Modernidade, universalismo e assimetrias[1]

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

*cassio@bndes.gov.br; **hcukier@cos.ufrj.br

Resumo

O artigo apresenta um convite à reflexão sobre a assimetria entre países do Primeiro Mundo e países periféricos, estes aparentemente fadados ao papel de consumidores do desenvolvimento científico e tecnológico daqueles, com anos de atraso. Toma-se como ponto de partida a Engenharia de Software, caracterizando-a como uma construção moderna, uma disciplina que escora-se no pressuposto da difusão de modelos e padrões «universais», em cujo pressuposto subjaz a idéia de separação entre «contexto» e «conteúdo».

1. Introdução

É preciso reconhecer a Engenharia de Software (ES) como uma construção, bem como reconhecer o *estilo de pensamento*[2] que a fundamenta, «desnaturalizando» assim 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

[1] Uma versão deste artigo foi publicada na revista Scientia - Interdisciplinary Studies in Computer Science 19(2): 94-101, July/December 2008.

[2] Ludwik Fleck define o estilo de pensamento não como um tom particular dos conceitos ou uma forma peculiar de reuní-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 artefato tecnológico] ao estilo de pensamento é evidente.» (FLECK, 1986, p.111).

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.

Neste artigo, traçaremos uma breve recapitulação, na seção 2, sobre os vínculos entre a ES e a modernidade, caracterizando-a como uma disciplina que escora-se no pressuposto da *difusão* de modelos e padrões «universais», pressuposto no qual subjaz a idéia de separação entre «contexto» e «conteúdo». Os modelos e padrões «universais» seriam «conteúdos técnicos» que, apartados de suas condições de produção, reclamam para si uma competência «intrínseca», «universalmente» válida em quaisquer outros «contextos». Na seção 3, abordando um exemplo empírico, baseados em entrevistas, observação direta e participação ativa, os autores retratam a história da informática de uma grande empresa estatal brasileira, a partir dos anos 1970, apresentada em «Um olhar sociotécnico sobre a engenharia de software: o caso do BNDES[3]» (TEIXEIRA, 2007). Ao contrapor esta história à própria história da Engenharia de Software norte-americana, desde os anos 1950 até 1980, como apresentada por Barry Boehm em «*A View of 20th and 21st Century Software Engineering*» (BOEHM, 2006), revela-se uma grande coincidência de problemas e soluções, porém com uma defasagem de 20 anos. Embora cientes da impossibilidade de quaisquer generalizações a partir da narrativa de um único caso, os autores qualificam o argumento -acerca da coincidência e defasagem entre as histórias do BNDES e da ES norte-americana- como uma insinuação inspiradora para o debate das assimetrias entre países produtores e consumidores de tecnologia, na medida em que o caso parece repetir-se em diversos outros setores. Tais assimetrias referem-se aos investimentos em tecnologia em que os países periféricos são chamados a consumir esta ou aquela tecnologia, de acordo com um cardápio que procura privilegiar os cálculos econômicos dos países centrais. Trata-se de uma insinuação preliminar e, ao mesmo tempo, de uma sugestão para trabalhos futuros, que poderiam responder questões do tipo: insistir no engenheiro de software como um *difusor*[4] de padrões e modelos «universais», em vez de pensá-lo como um *tradutor/mediador*, não constituiria um dos principais motivos para a manutenção da defasagem temporal da prática da ES nas empresas brasileiras? Na seção 4, apresentamos nossas considerações finais.

[3] BNDES - Banco Nacional de Desenvolvimento Econômico e Social - é um grande banco estatal financiador da indústria brasileira.

[4] Veja Teixeira e Cukierman (2007) para um resumo dos modelos de tradução e difusão de fatos e artefatos, conforme definido por Bruno Latour (2000).

2. Vínculos da ES com a modernidade[5]

O status de engenharia conferido ao desenvolvimento de software faz com que o vejamos, ordinariamente, como disciplina técnica (HANSETH; MONTEIRO, 1998, p.6). Parece *natural* que a ES se dedique, sobremaneira, às ferramentas, métodos, modelos e princípios técnicos. 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 tecnocêntrica do mundo. (TEIXEIRA; CUKIERMAN, 2006).

A busca de cientificidade pela ES decorre, dentre outras razões, 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, 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 julgá-lo a mais surpreendente coincidência (DAHLBOM; MATHIASSEN, 1993, p.3).*

Naquela época, parecia natural enxergar a computação e sua programação como claros domínios da matemática e da ciência, pois os computadores eram extensivamente utilizados apenas para cálculos de engenharia e de pesquisas científicas. Somente de forma 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 -desde a descoberta das Américas até a mais recente viagem espacial- conferiram à modernidade a reputação de via segura para o progresso, e, por isso mesmo, consolidado como o *estilo de pensamento* dominante no ocidente.

