

## Modernidade, universalismo e assimetrias

Cássio Adriano Nunes Teixeira, Henrique Luiz Cukierman

Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ  
Bloco H, CT, Cidade Universitária, Ilha do Fundão, Rio de Janeiro, 21995-970, RJ, Brasil

{cant, hcukier}@cos.ufrj.br

### Resumo

Este artigo convida a uma reflexão sobre a Engenharia de Software (ES) no que diz respeito à sua crença na difusão dos modelos ditos “universais”. No entanto, tal convite pode ser estendido até mesmo para a reflexão acerca da grande assimetria entre os países do Primeiro Mundo, que centralizam o desenvolvimento científico e tecnológico, portanto dos modelos “universais”, e os países periféricos, aparentemente fadados ao papel de consumidores de tais modelos. Para tanto, partiu-se da inspiração advinda de um caso pontual onde a contraposição da história da informática de uma estatal brasileira com a história da ES norte-americana revela uma grande coincidência de problemas e soluções ao longo dos anos 1970, 1980 e 1990, porém defasados por 20 anos.

PALAVRAS-CHAVE: abordagem sociotécnica, engenharia de software.

### Abstract

*Modernity, universalism and asymmetries.* This paper is an invitation to reflect about Software Engineering (SE) in what concerns the widely known “universal” models. However, such an invitation can be extended to include a reflection about the great asymmetry between the First World, which centralizes the scientific and technological development, that is, the SE “universal” models, and peripheral countries, apparently granted the role of consumers of such models. To make this invitation attractive, the paper is based on one specific Brazilian case – that of a public corporation – because its problems and solutions, as recognized and proposed by local SE along the 1970s, 1980s and 1990s, repeat the same problems and solutions as identified by North-American SE 20 years before.

KEY WORDS: sociotechnical approach, software engineering.

## 1 Introdução

É preciso reconhecer a Engenharia de *Software* (ES) como uma construção, bem como reconhecer o *estilo de pensamento*<sup>1</sup> que fundamenta esta construção, assim “desnaturalizando” a idéia de que a ES é como é em decorrência da “ordem natural das coisas”. A ES se estrutura

em torno da busca de padrões e modelos “universais” que, acredita-se, podem ser *difundidos* para replicar as “melhores práticas”, das quais, supostamente, são os porta-vozes. Problemas com a difusão dos modelos têm, muitas vezes, respostas aparentemente pré-elaboradas, via de regra imputando-se aos chamados “fatores não-técnicos” a culpa pela maioria dos malogros dos projetos de *software*.

---

<sup>1</sup> Ludwik Fleck define o estilo de pensamento não como um tom particular dos conceitos ou uma forma peculiar de reuni-los. “Trata-se de uma coerção determinada de pensamento e mais ainda: a totalidade da preparação e disponibilidade intelectual orientada a ver e atuar de uma forma e não de outra. A dependência de qualquer fato científico [e tecnológico] ao estilo de pensamento é evidente” (Fleck, 1986, p.111).

Neste artigo, traçaremos uma breve recapitulação, na seção 2, sobre os vínculos entre a ES e a modernidade. Na seção 3, a aparente coincidência entre a história norte-americana da ES, como apresentada por Barry Boehm em “A View of 20th and 21st Century Software Engineering” (Boehm, 2006), e a história da informática do BNDES – Banco Nacional de Desenvolvimento Econômico e Social – como apresentada em “Um olhar sociotécnico sobre a engenharia de software: o caso do BNDES” (Teixeira, 2007). A questão que se coloca com as histórias aparentemente coincidentes é a defasagem de 20 anos entre a história do BNDES e a norte-americana da ES. Trata-se de uma insinuação preliminar e, ao mesmo tempo, de uma sugestão para trabalhos futuros, que poderiam responder questões do tipo: deixar de pensar no engenheiro de software como um *tradutor* e continuar pensando nele como um *difusor*<sup>2</sup> de padrões e modelos “universais” contribui para a manutenção da defasagem temporal da prática da ES nas empresas brasileiras? Em seguida, na seção 4, apresentamos nossas considerações finais.

Cabe lembrar que, no que se segue, não levamos em consideração o modelo MPS.Br, o qual, sem dúvida, poderia ser tratado como parte das possíveis alternativas frente à problematização proposta pelo artigo, cujo principal objetivo, todavia, é somente evidenciar as assimetrias “silenciosas” envolvidas nas soluções “universalistas”, assimetrias essas assumidas de forma acrítica, para não dizer despercebida, por boa parte dos engenheiros de software brasileiros. Também é de se considerar a questão que subjaz ao artigo (ainda que não explicitada), a da adoção de modelos fortemente estruturados e pré-concebidos em contraposição à adoção de soluções caso a caso, sendo estas últimas uma possível alternativa mais eficiente para alguns exemplos da realidade brasileira, ao menos os das micro, pequenas e médias empresas, o que igualmente validaria outra possível reflexão a respeito das metodologias ágeis. Porém, reiteramos que nenhuma das duas discussões, a do MPS.Br e a das metodologias ágeis, pertencem ao escopo do artigo, mas situam-se sim nas possibilidades de desdobramentos que o artigo suscita.

Vale também tecer um breve comentário quanto à falta de explicitação neste artigo de alternativas positivas frente à realidade de desenvolvimento de software no Brasil. Em verdade, o artigo procura somente mostrar os problemas da alternativa adotada pelo BNDES, a saber,

a da imitação tardia de modelos estrangeiros. Quanto a possíveis alternativas, procuramos tratá-las em outro artigo, intitulado “Um Olhar Sociotécnico sobre a Engenharia de Software”<sup>3</sup>, de nossa autoria em conjunto com Rafael Prikladnicki, recém publicado na revista *RITA* (Cukierman *et al.*, 2007).

## 2 Vínculos da ES com a modernidade<sup>4</sup>

O status de engenharia conferido ao desenvolvimento de *software* faz com que o vejamos, ordinariamente, como disciplina técnica (Hanseth e Monteiro, 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” – da vasta gama de questões indissociáveis da prática da ES como as questões culturais, organizacionais e econômicas, por exemplo, revela que os problemas “técnicos” têm *status* mais importante e prioritário para a ES. Uma possível explicação para este tecnicismo dominante na ES pode ser extraída de sua busca por um *pedigree* científico ao se estruturar como disciplina autônoma. Em tal busca, a rede sociotécnica da ES inevitavelmente se entrelaçou à rede da modernidade – o *estilo de pensamento* dominante – com sua visão mecanicista e tecnocêntrica do mundo (Teixeira e Cukierman, 2006).

A busca de cientificidade pela ES decorre, dentre outras coisas, das próprias circunstâncias do surgimento de computadores e de seus programas no meio científico, bem como da finalidade a que se destinavam. Howard Aiken, físico da universidade de Harvard, responsável pelo projeto de um dos primeiros computadores que existiu, portanto um profundo conhecedor da tecnologia da computação da época, declarou equivocadamente, em 1956:

*Se algum dia ocorrer da lógica básica de uma máquina projetada para solucionar numericamente equações diferenciais coincidir com a lógica de uma máquina cujo propósito é gerar faturas para uma loja de departamentos, eu iria julgar isto a mais surpreendente coincidência que já tenha encontrado (Dahlbom e Mathiassen, 1993, p. 3).*

Naquela época, parecia natural enxergar a computação e sua programação como claros domínios da matemática

<sup>2</sup> Veja Teixeira e Cukierman (2007) para um resumo dos modelos de tradução e difusão de fatos e artefatos, conforme definido por Bruno Latour (2000).

<sup>3</sup> O artigo procura condensar as reflexões que temos apresentado ao WOSSES desde a edição de 2006.

<sup>4</sup> Um detalhamento maior desta questão já foi apresentado nos artigos: *Algumas observações sobre os vínculos entre a Engenharia de Software e o pensamento moderno* (Teixeira e Cukierman, 2006) e *Por que falham os projetos de implantação de processos de software?* (Teixeira e Cukierman, 2007).

tica e ciência, pois os computadores eram extensivamente usados apenas para cálculos de engenharia e de pesquisas científicas. Somente de maneira muito restrita eram usados para o processamento de dados em atividades de negócios (Teixeira, 2007).

Ao buscar a chancela científica, a ES revela sua tentativa de compartilhar “a verdade detida pela ciência ocidental”, cujos grandes feitos técnicos e tecnológicos – desde a descoberta das Américas até a última viagem espacial – conferiram à modernidade a reputação de via segura para o progresso, e, por isso mesmo, consolidada como o *estilo de pensamento* dominante no ocidente.

Alguns dos elementos da visão moderna do mundo – mecanicista e positivista – 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 assim como estabelecer regras que garantam o comportamento esperado e a coordenação central necessária. Nos projetos de *software*, normalmente parte-se do pressuposto de que é possível saber antecipadamente o que deve ser feito, existindo pouca incerteza acerca das tarefas (Dahlbom e Mathiassen, 1993, p. 16).

A *modelagem de informação*<sup>5</sup>, por exemplo, ferramenta chave com presença muito forte nos ciclos de desenvolvimento de sistemas, 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 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 (Hanseth e Monteiro, 1998, p. 83, 141).

*[Segundo uma visão realista, os componentes da modelagem do sistema] – funções, dados ou objetos – existem no mundo real. O trabalho do desenvol-*

*vedor seria o de encontrar estes elementos que comporão o sistema, ‘como se fossem um tesouro afundado’ (Hirschheim et al., 1995, p. xi-xii).*

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 em deter todo o controle da situação (Hanseth e Monteiro, 1998, p. 9). Deixam de considerar o mundo real e passam a focalizar apenas a representação do sistema (Dahlbom e Mathiassen, 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 e Monteiro, 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 também 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 completude; 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<sup>6</sup> foi excessivamente persuasivo e que muitos em nossa indústria [de software] foram simplesmente seduzidos por ele. Isso em parte resultou de meu irrestrito entusiasmo com um método que funcionou soberbamente para mim (num domínio limitado) [...].*

<sup>5</sup> Uma introdução à modelagem da informação pode ser obtida em Coad e Yourdon (1990).

<sup>6</sup> O livro em questão é *Structured Analysis and System Specification* (DeMarco, 1975).

Com a valorização dos padrões e modelos “universais”, os projetos de *software*, sejam eles desenvolvimento/implantação de sistemas ou melhoria/implantação de *processos de software*, são guiados sob uma visão que divide, nitidamente, o que seria o conteúdo técnico (leia-se: o modelo “universal” a ser implantado) daquilo que seria o contexto social/organizacional de implantação. “Fatores não-técnicos”, exteriores ao “conteúdo” (ou seja, fora dos enquadramentos propiciados pelo modelo “universal”), são reconhecidos como determinantes no sucesso dos projetos e, por isso, deve estar garantido que preexistam em uma configuração apropriada no “contexto” (Teixeira e Cukierman, 2007). Ou seja, padrões e modelos focalizam o problema sob um viés tecnicista, desprezando questões culturais, sociais e políticas, além de demarcar uma separação entre “conteúdo” e “contexto”. Mas, no entanto, 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 – do “conteúdo técnico”. Apresentado sem sua história e contexto, um padrão “universal” reclama a possibilidade de replicar a presumida competência que nele se 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? Na verdade, um padrão “universal” nunca será utilizado, de fato, pois sempre será necessário elaborar seu significado local, um problema caso-a-caso, empírico, como denotam as afirmações (Teixeira e Cukierman, 2006):

*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] 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 et al., 2003, p. 97, grifo nosso).*

É preciso perceber que um padrão ou modelo quando ganha ares de universal oculta o processo de negociação sociotécnica que viabilizou sua existência. A perspectiva sociotécnica, quando esclarece que contexto e conteúdo são indissociáveis, instrumentaliza a percepção de que ao implantar um “conteúdo técnico” implanta-se também seu “contexto” de produção. Conteúdo e contexto são reconstruídos localmente nos esforços de implantação de modelos “universais”, de sorte que implantar, digamos, o CMMI, é tentar implantar também o contexto, por exemplo, de medições, levantamentos, estatísticas e formalizações de norte-americanos. Para termos uma idéia a esse respeito, poderíamos refletir sobre como brasileiros e norte-americanos reagem à formalização. Nos EUA, seria admissível uma pergunta, por vezes usual entre nós brasileiros: “será que a lei ‘tal’ vai pegar?” (Teixeira, 2007).

Nesta seção apresentamos, muito brevemente, o enquadramento modernista da ES com seu claro viés de busca por modelos e padrões “universais” que possam ser *difundidos*. Nas histórias que virão na seção seguinte, é possível perceber, ainda que superficialmente, essa característica da ES. Antes da ES estar estruturada como disciplina autônoma, sínteses locais surgem para dar conta da complexidade da tarefa de desenvolvimento de *software*. É o que ocorreu nos projetos de defesa norte-americanos nos anos 1950 e no BNDES nos anos 1970. À medida que a ES vai se identificando como uma disciplina, passa a existir um movimento de “universalização” das práticas locais e de sua incorporação aos modelos e padrões, de modo que essas práticas supostamente poderiam deixar os “contextos” em que surgiram e serem aplicadas em quaisquer outros “contextos”. Isso também pode ser percebido nas histórias a seguir, que citam o surgimento dos métodos estruturados e dos modelos de maturidade nos EUA e sua tentativa de utilização em outra localidade específica, no caso, no “contexto” do BNDES.

### 3 Histórias paralelas

Aqui mostraremos a aparente coincidência, ainda que com 20 anos de defasagem, entre a história norte-americana e a história do BNDES daquilo que hoje denominamos ES, conforme descrito nos trabalhos: “*A View of 20<sup>th</sup> and 21<sup>st</sup> Century Software Engineering*” (Boehm, 2006) e “Um olhar sociotécnico sobre a engenharia de *software*: o caso do BNDES” (Teixeira, 2007, capítulos 4 e 5).

