Algumas observações sobre os vínculos entre a Engenharia de *Software* e o pensamento moderno

Cássio Adriano Nunes Teixeira

Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, Bloco H, CT, Cidade Universitária, Ilha do Fundão, Rio de Janeiro, RJ, CEP. 21.995-970

cassio@bndes.gov.br

Abstract. *Considering the modern thought, this paper intends an explanation of the technicist bias of Software Engineering (SE) and therefore of the rising recognition of the relevance of the so called "non technical factors". We suggest an enlarged framing for SE inspired by the sociotechnical approach of Science Studies (SS).*

Resumo. Tomando por base uma visão de mundo moderna, ensaiaremos uma explicação para a origem do tecnicismo existente na *Engenharia de Software* (ES) e para o reconhecimento, cada vez mais freqüente, da relevância dos chamados "fatores não técnicos" para o alcance dos objetivos dos projetos de *software*. Sugerimos uma ampliação do enquadramento tecnicista da ES a partir do enfoque sociotécnico dos *Estudos de Ciência e Tecnologia* (ECT).

1. Introdução – a saga de um SEPG

Nosso ponto de partida será a história de um SEPG[1] – *Software Engineering Process Group* – no difícil esforço de viabilizar um projeto de *melhoria de processo de software*[2] (MPS) em uma grande empresa pública brasileira.

A principal premissa do SEPG era: *envolver os desenvolvedores induzindo-os, eles próprios, a estabelecerem o processo de software*. Um *diagnóstico* (McFEELEY, 1996) indicara apropriada a abordagem CMMI[3] – *Capability Maturity Model Integration*. O SEPG julgou suficiente para realizar o projeto sua capacitação de acordo com esse modelo, porém não conseguiu materializar sua premissa. Resultaram inócuas as tentativas de motivação, envolvimento e convencimento das pessoas em favor do projeto de MPS. Fugiam do alcance do SEPG as recomendações de praxe: busca de patrocínio – existiam apenas pálidas afirmações de que processo de *software* "era importante"; obtenção do apoio de um "campeão" – não existia nenhum líder natural no grupo que pudesse ser tomado como exemplo de dedicação ao esforço de definição e uso do processo de *software*; vinculação do projeto MPS à estratégia da área de informática da empresa – a área acabara de ser reestruturada, não era claro qual seria

[1] SEPG é o grupo encarregado de guiar a implantação e utilização de um processo de *software* (CARTER, L. et al., 2002, *The Road to CMMI: results of the first technology transition workshop*. Technical Report, CMU/SEI-2002-TR-007, Software Engineering Institute, Pittsburgh – PA).

[2] Um *processo de software* é um conjunto de atividades, e seus resultados associados, que produzem um produto de *software*. Quatro atividades são fundamentais: especificação do *software*; desenvolvimento do *software*; validação do *software*; e evolução do *software* (SOMMERVILLE, 2004, p.8).

[3] CMMI é um agregado de modelos que fornecem direcionamento para o desenvolvimento e melhoria de processos, desenvolvido pelo *Software Engineering Institute*, Univ. Carnegie Mellon (CHRISSIS, 2003).

seu planejamento futuro. Além disso, a gerência sênior que criara o SEPG tinha sido destituída. O cenário de atuação do SEPG não era nada promissor.

A troca da gerência sênior fazia necessária uma "reafirmação do patrocínio" (McFEELEY, 1996, p.4). Infelizmente, a nova gerência sequer recebeu o SEPG para se inteirar do projeto de MPS, obrigando-o a recorrer ao nível gerencial imediatamente abaixo, na tentativa de negociar o escopo e a estratégia de condução do projeto. A proposta do SEPG era definir e implantar o processo de *software* de forma incremental e participativa. Instaurou-se uma enorme *controvérsia*[4], posto que um reduzido e influente grupo de gerentes de linha defendia a rápida aquisição de um conjunto de ferramentas CASE[5] – *Computer Aided Software Engineering* – e a adaptação de um processo "padrão de mercado", implantando-o através de pilotos. Além de não seguir a ordem recomendada no CMMI, tratava-se de uma abordagem de tentativa e erro, que ignorava o valor da participação ativa dos desenvolvedores e acreditava que o processo padrão, por si só, seria o garantidor dos benefícios esperados. Eram inócuos argumentos como:

> É necessário [num projeto de MPS] envolver o time ao longo de todo o processo de mudança, entendendo suas dúvidas e envolvendo-o no planejamento do novo processo [de *software*]. Tornando-os *stakeholders* no processo de mudança, é muito mais provável que eles desejarão realizar o trabalho (SOMMERVILLE, 2004, p.680). Padrões e procedimentos são a base para a gerência de qualidade, porém gerentes experientes reconhecem que existem aspectos intangíveis sobre a qualidade do *software* (...) que não podem ser corporificados nos padrões[6].

Instaurou-se um impasse que durou meses sem que argumentos "técnicos" resolvessem a situação. Não havia ainda sequer um planejamento para o projeto, pois nem o escopo estava negociado. Até que, em um determinado momento, o departamento de desenvolvimento de sistemas – o cliente do SEPG – contratou uma consultoria para ajudá-lo a estabelecer uma forma segura de utilizar-se fábrica de *software*, por determinação da gerência sênior da época. Coincidentemente, o parecer da consultoria era muito aderente à proposta de escopo e de estratégia para implantação de processo de *software* do SEPG. Associando-se a esses dois novos elementos – o parecer da consultoria e a definição de se usar fábricas de *software* –, o SEPG reiterou sua proposta para o projeto MPS com pequenas modificações. Com isso, quase um ano após ser formado, o SEPG, que não obteve sucesso através de argumentações "técnicas", finalmente conseguiu estabelecer um acordo que julgava adequado sobre o escopo e a estratégia para o projeto MPS. O trunfo do SEPG foi estabelecer o seguinte enredamento: *rejeitar o escopo e a estratégia propostos equivaleria a rejeitar a*

[4] Para os *Estudos de Ciência e Tecnologia* – ECT (tradução da denominação do campo interdisciplinar reconhecido internacionalmente como *Science and Technology Studies*), o processo de resolução das controvérsias é fundamental no estabelecimento dos fatos – científicos ou técnicos – e dos artefatos – objetos tecnológicos. Uma afirmação, por si só, não é capaz de estabelecer-se como fato, ou ser relegada à categoria de ficção. Somente o uso que os outros dela fizerem, somado a afirmações ulteriores, fará com que uma afirmação de um determinado autor consolide-se em fato ou vire ficção. Para existir como fato, uma afirmação qualquer precisa superar as controvérsias em torno de sua validade. Deve resistir às *provas de força* que surgirão. Se vencer as controvérsias, a afirmação se tornará fato e a realidade subjacente a ela será conformada. A verdade surge somente quando a controvérsia se encerra, a partir deste instante passamos a aceitar que ela estivera lá o tempo todo. Quando isto ocorre, foi produzida uma *caixa-preta* (LATOUR, 2000, p.40ss). *Prova de força* é o questionamento da validade de uma afirmação frente ao que ela representa, a explicitação daquilo que sustenta a afirmação (*ibid.*, 2000, p.129).

[5] As ferramentas CASE cobrem uma vasta gama de diferentes tipos de programas de computador usados para suportar o processo de desenvolvimento e manutenção de *softwares* (SOMMERVILLE, 2004, p.12).

[6] *Ibid.*, p. 642.

utilização segura de fábricas de software na empresa. O grupo de gerentes de linha que divergiam ficou na seguinte situação: negar a proposta do SEPG equivaleria a negar a determinação da gerência sênior de utilizar fábricas de *software*. O processo de construção dessas equivalências e enredamentos é denominado *tradução*[7], pelos *Estudos de Ciência e Tecnologia* (ECT).

Pouco tempo depois, ocorreu uma nova troca de gerência sênior, mas, naquela altura, novos elementos dificultavam o ressurgimento do impasse, como: o plano para o projeto já estava pronto e tacitamente aceito e a contratação de consultoria técnica para o projeto MPS já estava em andamento.

Poderíamos achar que a situação começava a melhorar para o SEPG, contudo, sem patrocínio, prestígio e liderança reconhecida pelo grupo de desenvolvimento, com a quase totalidade dos desenvolvedores, na ocasião, sem a menor motivação para o projeto, ocorreu mais uma troca de gerência sênior. Desta vez, toda a área de informática mudou de posição organizacional, e, conseqüentemente, passou a ser orientada por outra assessoria jurídica, a qual discordou do processo de contratação da consultoria técnica para o projeto, conforme orientado pela assessoria jurídica anterior. Resultado: mal acabara de ser planejado, o projeto MPS amargou 5 meses de atraso. O cenário só piorava!

Buscando mais aliados e novas *traduções*, o SEPG pediu apoio ao departamento de comunicação da empresa que contratou uma consultoria especializada em comunicação e gestão de mudanças para ajudá-lo. Essa consultoria entrevistou a gerência média, a gerência de linha e os desenvolvedores. Descortinou-se o já sabido: àquela altura não havia resistência ao projeto MPS porque a maioria das pessoas sequer considerava que ele continuava existindo. Vários desenvolvedores acreditavam que o SEPG em breve seria extinto, pois mais de um ano se passara e, na prática, o que havia era apenas um plano e alguns modelos de documentos oferecidos aos desenvolvedores. A consultoria esclareceu que era crucial considerar as individualidades dos desenvolvedores. É certo que deveriam ser convencidos da importância do projeto MPS para a organização, porém o SEPG deveria mostrar para cada um os *benefícios pessoais* que poderia ter. O SEPG precisava considerar valores, expectativas e anseios de cada desenvolvedor.

Iniciando uma abordagem de comunicação menos impessoal, o SEPG começou a trabalhar na direção do *reconhecimento da excelência dos profissionais da área de informática da empresa*. Experiências de utilização bem sucedidas dos modelos de documentos, há tempos disponibilizados pelo SEPG, passaram a ser melhor exploradas e divulgadas. O uso desses modelos fez surgir, tácita e espontaneamente, (partes de)

[7] Tradução é um conceito chave para os ECT, que chegam a ser denominados de *sociologia das traduções*. Tradução é um processo coletivo no qual construtores de fatos e artefatos, em movimentos de ajustes mútuos e negociações de interesses com os elementos que querem alistar, buscam equivalências e coincidências, nem sempre existentes *ab initio*, de objetivos que viabilizem a associação e socialização dos elementos para a construção e disseminação do fato ou artefato. Não se faz distinção analítica entre elementos humanos e não humanos alistados, nem delimitações nítidas entre fatores "técnicos" e fatores "sociais". O fato ou artefato produzido (conteúdo) é indissociável de seu contexto de produção. Uma *rede sociotécnica* resulta do processo de tradução. Existe uma forte característica de incerteza no processo de tradução, pois, *a priori*, não se pode garantir que os elementos alistados agirão como esperado ou mesmo se permanecerão enredados. Maiores detalhes em Bruno Latour (2000) ou Michel Callon (1999).

processos que ao mesmo tempo derivavam-se do *reconhecimento da excelência dos profissionais* e serviam para robustecê-lo.

Finalizando esta história, somente em março de 2005, quase dois anos após a criação do SEPG, o processo de *software* começou a ser definido e discutido com o departamento de desenvolvimento de sistemas. Chama muito a atenção que, o tempo todo, o SEPG se envolveu em batalhas nas arenas política, social, cultural e organizacional. Em momento algum, mesmo decorrido bastante tempo, questões "técnicas" ameaçaram o projeto de MPS, tais como o desconhecimento das práticas do CMMI ou o não domínio de alguma ferramenta de suporte. Mas, de modo geral, se explorarmos a literatura sobre *Engenharia de Software* (ES) veremos muito mais ostensivamente abordadas as chamadas questões "técnicas", restando apenas alertas sobre a "importância" de questões como patrocínio, motivação, liderança, dentre outras, como antídoto para dificuldades semelhantes às enfrentadas pelo SEPG. Porém, o tratamento destas que são qualificadas como questões "não técnicas" fica relegado a outras disciplinas, contribuindo para *naturalizar* a "aura tecnicista" da ES.

A motivação neste texto é ensaiar, na seção 2, uma possível explicação para o enquadramento tecnicista limitante da ES e argumentar que, no transbordar desse enquadramento, residem questões que apontam para a importância das "questões não técnicas" nos projetos. Na seção 3, é proposta uma ampliação desse enquadramento a partir da desconsideração de fronteiras rígidas entre o técnico e o social e do reconhecimento da complexidade sociotécnica do objeto da ES. Um breve comentário, na seção 4, dá conta das dificuldades na aplicação de modelos ou padrões universais. Por último, na seção 5, tem-se uma provocação para a necessidade de enriquecermos o perfil do engenheiro de *software*.

2. Vínculos com o pensamento moderno

A comunidade acadêmica e prática de Engenharia de *Software* (ES) reconhece, cada vez mais, a necessidade de abordarmos os assim chamados "aspectos não-técnicos", ou as "questões sociais", que vêm ganhando importância na literatura sobre desenvolvimento de sistemas (ARNOLD, 2003, p.228). Paradoxalmente, no entanto, a mesma literatura que vem enfatizando a importância dessas questões, na consecução dos objetivos da ES, reserva pouquíssimo espaço para tratar ostensivamente como devem ser abordadas. Alfonso Fuggetta (2000) alerta que a maioria das indicações sugeridas pelo CMM foca em aspectos de engenharia[8] somente, isto encerra riscos de ignorarmos questões que podem ser críticas para os projetos de melhoria de processo de *software*. Watts Humphrey também parece reconhecer o tecnicismo preponderante quando diz:

> Avanços oriundos de praticantes (*user driven*) (...) gastam muito tempo para penetrar na academia. Estes assuntos não envolvem, geralmente, um *excitante material técnico*, ao contrário, focalizam questões mais pragmáticas (...) (HUMPHREY, 2002, p.70, grifo nosso).

O status de engenharia conferido ao desenvolvimento de *software* faz com que o vejamos, ordinariamente, como disciplina técnica (HANSETH, 1998, p.6). Parece *natural* que a ES se dedique, sobremaneira, às ferramentas, métodos, modelos e princípios técnicos. A própria denominação pela negativa – "não técnico" – revela que

[8] CMM – *Capability Maturit Model* – é um precursor do CMMI. O autor, Fuggetta, utiliza a palavra *engenharia* na acepção de algo "puramente técnico", exatamente no sentido ora problematizado.

os problemas "técnicos" têm *status* mais importante e prioritário para a ES. Vejamos uma possível explicação para este tecnicismo dominante na ES.

2.1. Herança da modernidade

Com Descartes, a racionalidade científica ocidental passou a valorizar o método como diretriz para a verdade. Isso se entranhou na matemática e nas ciências naturais, que, de sua parte, influenciaram o estabelecimento daquilo que conta como conhecimento no ocidente (HIRSCHHEIM, 1995, p.21). A construção de modelos universalizantes, representativos de sistemas ordenados, é uma das idéias básicas do pensamento moderno e, sob esse estatuto, achamos natural que o desenvolvimento de sistemas de *software* se ampare nas idéias de representação, formalização, ordem, controle, sofrendo profunda influência do racionalismo, da metáfora mecanicista e da valorização do método (DAHLBOM, 1993, pp.12-14). Assim, o conceito de *software* como um produto de engenharia, como um objeto matemático, encontrou livre curso nos primeiros 50 anos da história do desenvolvimento de *software* (LEVESON, 1997). De fato, a maioria da literatura e pesquisa tende a focalizar apenas os "aspectos técnicos" de desenvolvimento e implementação dos sistemas de *software* (WARNE, 2003).

Alguns dos elementos da visão moderna – mecanicista e positivista – do mundo são facilmente resgatados na ES. Os chamados métodos estruturados (ou *hard methods*) de projeto de sistemas, por exemplo, se auto-identificam como objetivos e científicos (ARNOLD, 2003, p. 228). O imperativo moderno da representação e formalização está explícito nas inúmeras linguagens, métodos e ferramentas. A perspectiva do controle jaz na disciplina dos *processos de software*, herdeiros das idéias de burocracia – que considera possível prever, predeterminar e explicitar papéis; estabelecer regras que garantam o comportamento esperado e a coordenação central necessária. Nos projetos de *software*, normalmente parte-se do pressuposto que é possível saber antecipadamente o que deve ser feito, existindo pouca incerteza acerca das tarefas (DAHLBOM, 1993, p.16).

Vejamos a *modelagem de informação*[9], ferramenta chave com presença muito forte nos ciclos de desenvolvimento de sistemas. Ela denota "uma posição realista ingênua", que pressupõe bastar existir um método adequado para que o mundo real, objetivo, possa ser descoberto. Mais ainda, se descoberto através desse método, tal mundo seria consistente, de complexidade gerenciável (HANSETH, 1998, p.141) e, portanto, controlável. A pressuposição subjacente é que existe um mundo ordenado *a priori*, bastando ao engenheiro de *software* descobrir / capturar os requisitos preexistentes, formalizar uma especificação e desenvolver o sistema desejado a partir dela. A maioria das abordagens tende a considerar que é possível, de antemão, definir os requisitos e que eles se manterão estáveis ao longo do desenvolvimento[10].

> [Segundo uma visão realista, os componentes da modelagem do sistema] – funções, dados ou objetos, – existem no mundo real. O trabalho do desenvolvedor seria o de encontrar estes elementos que comporão o sistema, 'como se fossem um tesouro afundado' (HIRSCHHEIM, 1995, p.xi-xii).

[9] Uma introdução à *modelagem da informação* pode ser obtida em: COAD, P., YOURDON, E. *Object Oriented Analysis*. Englewood Cliffs, New Jersey: Yourdon Press, 1990.
[10] *Ibid.*, p.83.

Um exemplo dessa posição realista pode ser visto na reflexão de Tom DeMarco, acerca de suas importantes idéias sobre a *análise estruturada*:

> É uma importante *verdade*: quando você está atacando a complexidade através de particionamento, quanto mais tênues as interfaces melhor o particionamento – se as interfaces ainda estão grosseiras, exageradas, volte e particione novamente, buscando os contornos *naturais do domínio* (DeMARCO, 2002, p.526, grifo nosso).

De posse da representação, os processos de desenvolvimento assumem uma estrutura hierárquica, com uma forte crença no poder do projetista deter todo o controle da situação (HANSETH, 1998, p.9). Deixam de considerar o mundo real e passam a focalizar apenas a representação do sistema (DAHLBOM, 1993, p.51).

Por estar estruturada sobre essas idéias mecanicistas / positivistas, a ES se reveste de tecnicismos e devota grande importância à busca por padrões e modelos, influência da própria busca da ciência por modelos, padrões, fatos e leis universais (HANSETH, 1998, p.133). Ser científico é ser universal, valer em qualquer lugar, facultar a qualquer um a repetição dos fenômenos observados, desde que respeitado o método. Assim, modelos e métodos universais adequados garantiriam, *per si*, o sucesso dos projetos. Podemos ver esse ponto de vista nas palavras de DeMarco (2002, p.524).

> Eles [seus leitores] estavam convencidos com [a validade de] o método porque ele dava uma confortável sensação de completute; ele aparecia para eles como A Resposta para todos os problemas. Quando não conseguiam resolver seus problemas eles culpavam a si próprios e tentavam com maior rigor ainda. Hoje acredito que meu livro de 1975[11] foi excessivamente persuasivo e que muitos em nossa indústria [de *software*] foram simplesmente seduzidos por ele. Isto em parte resultou de meu irrestrito entusiasmo com um método que funcionou soberbamente para mim (num domínio limitado) (...).

A comunidade de ES tem reconhecido e alertado sobre as limitações dessa visão:

> (...) Não devemos desqualificar automaticamente, como 'não' científico [portanto, como não útil] aqueles esforços não baseados em evidências estatísticas e experimentos controlados. (...) Algumas das contribuições mais importantes para a ciência da computação não foram baseadas em estudos empíricos (...). Parnas verificou estatisticamente que a adoção do ocultamento de informações tem correlação positiva com a qualidade do *software* desenvolvido?[12] Certamente Parnas fez sua afirmação com base em sua *profunda e madura experiência*. Mas a relevância de sua *intuição* foi comprovada por observações quantitativas. Como uma provocação, eu afirmo que pelos critérios de avaliação de hoje, o trabalho dele não seria considerado *cientificamente válido* (FUGGETTA, 2000, grifo nosso).

2.2. Limitações da visão mecanicista positivista

> Quando tentamos controlar o mundo com programas de computador ou métodos para o desenvolvimento de sistemas, não podemos nos esquecer que a visão mecanicista tem por premissa que o mundo que tentamos entender e controlar é, por si próprio, um mundo ordenado, um sistema fundamentalmente imutável (DAHLBOM, 1993, p. 15).

Essa premissa mecanicista enquadra os tipos de problemas considerados e o instrumental para abordá-los. É o que denota o quadro mais interno da Figura 1 que destaca alguns pressupostos modernos. Trata-se do enquadramento tradicional da ES, com sua clássica visão sistêmica explicitada na denominação *análise de sistemas*.

[11] DeMARCO, T., 1975, *Structured Analysis and System Specification*. Prentice Hall.

[12] O autor se refere ao texto seminal sobre *information hiding* de David Lorge Parnas: PARNAS, D.L., 1972, "On the Criteria to Be Used in Decomposing Systems into Modules". In Communications of the ACM, v. 15, n. 12, pp.1053-1058.

Uma solução mecanicista tenderá a ser bem sucedida quando for coerente com os princípios delimitados pelo quadro mais interno da Figura 1. Tomemos, como exemplo, o avanço das linguagens de programação. Com suas interfaces gráficas e ambientes integrados de suporte ao desenvolvimento, os compiladores atuais estão muito à frente dos primeiros montadores (*assemblers*) criados. Certamente, neste caso, o problema existente pôde ser enquadrado com sucesso na visão mecanicista, pois, dada a formalização, a ordem, a estabilidade e a possibilidade de controle subjacentes, comandos numa linguagem de programação de alto-nível puderam ser "facilmente" traduzidos em comandos equivalentes nos níveis de abstração mais baixos.

Sob a influência moderna, a ES buscou extirpar "contaminantes sociais" tentando manter-se como disciplina "puramente técnica". A maioria das abordagens considera que projeto e desenvolvimento de *software* tratam-se apenas de uma questão técnica, quando muito, com conseqüências sociais. Isto conformou a ES em uma disciplina que crê lidar sobretudo com objetos de complexidade "técnica", demandando somente, e cada vez mais, ferramentas, métodos, modelos e princípios técnicos (HIRSCHHEIM, 1995, p.1), ficando "fora" da ES várias questões (Figura 1).

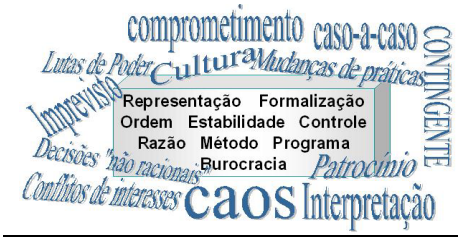

Figura 1– O enquadramento mecanicista / positivista da ES, e seus transbordamentos "sociais, culturais, políticos"

Um enquadramento pressupõe seu transbordamento. Na Figura 1, notamos que enquadrando premissas mecanicistas, ficam de fora, transbordam, questões como conflitos de interesses, lutas de poder, imprevistos, cultura, decisões não racionais, mudanças de práticas. Tendemos a deixar fora do enquadramento tudo aquilo que produz incertezas, tal como o comportamento das pessoas, tido como incerto e pouco previsível. Nancy Leveson (1997) diz que a ES, como outras engenharias, buscou automatizar e retirar o elemento humano, o elo mais sujeito a erros, do ciclo de controle e execução das atividades. Os problemas começam a aparecer quando tentamos aplicar a visão mecanicista fora do enquadramento que a suporta. Fuggetta corrobora-o quando cita o exemplo notoriamente bem sucedido das ferramentas de suporte à *gerência de configuração* cujo sucesso advém da efetividade com que abordam tarefas extremamente maçantes e repetitivas, *razoáveis de serem automatizadas*. E questiona: será que não estaríamos tentando modelar e automatizar coisas que intrinsecamente não podem ser modeladas e automatizadas no ciclo de vida de desenvolvimento de *software*? (FUGGETTA, 2000, grifo do autor).

Configura-se um dilema da ES com seu viés tecnicista. Ao mesmo tempo que capitalizou grande desenvolvimento em várias áreas, a ES se vê crescentemente pressionada a lidar com elementos que escapam, transbordam, de seu enquadramento: os chamados "aspectos não-técnicos". Ao enfrentar problemas num mundo cada vez mais dinâmico, com demanda por respostas sempre mais rápidas, com imprevisibilidade e mutações aceleradas, a ES começa a perceber as limitações que se lhe impõe o enquadramento moderno, mecanicista / positivista, e a reconhecer a necessidade de lidar com o transbordo desse enquadramento. A maior causa das falhas de sistemas – abandono de projetos, sistemas construídos e nunca utilizados, estouro de prazos e orçamentos – em grande maioria, senão na totalidade, não é devida à falta/carência de

ferramentas e técnicas, mas sim à negligência dos "fatores humanos" nas práticas dominantes de análise e projeto de sistemas (DU PLOOY, 2003, p.43).

Por isso, atualmente, é possível observar uma diminuição da ênfase no "aspecto de engenharia" na ES. Um dos primeiros a alertar sobre a importância dos "aspectos humanos" envolvidos com o desenvolvimento de *software* foi Frederick Brooks no clássico de 1975: *The Mythical Man-Month*, reeditado em 1995[13]. Brooks reconhece a complexidade social envolvida no desenvolvimento de *software* e a natureza única deste processo. A diminuição da ênfase na engenharia pode ser explicada parcialmente pela impossibilidade de desenvolvimento de um processo, orientado a engenharia, que respondesse a todos os problemas típicos dos projetos de *software*. Abordagens mais recentes entendem a ES como disciplina multifacetada, devotando mais atenção aos "aspectos humanos" (TOMAYKO, 2004, p. 116).

3. Sem divisões: o enfoque sociotécnico

Herdeiros da modernidade, tendemos a reconhecer, separadamente, elementos "sociais" e "técnicos" nos fatos científicos e artefatos tecnológicos. Tal separação não é apropriada. O enfoque sociotécnico defende ser descabido esperar que uma ciência ou tecnologia seja "pura", livre de "contaminações sociais". Para existirem, fatos e artefatos demandam uma adequada associação/socialização de elementos humanos e não humanos em *redes sociotécnicas*. É possível abdicar das várias divisões, inerentemente convencionais, entre ciências naturais e ciências sociais e defender a impossibilidade de dissociar o "natural" ou "técnico" do "social" ou "cultural". Fatos e artefatos não são isentos de seus contextos, ou melhor, constituem-se imbricados de ciência, tecnologia e sociedade. Para lograr êxito em sua produção deve-se conseguir arregimentar e socializar aliados diversos. *Hipertrofiar o "lado técnico" simultaneamente significará hipertrofiar o "lado social"* (LATOUR, 2000), o que impossibilita divisar uma fronteira rígida entre técnico e social.

Este artigo argumenta favoravelmente à ampliação do enquadramento atual da ES. Mas não pelo caminho aparentemente trilhado até aqui, que, ao dar voz a alguns autores, parece admitir a fatoração da ES em "aspectos técnicos" de um lado e "aspectos humanos, sociais ou não-técnicos" de outro. Não se trata de ampliar o enquadramento, conservando a idéia moderna de que grandes divisões em fatores ou aspectos são possíveis, como ilustrado na Figura 2, que mantém os princípios mecanicistas ainda isolados.

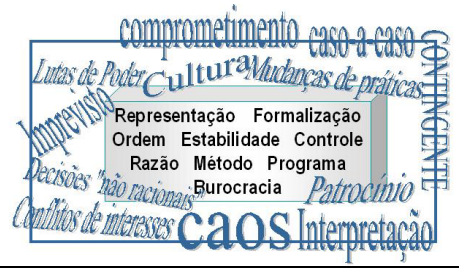

Figura 2 – Enquadramento ampliado, ainda dividindo: "técnicos" x "não-técnicos"

Segundo o enfoque sociotécnico, não devemos reconhecer "fatores não-técnicos" como externos. Devemos ampliar o enquadramento através da desconstrução da divisão em fatores, reconhecendo que o técnico está presente no social tanto quanto o

[13] BROOKS F.P., 1995, *The Mythical Man-Month*: *essays on software engineering.* Addison Wesley.

social está presente no técnico: indissociavelmente; sociotecnicamente. A Figura 3 ilustra um enquadramento mais amplo, sem divisões *a priori*.

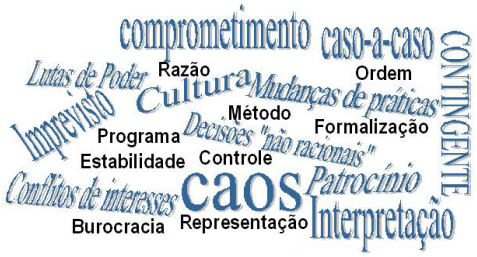

Figura 3 – Enquadramento ampliado, sem divisões entre "técnico" e "não-técnico"

Visitando a história da ES, podemos resgatar muitos casos que demonstram a dificuldade da separação clara entre o técnico e o social. Por exemplo, a denominação *engenharia de software*, termo hoje em dia naturalizado, que transparece uma suposta inerência técnica ao desenvolvimento de *software*. Quando foi utilizado pela primeira vez, em 1968 numa conferência do comitê de ciência da OTAN – Organização do Tratado do Atlântico Norte, o termo engenharia foi escolhido porque poderia *provocar interesse* (PARNAS, 1997, grifo nosso). Não se tratava, portanto, primordialmente, de denotar que desenvolvimento de *software* era de fato e naturalmente uma engenharia. A utilização do termo "engenharia" naquela conferência teve conotação social – atrair atenção – ou técnica – infundir novo rigor ao desenvolvimento de *software*?

A criação da abordagem de Orientação a Objetos remonta ao ano de 1962 quando a linguagem destinada à simulação – SIMULA – estava sendo projetada. Seus autores, Ole-Johan Dahl e Kristen Nygaard, ao desenvolvê-la, utilizando idéias já existentes, perceberam ser muito útil, para o problema em que estavam envolvidos, criar um conceito que associasse estrutura de dados e operações – os objetos. Um dos motivos que levou os autores de SIMULA a se basearem na linguagem ALGOL 60 para desenvolvê-la foi o "patriotismo europeu" (DAHL, 2002, p.80). Escolha técnica?

Como última provocação, vejamos o que diz Michael Fagan sobre a dificuldade de estabelecermos de forma exclusivamente "técnica", limites de uma especificação de requisitos:

> (...) Os requisitos terminam quando param de ser anunciados, quando há um produto que pode ser testado ou uma característica definida, ou quando o projeto (*design*), a próxima operação [do ciclo de desenvolvimento], aceita o documento [especificação de requisitos] como uma entrada válida e começa o trabalho? Isso realmente *esta sujeito a uma grande possibilidade de interpretação* e, muito freqüentemente, *é mais uma questão de tempo* (cronograma) do que uma questão de critérios [técnicos] (FAGAN, 2002, p. 566, grifo nosso).

Ao ilustrar a dificuldade de separar claramente o técnico do social na história e prática da ES, é possível perceber com mais nitidez seu viés tecnicista. O argumento de que várias "questões" devem ser deixadas para outras disciplinas externas à ES, por serem "não-técnicas", fica enfraquecido, pois a pressuposição de existência de uma realidade técnica isolada de uma outra realidade social pode ser contestada. Hoje em dia, é cada vez mais familiar a premissa conceitual que interpreta os SI como sistemas sociotécnicos (ARNOLD, 2003).

Reconhecendo a complexidade sociotécnica do objeto da ES, fica mais fácil aceitar que gerir um projeto de SI, desde sua concepção até sua implantação, pode ser visto como gerir o interesse das pessoas (METCALFE, 2003), um processo de negociação sociotécnica, um desafio de *tradução*[7]. Entender projetação (*design*) como esforço de *tradução* é bem diferente da confortadora, embora ilusória, premissa tradicional onde se crê que o projetista detém, *a priori*, o controle total do processo.

Mesmo assim, historicamente, os projetos têm sido vistos apenas como a gerenciável tarefa de especificar (e seguir) padrões técnicos, deixando de lado a grande e incontrolável tarefa de alinhar inovações, tecnologias, cultura, políticas, condições de mercado e organizacionais (HANSETH, 1998, pp.8-9).

A visão sociotécnica de não dividir *a priori* a complexidade do objeto da ES em aspectos técnicos e não técnicos, ou humanos e não humanos, facilita o entendimento e o tratamento dos problemas enfrentados. Pois isso encoraja-nos detalhar a percepção e a descrição dos mecanismos concretos que viabilizam manter coesas as redes sociotécnicas que garantem a existência e o sucesso de *softwares*, de processos de *softwares*, de métodos, técnicas e tecnologias de desenvolvimento[14].

4. Padrões e modelos universais

Nesta seção, permitiremos um breve comentário sobre padrões e modelos "universais" tão procurados pela ES em decorrência de seus vínculos com a modernidade.

É preciso perceber que um padrão ou modelo quando ganha ares de universal oculta o processo de negociação sociotécnica que viabilizou sua existência. Retornando ao exemplo do SEPG (seção 39), suponhamos que a execução bem sucedida do projeto de MPS resultasse em melhorias na qualidade dos produtos. Um observador que não tivesse acompanhado o esforço desde o início tenderia a aceitar que esse resultado decorre apenas da excelência técnica do modelo utilizado – o CMMI, no caso. A abordagem sociotécnica auxilia esse observador tardio, esclarecendo-o que se tivesse acompanhado o estabelecimento do processo de *software* enquanto perduravam as *controvérsias*[15], perceberia inadequado, entre outros: atribuir seu sucesso a qualquer padrão universal; julgá-lo "puramente técnico"; ou, ainda, reputar exclusivamente a algum "campeão", patrocinador, ou líder o êxito de estabelecê-lo. O observador perceberia que o processo estabelecido seria fruto de sua rede sociotécnica e das inevitáveis *traduções*[16] envolvidas.

Usualmente padrões e modelos focalizam o problema sob um viés tecnicista, desprezando questões culturais, sociais e políticas. Contudo, quando a implantação de um padrão não é bem sucedida, essas questões, muito freqüentemente, surgem como explicação para o fracasso, mantendo a aura de infalibilidade "técnica" do padrão. Apresentado sem sua história e contexto, um padrão universal reclama a possibilidade de replicar a presumida competência que encerra. Porém, por si só, nenhum padrão pode garantir a repetição de um suposto sucesso obtido em alguma situação; em verdade, ele replica apenas a si mesmo sob a alegação de replicar a presumida competência que encerra. O CMMI, por exemplo, que valor teria se utilizado apenas em uma única organização? Um padrão universal, de fato, nunca será utilizado, pois sempre será necessário elaborar seu significado local, um problema caso-a-caso, empírico, como denotam as afirmações:

> Melhoria de processo não significa simplesmente adotar métodos e ferramentas particulares ou usar algum modelo de processo utilizado em outro lugar. (...) existem sempre fatores, procedimentos e padrões *locais* que *influenciam* o processo. Raramente será bem sucedido na introdução de melhorias no processo aquele que simplesmente tentar mudar o processo [local]

[14] Adaptado de HANSETH, 1998, p. 97.
[15] *Vide* nota 4.
[16] *Vide* nota 7.

> para algum outro utilizado em outro lugar. Sempre deve-se olhar para *melhoria de processo como algo específico para a organização* (SOMMERVILLE, 2004, p.666, grifo nosso).
>
> Embora as *áreas de processos* descrevam comportamentos que devem ser exibidos em qualquer organização, práticas devem ser interpretadas usando um profundo conhecimento de CMMI, da(s) disciplina(s), *da organização, do ambiente de negócios e das circunstâncias específicas envolvidas*. (CHRISSIS, 2003, p.97, grifo nosso).

Elaborar as relações entre o universal e o local é matéria típica de projetação (HANSETH, 1998, p.135). Na implantação de CMMI, por exemplo, o desafio não é apenas implantar um modelo universal, mas sim de projetar relações, papéis e habilidades que se esperam dos desenvolvedores. Dominar as prescrições do CMMI talvez seja a parte simples do esforço, porquanto a grande tarefa é construir localmente uma instância desse modelo. Padrões ou modelos utilizados são apenas mais um elemento com o qual se negocia para a construção da rede sociotécnica demandada. A verdadeira dificuldade subjaz no transbordo das prescrições dos modelos universais.

5. Considerações Finais

Por premissa, este artigo considerou que discutir as limitações que o paradigma tecnicista moderno nos impõe é um passo inicial, válido, rumo à elaboração de novas propostas práticas e concretas para a ES. Existe, porém, a possibilidade de tais propostas terem formatos e abordagens muito diferentes das que estamos acostumados, exatamente por serem exteriores ao paradigma que contestam. Não basta ao engenheiro de *software* saber sobre informação, computação, linguagens e processos para resolver os desafios que enfrenta na prática (DAHLBOM, 1993, p. 47). A "saga" do SEPG serve para ilustrá-lo. À medida que parece necessário reconhecer a importância da ES lidar com o transbordo de seu enquadramento atual, também parece razoável crer na necessidade de enriquecermos o perfil de formação do engenheiro de *software* (TEIXEIRA, 2005), que deve ser instrumentado para criar redes sociotécnicas; proceder a projetação local de padrões universais ao invés de simplesmente "dominar" tais padrões; reconhecer e lidar com realidades onde questões humanas, sociais e políticas não são externas ao problema, mas, ao contrário, compõem, com ferramentas, métodos, modelos e princípios técnicos, um mesmo tecido sem costura.

Referências

ARNOLD, M., 2003, "Systems Design Meets Habermans, Foucault and Latour". In: CLARKE, S., et. al (eds.), *Socio-Technical and Human Cognition Elements of Information Systems*. London, Information Science Publishing, pp.226-248.

CALLON, M., 1999, "Some Elements of a Sociology of Translation: domestication of the scallops and the fishermen of St. Brieuc bay". In: BIAGIOLI, M., *The science studies reader*, New York, Routledge, pp.67-83.

CHRISSIS, M., KONRAD, M., SHRUM, S., 2003, *CMMI: Guidelines for Process Integration and Product Improvement*. Addison-Wesley.

DAHLBOM, B., MATHIASSEN, L., 1993, *Computers in Context: The Philosophy and Practice of Systems Design*. Oxford, NCC Blackwell.

DAHL, O.-J., 2002, "The Roots of Object Orientation: the SIMULA language". In: BROY, M., DENERT, E. (eds), *Software Pioneers: contributions to software engineering*, Berlin, Germany, Springer-Verlag, pp.79-90.

DeMARCO, T., 2002, "Structured Analysis: Beginnings of a New Discipline". In: BROY, M., DENERT, E. (eds), *Software Pioneers: contributions to software engineering*, Berlin, Germany, Springer-Verlag, pp.521-527.

FAGAN, M., 2002, "A History of Software Inspections". In: BROY, M., DENERT, E. (eds), *Software Pioneers: contributions to software engineering*, Berlin, Germany, Springer-Verlag, pp.563-573.

FUGGETTA, A., 2000, "Software Process: A Roadmap". In: FINKELSTEIN, A. (ed.), *The Future of Software Engineering*.

HANSETH, O., MONTEIRO, E., 1998, *Understanding Information Infrastructure*. Manuscript. Disponível em <http://heim.ifi.uio.no/~oleha/Publications/book.pdf>. Acesso em: 01 abr. 2005.

HUMPHREY, W.S., 2002, "Three Process Perspectives: Organizations, Teams, and People". In: Annals of Software Engineering, 14, pp.39-72.

HIRSCHHEIM, R, HEINZ, K.K., LYYTINEN, K., 1995, *Information Systems Development and Data Modeling: Conceptual and Philosophical Foundations*. Cambridge University Press.

LATOUR, B., 2000, *Ciência em Ação: como seguir cientistas e engenheiros sociedade afora*. São Paulo, Editora UNESP.

LEVESON, N.G., 1997, "Software Engineering: Stretching the Limits of Complexity". In: Communications of the ACM, v. 40, n. 2, pp.129-131.

McFEELEY, B., 1996, IDEAL: *A User's Guide for Software Process Improvement*. Handbook, CMU/SEI/96-HB-001, Software Engineering Institute, Pittsburgh – PA.

METCALFE, M., 2003, "Concern Solving for IS Development" In: CLARKE, S., et. al (eds.), *Socio-Technical and Human Cognition Elements of Information Systems*. London, Information Science Publishing, pp.135-151.

PARNAS, D. L., 1997, "Software Engineering: An Unconsummated Marriage". In: Communications of the ACM, v. 40, n. 9, p.128.

PLOOY, N.F. du, 2003, "The Social Responsibility of Information Systems Developers". In: CLARKE, S., et. al (eds.), *Socio-Technical and Human Cognition Elements of Information Systems*. London, Information Science Publishing, pp.41-59.

SOMMERVILLE, I., 2004, *Software engineering*. 7th ed., Addson-Wesley.

TEIXEIRA, C.A.N., CUKIERMAN, H., 2005, Apontamentos para Enriquecer o Perfil do Engenheiro de *Software*. In Congresso da Sociedade Brasileira de Computação, 25., 2005, São Leopoldo/RS. Anais Eletrônicos... São Leopoldo/RS: Unisinos. Disponível em <http://www.unisinos.br/_diversos/congresso/sbc2005/_dados/anais/pdf/arq0026.pdf >. Acesso em: 02 mar. 2006.

TOMAYKO, J., HAZZAN, O., 2004, *Human Aspects of Software engineering*. Hingham, Charles River Media, Inc.

WARNE, L., 2003, "Conflicts and Politics and Information Systems Failure: A Challenge for Information Systems Professionals and Researchers". In: CLARKE, S., et. al (eds.), *Socio-Technical and Human Cognition Elements of Information Systems*. London, Information Science Publishing, pp.104-134.