Alguns dos elementos da visão moderna do mundo são facilmente resgatados

[5] Um detalhamento maior desta questão pode ser visto em: «Algumas observações sobre os vínculos entre a Engenharia de Software e o pensamento moderno» (TEIXEIRA; CUKIERMAN, 2006) e «Por que Falham os Projetos de Implantação de Processos de Software?» (TEIXEIRA; CUKIERMAN, 2007).

na ES. Os chamados métodos estruturados (ou *hard methods*) de projeto de sistemas, por exemplo, se auto-identificam como objetivos e científicos. Os imperativos modernos da representação e da formalização estão explícitos nas inúmeras linguagens, métodos e ferramentas. A perspectiva do controle subjaz à disciplina dos *processos de software*, a qual 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, parte-se do pressuposto que é possível saber antecipadamente o que deve ser feito, existindo pouca incerteza acerca das tarefas (DAHLBOM; MATHIASSEN, 1993, p.16).

A *modelagem de informação*[6], por exemplo, ferramenta chave com presença muito forte nos ciclos de desenvolvimento de sistemas, denota um realismo ingênuo, por pressupor que a existência de um método adequado seja o bastante para que o mundo real, objetivo, possa ser «descoberto». Mais ainda, se descoberto através desse método, tal mundo seria consistente, de complexidade gerenciável (HANSETH; MONTEIRO, 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.[7]

> *[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; HEINZ; LYYTINEN, 1995, p.xi-xii).

Um exemplo desse realismo pode ser encontrado na reflexão de Tom DeMarco, acerca de suas importantes idéias sobre a *análise estruturada* e sua «naturalidade»:

> *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, grifos nossos).*

[6] Uma introdução à modelagem da informação pode ser obtida em: COAD, P., YOURDON, E. Object Oriented Analysis. EnglewoodCliffs, New Jersey: Yourdon Press, 1990.

[7] Ibid., p.83.

Subjaz a uma tal proposição que a ES busca revestir-se de tecnicismos, devotando grande importância à busca por padrões e modelos, influência da própria busca da ciência por modelos, padrões, fatos e leis «universais» (HANSETH; MONTEIRO, 1998, p.133). Ser científico equivale a ser universal, a 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 constatar este ponto de vista mais uma vez 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*[8] *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) [...].*

Com a valorização dos padrões e modelos «universais», os projetos de software, sejam eles de desenvolvimento/implantação de sistemas ou de melhoria/implantação de *processos de software*, são guiados por 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; 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 uma situação determinada; em verdade, ele replica apenas a si mesmo sob a alegação de

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

replicar a presumida competência que encerra. Na verdade, um padrão «universal» nunca será utilizado, de fato, pois será sempre necessário elaborar seu significado localmente (TEIXEIRA; CUKIERMAN, 2006), constituí-lo como um problema caso-a-caso, empiricamente, 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. [...] Sempre deve-se olhar para melhoria de processo como algo específico para a organização (SOMMERVILLE, 2004, p.666, grifos nossos).*
>
> *Embora as áreas de processos descrevam comportamentos que devem ser exibidos em qualquer organização, práticas devem ser interpretadas usando um profundo conhecimento de CMMI[9], da(s) disciplina(s), da organização, do ambiente de negócios e das circunstâncias específicas envolvidas. (CHRISSIS, 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, ao esclarecer 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, como: «será que a lei 'tal' vai pegar»? (TEIXEIRA, 2007).

Nesta seção apresentamos o enquadramento modernista da ES com seu claro viés de busca por modelos e padrões «universais» que possam ser *difundidos*.[10]

---

[9] CMMI -Capability Maturity Model Integration- 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).

[10] A posição que aqui adotamos não é necessariamente contrária à adoção de modelos de desenvolvimento de software, mas certamente o é quanto à sua «universalidade», o que implica em pugnar por modelos, se for possível construí-los, que encontrem ressonância com os desafios sempre locais do desenvolvimento de software.

Esse viés pode ser percebido, ainda que superficialmente, nas histórias que virão a seguir. Antes da ES estar estruturada como atividade autônoma, sínteses locais surgem para dar conta da complexidade da tarefa de desenvolvimento de software. É o que ocorreu nos projetos de defesa norte-americano 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» originais para serem aplicadas em quaisquer outros. É o que também se pode perceber nas histórias a seguir, que citam o surgimento de modelos da ES e a tentativa de sua utilização em outra localidade específica, no caso, no «contexto» do BNDES.

3. Histórias paralelas

Nesta seção será mostrada 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 20th and 21st 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 data dos anos 1950 nos EUA. Não compreenderemos 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 SAGE -*Semi-Automatic Ground Environment*- nos anos 1950.