O início da programação de computadores em larga escala, num momento em que sequer existia o termo ES, data dos anos 50 nos EUA. Dificilmente depreenderemos o estado atual da ES se não conhecermos seu passado. A ES como um produto da tecnociência do século 20 faz parte de uma “máquina de guerra” (Latour, 2000, p. 282).



Seu desenvolvimento, tal qual o da própria computação, está associado aos bilionários projetos de defesa norte-americanos durante a Guerra Fria — os “BIG-L”. Uma resposta dos EUA à ameaçadora existência de armas atômicas aerotransportadas por bombardeiros soviéticos. Precursor e padrão para vários outros, o projeto 416L da Força Aérea criou o sistema SAGE — *Semi-Automatic Ground Environment* — nos anos 1950.

Naquele cenário de desenvolvimento inicial, ainda não havia sistemas de *software* (o termo *software* também ainda não existia). Era um cenário comumente referenciado como “campo verde”<sup>7</sup>, no qual a racionalidade dominante era a de maximização da eficiência de utilização dos extremamente caros e limitados recursos de *hardware*. A própria engenharia do *hardware* influía diretamente na maneira de encarar o desenvolvimento de *software*, sugerindo um processo que enfatizava revisões e verificações exaustivas das especificações antes do processo de produção<sup>8</sup>. Efetivamente, o SAGE impulsionou o desenvolvimento da disciplina de gerenciamento de projetos de *software*. Técnicas de gerenciamento já maduras na engenharia foram adaptadas em uma seqüência de estágios que partia do estabelecimento inicial dos conceitos e seguia até o artefato final de programação. Esse esquema hierárquico, guiado por especificações, foi largamente difundido na indústria de *software* no final dos anos 1950, por milhares de programadores que o vivenciaram no projeto SAGE, formador de toda uma geração de profissionais altamente qualificados para o desenvolvimento de *software* militar de missão crítica (Campbell-Kelly, 2004, p. 67-69). O entendimento de que o SAGE formou uma geração de profissionais permite perceber a inexistência de programadores naquele momento em que praticamente nascia a computação eletrônica. Assim, esses profissionais foram formados na prática (*on the job*) em um ambiente organizacional (militar) e tecnológico (*hardware*) altamente padronizados.

Já no BNDES dos anos 1970, o que se via? Com um *mainframe* IBM centralizando todo o processamento de dados, via-se uma instalação de computação cuja ope-

ração era algo complexo, envolvendo diversas categorias de profissionais, tais como digitadores, preparadores de lotes, operadores, programadores e analistas de sistemas (à época com atuações bem distintas). Esses profissionais traziam uma formação muito aderente ao negócio de informática, pois virtualmente todos tinham uma formação prática nas empresas — tanto nas fornecedoras (IBM, Burroughs, etc.) quanto nos birôs de prestação de serviço<sup>9</sup>. Nesse cenário, os profissionais dependiam uns dos outros para a consecução de quase todas as tarefas que demandavam. Por exemplo, para a simples compilação de um programa, o programador dependeria, pelo menos, do perfurador de cartões e do operador. Configuravam-se relações que viabilizavam a existência/aceitação de hierarquias e pontos de controle explícitos, favorecendo a aceitação de uma disciplina de padronização e métodos de trabalho estruturados hierarquicamente.

O custo de manutenção e operação da instalação de computador, além das próprias restrições tecnológicas existentes à época, justificavam a racionalidade de otimização do uso dos recursos computacionais, mesmo em detrimento de algumas necessidades dos usuários. Também cabe lembrar que inexistiam no BNDES, nos anos 1970, sistemas de informática, isto é, configurava-se também uma situação de “terra virgem”.

Comparando os dois cenários, temos nos EUA dos anos 1950 a engenharia do *hardware* como inspiração primária para os processos de desenvolvimento de *software*. A palavra chave é o uso eficiente do caro e limitado *hardware*. Programadores são treinados na prática em projetos como o SAGE. Tem-se uma ambiência com grande respaldo metodológico, tecnologia favorecendo a padronização e um forte controle hierárquico e centralizado dos projetos, guiados por especificações formalizadas, em um cenário de “terra virgem”. Quadro semelhante o BNDES viveu nos anos 1970, quando a otimização do uso do *hardware* tinha peso preponderante, os profissionais eram formados nas empresas, a instalação de informática era centralizada no CPD, os processos eram formalizados e controlados hierarquicamente, num ambiente também

<sup>7</sup> “Campo verde” ou “terra virgem” é um lugar-comum, não só no jargão de informática, que serve para figurar uma área, ou situação, ainda não cultivada, tratada, construída — *greenfield land* —, em contraste a uma outra situação — *brownfield land* — onde algo já foi cultivado, tratado, construído, podendo até conter os escombros do que ali teria existido.

<sup>8</sup> A influência da engenharia do *hardware* é compreensível. Além dos primeiros programadores serem os próprios engenheiros que construía os computadores, a maioria dos cursos de computação, quando começaram a existir, surgiu nos departamentos de engenharia e, por isso, tendiam a ser mais direcionados para a máquina do que para a produção de *software*. A forte orientação da engenharia do *hardware* é presença manifesta nas principais sociedades de profissionais de *software*, como se pode entrever nos nomes, por exemplo, da *Association for Computing Machinery* e da *IEEE Computer Society* (Boehm, 2006, grifos do autor).

<sup>9</sup> Nos anos 1970, apenas começavam a existir os primeiros cursos na área de computação no Brasil. Atualmente, os profissionais formados nos cursos de graduação, na maioria dos casos portadores de grande carga teórica, trazem uma formação mais distanciada das necessidades imediatas do mercado.

de “terra virgem”, posto que se iniciava a construção dos primeiros sistemas.

Em meados dos anos 1960, nos EUA, o poder de processamento e armazenagem de dados dos computadores havia crescido vertiginosamente e seu custo relativo havia diminuído muito. Em decorrência disso, passaram a existir um expressivo número de instalações civis de computadores, nos EUA, e uma explosiva demanda por *software*. Na década anterior, como praxe de mercado, os fabricantes de computadores disponibilizavam “sem custo” para as organizações usuárias, além do *software* básico para a utilização dos computadores, vários aplicativos que suportavam atividades dos maiores ramos de negócio, como o setor bancário, de seguro, industrial e de varejo. O custo do desenvolvimento desses aplicativos era considerado uma despesa de *marketing* pelos fabricantes, uma condição necessária para a venda de *hardware*, este sim seu verdadeiro negócio à época. No entanto, a partir de meados dos anos 60, o custo relativo do *software* começou a se tornar mais expressivo do que o custo do *hardware* e o fator preponderante nos custos, à medida que também crescia explosivamente a demanda por programadores, passou a ser a mão-de-obra (Campbell-Kelly, 2004, p. 89-98). Paul Edwards (1997, p. 247-249) relaciona a falta de programadores com o desenvolvimento das linguagens de programação de alto nível. Os primeiros programadores eram basicamente os matemáticos e engenheiros que projetavam e construíam os computadores. Para eles, prevalecia a estética matemática de concisão como norma de elegância, ainda mais sob a injunção dos escassos recursos das máquinas. Linguagens de programação de alto nível, facilmente apreendidas por não especialistas, eram necessárias para viabilizar os projetos militares, como o SAGE, e a própria indústria da computação.

Com as linguagens de programação de alto-nível e novos programadores sem a disciplina do rígido formalismo matemático, começou a ser questionado, como inspiração para o desenvolvimento de *software*, o exemplo do processo de produção de *hardware* que, como dissemos, enfatizava a revisão e verificação exaustivas das especificações antes do processo de produção. Os artefatos de *software*, aparentemente muito mais fáceis de alterar, permitiam uma abordagem de “tentativa e erro” (*code*

*and fix*), além do que produzir especificações precisas de *software* era (e ainda é) muitíssimo mais complicado do que para *hardware*. Tem-se a era do “*software crafting*” que junto à postura libertária dos anos 1960, questionadora do regime centralizado de autoridade, usualmente fazia com que os programadores seguissem métodos próprios em detrimento dos métodos que suas empresas tentavam utilizar. Surgia a figura dos programadores *cowboys*, os profundos conhecedores dos sistemas e os únicos capazes de, misteriosamente, nos momentos decisivos, reparar os defeitos e sanar as falhas. “Código espaguete” em profusão começou a ser gerado, trocando o aparente sucesso dos projetos da década anterior pelos altos custos com retrabalho e casos de fracasso (Boehm, 2006). A expressão *crise do software*<sup>10</sup>, cunhada na famosa conferência da OTAN de 1968 (Naur e Randell, 1969), sintetiza bem o que ocorria.

Vinte anos depois, no BNDES dos anos 1980, passou a ser questionado o controle centralizado da informática exercido pelo DESIS – Departamento de Sistemas. O DESIS tinha praticamente a mesma organização desde quando fora criado, nos anos 1970, uma instalação típica de CPD (Centro de Processamento de Dados) baseada em *mainframe* com processamento *batch*<sup>11</sup>. No entanto, uma vez atendida a demanda, nos anos 70, pelos sistemas que compunham o chamado *back office* – sistemas administrativos, financeiros e contábeis – passou a existir uma grande pressão por sistemas e serviços de informática que atendessem às diversas áreas de negócio do BNDES. Por não conseguir dar conta de toda essa demanda, o DESIS passou a ser visto com uma imagem pejorativa de burocracia e lentidão.

Nesse mesmo momento em que a imagem do DESIS piorava por não conseguir atender tempestivamente à demanda que explodira, entram em cena os microcomputadores e, com eles, novos aplicativos, novas linguagens de programação e, principalmente, muitos *novos programadores*. Diferentemente do *mainframe* que parecia impor diversos elementos de padronização, controle e metodologias, o microcomputador, ao alcance de todos, favoreceu a desestabilização da estrutura organizacional da informática do BNDES. Houve uma descentralização da informática, de modo que passaram a existir núcleos nas áreas de negócio onde os próprios profissionais des-

<sup>10</sup> Com o rápido aumento do poder dos computadores e da complexidade dos problemas que podiam ser atacados, basicamente, o termo *crise do software* (ou *gap*) refletia a crença na dificuldade de se escrever, com correção, softwares inteligíveis e passíveis de validação precisa, em decorrência do crescimento de sua complexidade ser muito mais acelerado que o progresso da “engenharia de software”. Malgrado os grandes avanços, as demandas estariam além da capacidade, teoria, técnicas e métodos da época, o que supostamente traria um sombrio cenário futuro para a indústria de software e para a própria sociedade, cada vez mais dependente de artefatos de software (Naur e Randell, 1969, p.17, 121).

<sup>11</sup> A filosofia dos sistemas de processamento em lotes – *batch* – surgiu em decorrência dos altos custos dos primeiros sistemas computacionais e consistia em agrupar todas as tarefas, com seus respectivos recursos demandados, em lotes que, uma vez submetidos ao processamento, ocupassem a máquina ininterruptamente.

sas áreas desenvolviam os aplicativos que necessitavam à revelia de padrões, métodos e do controle centralizado do DESIS. Esses novos programadores passavam a ser conhecidos como os “donos” dos respectivos sistemas e, a exemplo dos citados programadores *cowboys*, dada a falta total de documentação, só eles conseguiam manter os aplicativos que desenvolviam.

Então, 20 anos após a explosão da demanda por *software* nos EUA dos anos 1960, quando vivenciamos a era da programação por tentativa e erro (*code and fix*) – uma era de desenvolvimento de *software* insubordinado ao controle de processos rígidos – e a entrada de muitos novos programadores em cena por conta das linguagens de programação de alto-nível, o BNDES dos anos 1980 viveu um movimento bastante análogo.

Em resposta aos problemas decorrentes das práticas de desenvolvimento de *software* dos anos 1960 (*software crafting*) nos EUA, sobretudo os elevados custos com retrabalho, a comunidade da então nascente disciplina de ES desenvolveu, nos anos 1970, os *métodos estruturados*. No embalo do sucesso da programação estruturada, buscou-se estender a “idéia de estruturação” para a projeção (*design*) dos sistemas de *software*. Basicamente, propunha-se uma criteriosa análise de requisitos prévia ao esforço de desenvolvimento, de sorte que o desenvolvimento propriamente dito pudesse ser guiado pelos requisitos preestabelecidos. Em busca de uma maior formalização para o processo de desenvolvimento, Winston Royce também propôs nessa mesma época o popular modelo cascata de desenvolvimento de *software*. Em resumo, buscavam-se meios de depender menos das habilidades dos profissionais e garantir planejamento e controle mais efetivos dos projetos.

Voltando ao BNDES de 20 anos mais tarde, isto é, focalizando os anos 1990, vemos a materialização das conseqüências do “*software crafting*” ocorrido com a descentralização da informática nos anos 1980 e com o desenvolvimento de aplicativos de maneira bastante descontrolada. A falta de controle tornou-se uma preocupação nos anos 1990 e o antídoto para o problema parecia ser a implantação de uma metodologia baseada na análise estruturada e de uma abordagem mais formal para o gerenciamento e controle dos projetos de desenvolvimento de sistemas. Numa tentativa não muito bem sucedida de *difundir* os modelos “universais” da ES propostos nos anos 1970, foram criadas no BNDES, em 1993, a MEDES – Metodologia de Desenvolvimento de Sistemas– e a MPP – Metodologia de Planejamento de Projetos.

Encurtando o final dessas histórias coincidentes, porém “rigorosamente” afastadas por um intervalo de tempo de 20 anos, temos que nos anos 1980, nos EUA, decorrente da não conformidade freqüente dos sistemas desenvolvidos com o ciclo cascata e devido à sobrecarga

imposta pelos métodos estruturados e processos formalizados de desenvolvimento, surgem os padrões de qualidade e os modelos de maturidade, bem como a proposta de estruturação do trabalho em fábricas de *software*. Exatamente 20 anos depois, o BNDES, sofrendo de problemas análogos, deliberou nos anos 2000 o uso de fábricas de *software* e a implementação do CMMI – *Capability Maturity Model Integration*.

#### 4 Conclusão – Assimetrias

Evidentemente, a defasagem tecnológica de 20 anos entre a ES brasileira e a norte-americana, conforme explorada até aqui através do caso pontual do BNDES, nada mais é que uma insinuação. Uma insinuação deve-se ser problemática, cabe reconhecê-lo, uma vez que pode confundir-se com a idéia de que se trata de uma regra, quando está longe de sê-lo. Até porque não é razoável pensar em uma defasagem temporal estanque, quando na verdade o que se tem observado é o entrelaçar de distintas temporalidades, a convivência de diferentes estados-da-arte da tecnociência. Se no caso aqui analisado adotam-se tecnologias com duas décadas de retardo, não será difícil encontrar, quiçá na mesma instituição, outras áreas nas quais se está perfeitamente alinhado à chamada fronteira do conhecimento. Enfim, trata-se de apenas um caso, cuja suposta generalização reclama não somente muito mais casos que possam suportá-la como também uma exploração mais detida da entrada em cena, no quadro temporal analisado, do recente fenômeno da globalização, amparado por novas dinâmicas *difusionistas*, que não necessariamente amparadas pela lógica da defasagem temporal.

Porém, trata-se, a nosso ver, de insinuação “inspirada”, na medida em que parece repetir o padrão encontrado em outros setores tecnológicos. O padrão diz respeito aos investimentos em tecnologia realizados pelo Primeiro Mundo e sua necessária amortização, na qual relevam-se as negociações (quase sempre unilaterais) com os países periféricos, chamados a consumir esta ou aquela tecnologia de acordo com um cardápio que procura privilegiar os cálculos econômicos dos países centrais. Por tais cálculos entenda-se, de forma muito simplificada, a previsão e o controle (realizados evidentemente nos países centrais) de como devem ser distribuídos os custos de investimento em novas tecnologias, ao mesmo tempo em que têm de ser amortizados os investimentos nas velhas tecnologias. Assim, arrolam-se locais e momentos que deverão consumir as velhas tecnologias para financiar as novas, ou então para consumir de imediato as novíssimas (e, como apontamos acima, para se estar em linha com a fronteira do conhecimento) e assim contribuir para a aceleração de investimentos em larga escala. Quanto a este último caso,

vale citar, a título de exemplo, a dissertação de mestrado de Wagner do Carmo, na qual discute a adoção no Brasil da NGN (*Next Generation Network*), uma tecnologia que busca a convergência dos meios de comunicação – dado e voz – em uma mesma plataforma, colocada para as operadoras locais de telecomunicações como uma opção “necessária”. Um dos pontos de sua análise é que

*antes de tudo, [é necessário] estar atento às contingências locais e mostrar que nada é transferido sem se transformar. A rede NGN surge em um momento propício para os países do Primeiro Mundo e em um momento ingrato para os do Terceiro Mundo. Este lado ingrato representa uma rede não convergente que ainda não foi amortizada [diferente da situação do Primeiro Mundo] (Carmo, 2005, p. 111, grifos nossos).*

É por conta dessa contabilidade altamente centralizada que são produzidas as assimetrias entre países centrais e periféricos, entre “consolidados” e “emergentes”, cuja conseqüência é a distribuição de forma desigual da tão sonhada modernidade. Somos modernos “*ma non troppo*”, ou dito de outra forma, ocupamos um lugar no *ranking* da modernidade subrepticiamente pré-determinado por tal contabilidade. Pior ainda: o entendimento do que é ser moderno se faz a partir de indicadores constituídos justo por esta contabilidade desfavorável aos países periféricos, através dos quais forçosamente despontaremos sempre “em atraso” ou mesmo “em melhora” (os jornalistas de economia costumam referir-se à situação de melhora segundo a avaliação de que “estamos fazendo o dever de casa”, sem que se saiba mais exatamente à casa de quem se referem), mas sempre sem nenhuma autonomia para construirmos localmente uma outra contabilidade e, portanto, uma outra ES.

Ousaríamos dizer que a insinuação que propomos não só serve como convocação para uma futura investigação, mas que, em verdade, mais que uma insinuação, constitui-se como uma provocação necessária para que pensemos a respeito da importação e adaptação incontinente, sem maiores ponderações e avaliações locais, dos modelos que vêm de fora, os quais a modernidade procura salvar sob a aura da “universalidade”.

## Referências

- ARNOLD, M. 2003. Systems Design Meets Habermans, Foucault and Latour. In: E. COAKES; M.G. HUNTER; A. WENN; S. CLARKE (eds.), *Socio-Technical and Human Cognition Elements of Information Systems*. Londres, Information Science Publishing, p. 226-248.
- BOEHM, B. 2006. A View of 20<sup>th</sup> and 21<sup>st</sup> Century Software Engineering. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE), XXVIII, Shangai, China, 2006. *Anais...* Shangai, China, p. 12-29.
- CAMPBELL-KELLY, M. 2004. *From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry*. Massachusetts, MIT Press, 388 p.
- CARMO, W.R. do. 2005. *NGN – Uma análise sociotécnica da convergência das telecomunicações no Brasil*. Rio de Janeiro, RJ. Dissertação de Mestrado. COPPE/UFRJ, 123 p.
- CHRISISS, M.; KONRAD, M.; SHRUM, S. 2003. *CMMI: Guidelines for Process Integration and Product Improvement*. Pensilvânia, Addison-Wesley, 688 p.
- COAD, P.; YOURDON, E. 1990. *Object Oriented Analysis*. Englewood Cliffs, Yourdon Press, 233 p.
- CUKIERMAN, H.L.; TEIXEIRA, C.A.N.; PRIKLADNICKI, R. 2007. Um olhar sociotécnico sobre a Engenharia de software. *Revista de Informática Teórica e Aplicada*, XIV:199-219.
- DAHLBOM, B.; MATHIASSEN, L. 1993. *Computers in Context: The Philosophy and Practice of Systems Design*. Oxford, NCC Blackwell, 320 p.
- DeMARCO, T. 1975. *Structured Analysis and System Specification*. New Jersey, Prentice Hall, 352 p.
- DeMARCO, T. 2002. Structured Analysis: Beginnings of a New Discipline. In: M. BROY; E. DENERT (eds.), *Software Pioneers: contributions to software engineering*. Berlin, Springer-Verlag, p. 521-527.
- EDWARDS, P.N. 1997. *The Closed World: computers and the politics of discourse in Cold War America*. Massachusetts, MIT Press, 462 p.
- FLECK, L. 1986. *La génesis y el desarrollo de un hecho científico*. Madri, Alianza, 200 p.
- HANSETH, O.; MONTEIRO, E. 1998. Understanding Information Infrastructure. Manuscript. Disponível em: <http://heim.ifi.uio.no/~oleha/Publications/book.pdf>. Acessado em: 01/04/2005.
- HIRSCHHEIM, R.; KLEIN, H.K.; LYYTINEN, K. 1995. *Information Systems Development and Data Modeling: Conceptual and Philosophical Foundations*. Cambridge, Cambridge University Press, 303 p.
- LATOUR, B. 2000. *Ciência em Ação: como seguir cientistas e engenheiros sociedade afora*. São Paulo, Editora UNESP, 439 p.
- NAUR, P.; RANDELL, B. (eds.). 1969. *Software Engineering – Report on a Conference Sponsored by the NATO Science Committee*. Bruxelas, Scientific Affairs Division. Disponível em: <http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF>. Acessado em: 20/07/2007.
- SOMMERVILLE, I. 2004. *Software engineering*. 7<sup>th</sup> ed., Harlow, Addison-Wesley, 784 p.
- TEIXEIRA, C.A.N.; CUKIERMAN, H. 2006. Algumas observações sobre os vínculos entre a Engenharia de Software e o pensamento moderno. In: WORKSHOP UM OLHAR SOCIOTÉCNICO SOBRE A ENGENHARIA DE SOFTWARE, 2, Vila Velha, 2006. Disponível em: <http://www.cos.ufrj.br/~handrade/woses/woses2006/anais.htm>. Acessado em: 17/04/2007.
- TEIXEIRA, C.A.N.; CUKIERMAN, H. 2007. Por que Falham os Projetos de Implantação de Processos de Software? In: WORKSHOP UM OLHAR SOCIOTÉCNICO SOBRE A ENGENHARIA DE SOFTWARE, 3, Porto de Galinhas, 2007. Disponível em: <http://www.cos.ufrj.br/~handrade/woses/woses2007/anais.htm>. Acessado em: 01/07/2007.
- TEIXEIRA, C.A.N. 2007. *Um olhar sociotécnico sobre a engenharia de software: o caso do BNDES*. Rio de Janeiro, RJ. Dissertação de Mestrado. COPPE/UFRJ, 168 p.

Submitted on July 18, 2008.

Accepted on September 8, 2008.