **(1) Quem é o verdadeiro cliente do projeto do processo de desenvolvimento de software?**

Na área de automação de processos há vários candidatos a clientes do processo. Há os desenvolvedores, interessados numa oportunidade de exercitar, desenvolver e tornar visíveis suas competências técnicas e pessoais; a empresa desenvolvedora que precisa controlar os recursos empregados no desenvolvimento e esperam retorno financeiro. Entre os afetados estão: o pessoal da engenharia de aplicações da empresa desenvolvedora ou os integradores (terceiros) que desenvolvem soluções para o cliente final, interessados na facilidade de uso; o pessoal da assistência técnica que oferece suporte ao cliente final no caso de problemas após a instalação e precisam que o sistema ofereça diagnósticos precisos e seja fácil de reparar; o pessoal do marketing que precisa de apelo comercial para o produto; os clientes finais, interessados na produtividade da sua planta industrial, na modernização e segurança do seu processo, e na conformidade com a regulamentação governamental; e os usuários do sistema software, que esperam que ele seja amigável, facilite seu trabalho e, em última análise, melhore suas condições de trabalho. Os usuários compõem um grupo bastante diversificado do ponto de vista de expectativas e interesses: os operadores precisam que ele tenha uma interface amigável e seja rápido para interagir com o processo industrial; o pessoal da manutenção precisa que o sistema ofereça diagnósticos de problemas confiáveis e facilite a localização e a pronta correção de problemas que ocorrem nas horas mais impróprias; os engenheiros de processo que sonham com um sistema aberto, que possa ser estendido com facilidade e integrado a outros sistemas de outros fabricantes.

Entre as vítimas de um sistema de desenvolvimento de software podem eventualmente estar aqueles que perdem seus postos de trabalho por causa da automatização introduzida com os sistemas de software. Se o sistema de software for uma solução proprietária fechada e não padronizada, então o próprio cliente final que é (ou deveria ser) o maior beneficiário, acaba ficando refém da organização que o desenvolveu para quaisquer correções, adições, ou evolução.

### **(2) Qual é o verdadeiro propósito do projeto do processo de desenvolvimento de software?**

O processo de desenvolvimento de software pode ser concebido apenas para produzir código vendável sob pressão em grande quantidade, para agregar novas funcionalidades e reduzir o tempo de mercado. Ou ele pode ser concebido para atender cuidadosamente às especificações do produto, produzir qualidade e acomodar a participação de outros interessados no processo. Ele pode ser concebido para produzir uma boa documentação do projeto que facilite modificações futuras; ou para produzir uma arquitetura de software resiliente, que permita evolução no futuro para atender necessidades emergentes ainda desconhecidas dos interessados.

De maneira geral, um propósito imediatista pode reduzir o tempo de desenvolvimento e o custo, favorecendo a organização desenvolvedora a curto prazo. Por outro lado, se o processo de desenvolvimento for concebido para atender nos detalhes a todas as especificações do produto conforme o desejo dos usuários, produzir uma boa documentação e uma arquitetura de software resiliente para facilitar o trabalho dos desenvolvedores no futuro, acomodar a participação dos interessados no processo, então a satisfação será maximizada, pelo menos em termos de funcionalidade, se o tempo de mercado e o custo de desenvolvimento não for um problema para todos.

Os interesses são naturalmente conflitantes, de forma que nem todos podem ser atendidos plenamente, pelo menos não num primeiro momento. Então é comum que o propósito do sistema de desenvolvimento mude e evolua com o tempo.

### **(3) Qual é, a julgar pelas conseqüências do projeto, sua medida interna de sucesso?**

Um processo de desenvolvimento de software maduro adota algum tipo de métrica. O sucesso pode ser medido pela satisfação dos clientes e usuários, pela felicidade dos desenvolvedores, pela maximização do retorno para a empresa desenvolvedora, pela orientação de marketing, ou uma combinação dessas alternativas. Infelizmente, se o processo não for maduro o suficiente essas métricas ou algumas delas não são confiáveis porque são baseadas em dados imprecisos ou manipulações artificiais de informações que sutilmente conduzem a conclusões distorcidas acerca da realidade. Nesses casos, na melhor das hipóteses as métricas apontam para uma conclusão óbvia e tão pouco útil quanto geral: que o processo é falho e precisa ser melhorado. O Capability Maturity Model do SEI, SEI-CMM (PAULK, 1993), define um modelo de maturidade de processo que é uma referência muito boa para avaliação e implementação de processos de desenvolvimentos de software.

### **(4) Quem é verdadeiramente o tomador de decisão, ou seja, quem pode de fato mudar a medida de sucesso?**

Num ambiente em que a inovação é um fator importante os desenvolvedores inclusive gozam da liberdade necessária para estimular a criatividade e têm, por isso, algum poder de decisão. Se o processo de software não previr de alguma forma a participação dos outros interessados os desenvolvedores assumirão o controle do processo, privilegiando, de forma intencional ou não, a sua idéia de sucesso.

A empresa desenvolvedora, porque detém o controle sobre os recursos de desenvolvimento, pode tentar impor sua concepção de sucesso.

**(5) Que condições de concepção e implantação bem sucedidas do processo de desenvolvimento de software são verdadeiramente controladas pelo tomador de decisão?**

A definição dos sub-processos e a escolha das ferramentas mais adequadas ao desenvolvimento considerando-se a estratégia e o mercado da empresa desenvolvedora são condições de concepção e implantação bem sucedidas do processo de desenvolvimento de software. Isso inclui definir qual será o ciclo de vida de desenvolvimento (PRESSMAN, 1995, p. 30ss; KRUCHTEN, 2000; BOHEM, 2000), como será o gerenciamento da configuração do software (PRESSMAN, 1995, p. 917ss), como será feito o controle de mudanças do software (id., ibid., p. 930ss), como será a verificação e a validação do software (id., ibid., p. 836), qual será a linguagem de modelamento (BOOCH, RUMBAUGH e JACOBSON, 1999), qual será a linguagem de programação e o compilador utilizado etc.

Após a implantação, o grau de sucesso do processo dependerá da sua flexibilidade para acomodar mudanças, da sua agilidade para mudar as prioridades de desenvolvimento, e da sua abertura à participação das testemunhas dos afetados.

**(6) Que condições não são controladas pelo tomador de decisão, ou seja, o que representa “ambiente” para ele?**

Por melhor que seja a engenharia de requisitos, alguns requisitos serão descobertos somente quando o sistema for implantado. Quanto às necessidades futuras, são insondáveis. As tecnologias de desenvolvimento e agregadas ao produto de software evoluem numa velocidade maior do que os desenvolvedores possam acompanhar.

**(7) Quem está verdadeiramente envolvido na concepção do processo de desenvolvimento?**

Consultores externos podem estar envolvidos na concepção do processo de software. Algumas empresas de consultoria vendem soluções prontas como o RUP (KRUCHTEN, 2000), que podem ser adaptadas aos mais diversos contextos de desenvolvimento. Elas vendem suítes completas de ferramentas, serviço de suporte anual, e o treinamento dos desenvolvedores. O custo de implantação e manutenção desse tipo de solução ainda exclui empresas menores. Além disso, há o perigo da rejeição da solução não seja feita uma adequação cuidadosa à cultura da organização desenvolvedora.

Alternativamente os próprios desenvolvedores podem criar o processo de software baseados na sua experiência. O perigo dessa abordagem é que os desenvolvedores freqüentemente aprendem a usar uma ferramenta de desenvolvimento sem entender o processo de desenvolvimento por trás dela. Com isso eles acabam adquirindo e disseminando muitos vícios que os impedem de explorar ao máximo o potencial de tais ferramentas.

Em alguns casos, a organização tem um grupo de engenharia de software, que dá apoio à implantação ou melhoria do processo de desenvolvimento.

**(8) Quem está envolvido como “especialista”, de que tipo é sua especialidade, que papel ele verdadeiramente desempenha?**

Os engenheiros de software e os engenheiros de qualidade de software é que podem contribuir mais efetivamente por causa da sua especialização nos processos de software em geral e do seu conhecimento abrangente das ferramentas, técnicas e disciplinas da engenharia de software.

Entre os desenvolvedores há também uma variedade de especialidades: arquitetos de software, projetistas, engenheiros de aplicação, programadores, testadores etc. São eles que enfrentam o dia-a-dia do desenvolvimento e podem portanto identificar que práticas e ferramentas seriam adequadas à solução dos seus problemas.

O cliente final e os usuários do sistema de software são especialistas nas áreas de aplicação do produto e precisam ser consultados porque eles em última análise darão a palavra final quanto à sua utilidade e a usabilidade.

**(9) Onde os envolvidos vêm a garantia de que a implantação do processo de desenvolvimento de software como concebido será bem sucedida?**

A garantia de uma implantação bem pode estar na experiência dos desenvolvedores, engenheiros de software ou consultores envolvidos na concepção do processo de desenvolvimento. O apoio da alta administração da empresa é vital, e então, a viabilidade econômica do processo é um garantidor do ponto de vista da organização que o está financiando. Adequação às características culturais da organização é outro garantidor.

A adaptabilidade do processo de software é também uma garantia, já que as condições internas e externas jamais permanecem estáticas e o processo precisa ser continuamente melhorado ou ajustado a novas condições.

Como a implantação bem sucedida é aquela que implanta um processo de desenvolvimento capaz de produzir software que atende às necessidades do cliente, então, a garantia está também em acomodar a participação dos clientes e usuários.

Algumas técnicas de desenvolvimento trazem garantias específicas. A análise de domínio (BOOCH, 1994, p. 157) provê aderência do modelo do software ao mundo real, garantindo a adaptabilidade do software quando ocorrerem mudanças no domínio de problema. A análise de casos de uso (BOOCH, 1994, p. 158; COCKBURN, 2001) provê uma referência para guiar processo de desenvolvimento ao longo da análise, projeto, implementação e testes, e ajuda a garantir que o software atenderá às necessidades dos seus usuários.

**(10) Quem dentre as testemunhas envolvidas representa as preocupações dos afetados? Quem é ou pode ser afetado sem estar envolvido?**

Já que os desenvolvedores são os primeiros a serem afetados, a participação deles na concepção e implantação do processo de desenvolvimento é fundamental. Se eles forem neófitos e não conseguirem ajudar na concepção do sistema, ou se o processo de software for implantado por uma empresa de consultoria interessada apenas em vender um produto pronto sem a devida adequação à realidade dos desenvolvedores, estes últimos poderiam se tornar vítimas de um processo desalinhado da sua cultura ou inadequado às suas necessidades. Se por outro lado eles participarem ativamente da concepção e implantação do processo de desenvolvimento, com o suporte da empresa de consultoria, é muito mais provável que suas preocupações sejam devidamente consideradas.

O processo de desenvolvimento precisa prever e estimular tal participação dos clientes e usuários para que suas preocupações sejam devidamente consideradas.

**(11) É dada aos afetados uma oportunidade de se emanciparem dos especialistas?**

No caso de uma implantação inadequada de um processo por uma empresa de consultoria, os desenvolvedores podem ficar dependentes do suporte dos consultores especializados. Se os recursos se tornarem escassos, então os desenvolvedores ficarão amarrados a um processo de software que não resolve seus problemas. Por outro lado, quando eles próprios forem capazes de conceber e implantar o processo, eles serão naturalmente capazes de trabalhar na sua modificação, adaptação, e evolução. É um caso particular de emancipação.

Quanto aos clientes e usuários da empresa desenvolvedora, eles dificilmente conseguirão se emancipar totalmente de um software inadequado, a não ser livrando-se dele e perdendo seu investimento. Por isso, eles dependerão da habilidade e interesse dos desenvolvedores em envolvê-los no processo de desenvolvimento, atendê-los bem toda vez que eles precisarem de algum suporte, e zelar pela sua satisfação.

**(12) Que visão de mundo é verdadeiramente subjacente ao projeto do processo de desenvolvimento de software?**

De modo geral, pode-se dizer que no caso de uma consultoria externa e desenvolvedores inexperientes, a visão de mundo dos consultores envolvidos tem a tendência de prevalecer em detrimento da visão dos desenvolvedores afetados. Se os próprios desenvolvedores forem capazes de conceber e implantar o processo, predominará a visão de mundo dos desenvolvedores. Há ainda a visão de mundo dos clientes e usuários.

**Como “deveria” ser o processo de desenvolvimento de software**

As doze perguntas no modo “deveria” aplicadas ao processo de desenvolvimento de software permitem ir além das limitações do modelo concebido pelos projetistas e levantar alternativas, que possam se mostrar mais justas e responsáveis.

**(1) Quem deveria ser o cliente (beneficiário) do processo de desenvolvimento de software a ser projetado ou melhorado?**

Os usuários e os clientes finais precisam estar entre os beneficiários mais importantes porque é da sua satisfação com o produto final é que depende a viabilidade do processo. Em seguida, os interesses e as preocupações dos desenvolvedores precisam ser considerados com a devida atenção, porque são eles que viverão o dia-a-dia do processo, e personificarão a motivação intrínseca e capacidade de reflexão crítica do processo.

**(2) Qual deveria ser o propósito do processo de desenvolvimento de software?**

O propósito primário do processo de desenvolvimento de software deveria ser investigar cuidadosamente as necessidades e expectativas dos usuários e clientes finais, produzir um sistema de software para satisfazê-las com qualidade, e avaliar a qualidade do processo de desenvolvimento, do produto final e a satisfação dos usuários.

Outro propósito importante para a sustentabilidade do processo, é o desenvolvimento pessoal e motivação dos desenvolvedores. Se o processo lhes permitir expressar sua competência criativa e promover o seu crescimento pessoal, eles zelarão pela continuidade e pelo aperfeiçoamento do processo.

**(3) Qual deveria ser a medida de sucesso (ou aperfeiçoamento) do processo de desenvolvimento de software?**

A medida de sucesso do processo deveria ser uma indicação do grau de satisfação das necessidades dos usuários finais e dos clientes, ou seja, do grau em que suas necessidades e expectativas

puderam ser efetivamente descobertas e satisfeitas no software desenvolvido e implantado. A produtividade dos desenvolvedores e do processo também são métricas importante para aperfeiçoamento do processo como um todo. Num processo socialmente responsável a métrica deveria levar em conta também o grau em que as vítimas eventuais do processo foram prejudicadas.

**(4) Quem deveria ser o tomador de decisão?**

À empresa desenvolvedora deveria caber a decisão sobre quais clientes e projetos têm importância estratégica. Uma vez aceito um determinado projeto, o cliente deveria ter primazia nas decisões quanto à definição escopo do produto no início do processo e mudanças de escopo no desenrolar do mesmo.

Os desenvolvedores deveriam participar das decisões que envolvem questões de tecnologia, já que eles são os especialistas nesse assunto. Os usuários deveriam participar das decisões relativas à interface gráfica e os casos de uso do software. A aprovação final deveria ser do cliente final ou de quem quer que representante seus interesses.

**(5) Que componentes (recursos e restrições) do processo de desenvolvimento de software deveriam ser controladas pelo tomador de decisão?**

Os desenvolvedores, juntamente com os engenheiros de software, deveriam ter participação decisiva na escolha de ferramentas, técnicas e ciclo de vida de desenvolvimento. Os clientes e usuários, em acordo com os desenvolvedores, deveriam ter controle sobre o escopo do produto. A empresa desenvolvedora deveria definir a estratégia de desenvolvimento e delinear o mercado de interesse para o produto.

É impossível que a empresa desenvolvedora tenha controle sobre a evolução tecnológica em geral, mas quando houver espaço, ela deveria se posicionar como agente de mudança participando ela própria das organizações internacionais que definem e padronizam as tecnologias que lhe são essenciais.

**(6) Que recursos e condições deveriam ser parte do ambiente do processo de desenvolvimento de software?**

A empresa desenvolvedora raramente conseguirá ter controle sobre a evolução de todas as tecnologias que ela emprega. Ele deve acompanhar a evolução daquelas que lhe são essenciais e encarar as demais como pertencentes a um ambiente em constante mudança sobre o qual ela não tem controle mas ao qual precisa se adaptar.

O processo de desenvolvimento dificilmente terá controle sobre todos os efeitos negativos da automatização crescente dos processos industriais, por exemplo, o desemprego. Decisões nesse âmbito fogem ao controle dos envolvidos no processo de desenvolvimento de software.

**(7) Quem deveria estar envolvido como projetista do processo de desenvolvimento de software?**

Engenheiros de software de uma empresa de consultoria e/ou da própria empresa desenvolvedora com experiência na área de atuação de empresa desenvolvedora deveriam estar envolvidos no projeto do processo de desenvolvimento de software. Eles deveriam sempre trabalhar em conjunto com os desenvolvedores.

**(8) Que tipo de especialidade deveria fluir no projeto do processo de desenvolvimento de software?**

A engenharia de software é a especialidade fundamental na implantação de um processo de software, e deveria ser a fonte de alternativas na busca de métodos, práticas e ferramentas para compor o processo de software.

**(9) Quem deveria ser o garantidor do processo de desenvolvimento de software?**

Os engenheiros de software devem buscar a garantia do seu projeto na aceitação do processo por parte dos desenvolvedores pela adequação do processo à sua cultura, e na aceitação do projeto pela administração da empresa desenvolvedora oferecendo soluções economicamente viáveis, e na participação dos clientes e usuários no processo de desenvolvimento.

Técnicas como a análise de domínio e a análise de casos de uso oferecem garantias adicionais e deveriam ser incluídas conforme a conveniência.

**(10) Quem deveria estar entre as testemunhas representando as preocupações dos aqueles que serão ou poderiam ser afetados pelo projeto do processo de desenvolvimento de software?**

Os desenvolvedores deveriam participar da concepção e implantação do processo de desenvolvimento de software, assim como os clientes e usuários, ou seus representantes idôneos, deveriam ser envolvidos na produção do sistema de software.

**(11) Em que grau e de que maneira deveria ser dada aos afetados a chance de emancipação das premissas e promessas dos envolvidos?**

O processo de software deveria ser flexível para acomodar a mudanças de forma a permitir que os desenvolvedores possam trabalhar na sua adaptação e evolução conforme a necessidade. Soluções proprietárias onerosas que amarram os desenvolvedores a um fornecedor único deveriam ser evitadas.

Quanto aos clientes e usuários, o sistema de software produzido deveria ser configurável para uso em diferentes situações de acordo com as necessidades de diferentes usuários, deveria ser flexível,

extensível, e aberto à integração com sistema de outros fabricantes. Os clientes e usuários devem certificar-se de que o software seja modular e adote soluções padronizadas que permitam sua substituição total ou parcial, quer dizer, a substituição de um ou mais dos seus módulos, por soluções de outros fabricantes. Isso quer dizer que caso o software se torne obsoleto ou se mostre inadequado ao uso em qualquer situação é preciso garantir a liberdade de escolha aos clientes e usuários.

### **(12) Sobre que visões de mundo dos envolvidos ou afetados o projeto do processo de desenvolvimento de software deveria se basear?**

O processo de desenvolvimento deveria se basear num equilíbrio entre a visão de mundo da engenharia de software, dos desenvolvedores e dos usuários. A visão de mundo da engenharia de software contribui para a eficiência dos métodos e ferramentas, e a confiabilidade das métricas. A visão de mundo dos desenvolvedores pode acrescentar sua experiência cultural na empresa desenvolvedora para escolha e adequação dos métodos e métricas. A visão dos usuários, finalmente, ajuda a focalizar num processo que resulte num software útil e utilizável, ou seja, que faça o que eles precisam da forma como eles esperam.

### **Submetendo o processo de desenvolvimento de software ao teste dos três padrões críticos**

Já foi mencionado que o processo de desenvolvimento dificilmente terá controle sobre os efeitos negativos da automatização, por exemplo, demissões que podem ocorrer em função da implantação do sistema. Os desenvolvedores não têm controle sobre a forma como os clientes conduzirão a implantação do sistema de software. Mesmo o cliente, pressionado pelas forças competitivas, não têm muita escolha senão buscar aumentar a produtividade do seu processo, e a automatização é com certeza uma solução eficaz. Esse aumento da produtividade a expensas de eventualmente contribuir para o aumento do desemprego expressa o risco de sub-otimização da automação industrial em geral, e do processo de desenvolvimento de software em particular.

Evitar a automação seria uma solução para evitar o desemprego. Mas isso deve estar fora de cogitação. Se as empresas nacionais não ocuparem o seu lugar no mercado nacional para prover soluções para a automação de processos industriais, as empresas estrangeiras certamente o farão e o resultado final será pior para o país. Um sistema que considerasse todos esse fatores assumiria uma dimensão planetária e estaria totalmente fora do controle do projetista do processo de software.

Por outro lado, a conscientização dos clientes para re-alocar os profissionais internamente, quando possível, indica uma direção para solucionar o problema. Quando as empresas clientes não puderem arcar com esse ônus social, então o sistema deveria assumir uma dimensão nacional, prevendo a ação de órgãos governamentais para recolocação no mercado de trabalho dos profissionais que perderam seus postos em função da automatização, para a qualificação da mão-de-obra que possa ter se tornado obsoleta, e para a criação de novos postos de trabalho.

Infelizmente, nas condições atuais, não há indícios de que possa existir um sistema propositado para realizar tal missão, quer dizer, um sistema intrinsecamente motivado. Isso significa que, pelo menos por enquanto, o processo de desenvolvimento de software tem que aceitar sua limitação de conhecimento da totalidade das conseqüências da sua atuação, sua falta de compreensão do sistema total.

Além disso, a análise acima também revela que o processo de desenvolvimento de software sofre de uma imperfeição moral. A automação de processos possui uma série de conseqüências positivas que contribuem para o melhoramento da condição humana. Porém, se o processo de desenvolvimento de software não pode ser projetado para lidar com possíveis conseqüências negativas, fica patente a imperfeição moral do seu projeto.

O processo de desenvolvimento de software encontra no modelo econômico liberal um fator de sucesso, que dirige as empresas a buscarem na automação uma forma de melhorar sua competitividade. Em função disso, um processo de desenvolvimento de software tem uma grande probabilidade de sucesso, bastando para isso que não descuide das garantias básicas para a implementação bem sucedida citadas anteriormente.

### ***Considerações finais***

Diferentemente das abordagens funcionalista, a HSC admite naturalmente a possibilidade do engano e a imperfeição dos projetos na prática. Como uma abordagem heurística, ela ajuda a descobrir aspectos importantes do problema e sua solução, e nesse processo de descoberta ela admite a inevitabilidade do engano. Sendo ao mesmo tempo uma abordagem crítica, ela permite identificar as imperfeições nos modelos pelo confronto entre projetos reais e conceitos ideais.

Uma das dificuldades da aplicação da HSC está na base conceitual que dá a consistência epistemológica ao paradigma heurístico sistêmico crítico. Essa base conceitual precisa ser compreendida para uma aplicação adequada da metodologia. Compreendidos os conceitos básicos, em especial as doze categorias heurísticas, a aplicação da HSC se torna relativamente natural.

As discrepâncias entre o que o sistema é o que o sistema deveria ser – identificadas pela aplicação das doze perguntas de fronteira – podem ser indícios de riscos potenciais que ocorrem em

função dos julgamentos de fronteira. Visto dessa forma, a HSC pode ser aplicada como uma forma de análise de riscos nos projetos.

A HSC pode alternativamente ser usada como uma forma de auditoria do processo de software. A aplicação das doze perguntas no modo “é” e no modo “deveria”, nesse caso, permite avaliar o processo e sugerir modificações. Através do emprego polêmico dos julgamentos de fronteira os auditores e os auditados podem questionar a concepção atual do processo e descobrir melhoramentos. Têm-se assim uma abordagem heurística crítica à auditoria.

Aplicada ao processo de desenvolvimento de software a HSC permite entendê-lo ou concebê-lo como um processo moralmente responsável, intrinsecamente motivado a cumprir seu propósito de contribuir para a melhoria das condições dos clientes e usuários sempre que for possível. E quando não for possível, a HSC ajuda a tornar explícitos para os clientes e usuários os julgamentos de fronteira dos envolvidos na concepção do processo. Isso permite aos clientes e usuários questionarem esses julgamentos, e discutirem sua validade em face das consequências com as quais eles próprios afinal de contas terão que conviver.

#### **Bibliografia**

BOEHM, Barry. *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, Grady. *Object-Oriented Analysis and Design with Applications*. California: Benjamin/Cummings, 1994.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. *The Unified Modeling Language User Guide*. Addison-Wesley, 1999.

COCKBURN, Alistair. *Writing Effective Use Cases*. Pittsburgh: Addison-Wesley, 2001.

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

PAULK, Mark C. et alii. *Key Practices of the Capability Maturity Model, Version 1.1*. Pittsburgh: CMU/SEI, 1993.

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

ULRICH, Werner. *Critical Heuristics of Social Planning: A New Approach to Practical Philosophy*. Bern: Haupt, 1983. Chichester: Wiley, 1994.

ULRICH, Werner. *Critical Systems Heuristics*. in: DAELLENBACH, H. G.; FLOOD, R. L. *The Informed Student Guide to Management Science*. London: Thomson Learning, 2002, p.72s.