Naquele cenário de desenvolvimento inicial, ainda não havia sistemas de software (os termos software e ES sequer existiam). Era um cenário comumente referenciado como «campo verde»[11], no qual a racionalidade dominante era a de maximização da eficiência de utilização dos então caros e limitados recursos de hardware. A própria engenharia do hardware influía diretamente na maneira

[11] «Campo verde» ou «terra virgem» é uma expressão (não só no jargão da informática) que serve para figurar uma área, ou situação, ainda não cultivada, tratada, construída - greenfield land-, em contraste com uma outra situação -brownfield land- onde algo já foi cultivado, tratado, construído, podendo até conter os escombros do que ali teria existido.

de encarar o desenvolvimento de software, sugerindo um processo que enfatizava revisões e verificações exaustivas das especificações antes do processo de produção. 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). Nota-se, então, que no momento em que praticamente nascia a computação eletrônica, os profissionais eram 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 operação era algo complexa, 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.[12] Neste cenário, os profissionais dependiam uns dos outros para a consecução de quase todas as tarefas. 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

[12] Somente em meados dos anos 1970 é que surgiram os primeiros cursos na área de computação no Brasil.

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, com a 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 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, passaram a existir um número expressivo 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 1960, 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) relaciona a necessidade 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, valorizando a estética matemática de concisão como norma de elegância. 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 os 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 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. Segue-se a era do «*software crafting*» que, junto à postura libertária dos anos 1960,

questionadora do regime centralizado de autoridade, 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*[13], cunhada na famosa conferência da OTAN de 1968 (NAUR, 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*.[14] No entanto, uma vez atendida a demanda, nos anos 1970, pelos sistemas que compunham o chamado *back office* – sistemas administrativos, financeiros e contábeis - passou a existir uma grande pressão por sistemas e serviços que atendessem às diversas áreas de negócio do BNDES. Por não conseguir dar conta de toda essa demanda, o DESIS passou a encarnar 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

---

[13] Com o rápido aumento do poder dos computadores e da complexidade dos problemas que podiam ser enfrentados, o termo crise do software (software crisis, ou software 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, muito mais acelerada 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, 1969, p.17, p.121).

[14] 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.

uma descentralização da informática, de modo que passaram a existir núcleos nas áreas de negócio onde os próprios profissionais dessas áreas desenvolviam os aplicativos que necessitavam, à revelia de padrões, métodos e o controle centralizado do DESIS. Esses novos programadores passaram 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 vivenciaram a era da programação por tentativa e erro (*code and fix*), do desenvolvimento de software insubordinado ao controle de processos rígidos e da 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 projetação (*design*) dos sistemas de software. Basicamente, propunha-se uma criteriosa análise de requisitos previamente 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 forma 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.

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.[15] Trata-se de apenas um caso, cuja 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*. Porém, trata-se, a nosso ver, de insinuação «inspirada», na medida em que parece repetir o padrão encontrado em alguns 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 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 (2005, p.111), 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 repre-

[15] Vale ressaltar que o padrão de desfasagem temporal entre EUA e Brasil observado neste caso não compreende todos os aspectos da informática do BNDES, restringindo-se unicamente ao enquadramento do estudo, a saber, a adoção de modelos de desenvolvimento de software. A título de exemplo, pode-se imaginar que tal padrão já não se sustente da mesma forma caso o foco do estudo recaísse sobre a adoção de hardware.

senta uma rede não convergente que ainda não foi amortizada [diferente da situação do Primeiro Mundo] (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 salvaguardar sob a aura da «universalidade».

Referências

BOEHM, B., 2006, «A View of 20th and 21st Century Software Engineering». In: 28th International Conference on Software Engineering (ICSE), Shangai, China, p.12-29.

CAMPBELL-KELLY, M., 2004, *From Airline Reservations to Sonic the Hedghog: A History of the Software Industry*, MIT Press.

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

CARMO, W. R. do, 2005, NGN – Uma análise sociotécnica da convergência das telecomunicações no Brasil. Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro.

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

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, p.521-527.

EDWARDS, P.N., 1997, *The Closed World: computers and the politics of discourse in Cold War America*. Massachusetts, MIT Press.

FLECK, L., 1986, *La génesis y el desarrollo de un hecho científico*. Madrid, Alianza.

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.

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.

NAUR, P.; RANDELL, B. (eds.), 1969, *Software Engineering: Report on a Conference Sponsored by the NATO Science Committee, Garmish, Germany, 7th to 11th October 1968*. Brussels, Scientific Affairs Division. Disponível em: <http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF>. Acesso em: 20 jul. 2007.

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

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., 2006, Vila Velha/ES. Disponível em: <http://www.cos.ufrj.br/~handrade/woses/woses2006/anais.htm>. Acesso: 01 jul. 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., 2007, Porto de Galinhas/PE. Disponível em: <http://www.cos.ufrj.br/~handrade/woses/woses2007/anais.htm>. Acesso: 01 jul. 2007.

TEIXEIRA, C.A.N, 2007, Um olhar sociotécnico sobre a engenharia de software: o caso do BNDES. Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro.