28
03 / 2015

Computação Brasil

Revista da
Sociedade Brasileira
de Computação

SBC
Sociedade Brasileira de Computação

35 ANOS

ENGENHARIA DE SOFTWARE

Qual é o impacto da ES no mercado de Computação e na sociedade como um todo?

Paulo Roberto Freire Cunha
Presidente da Sociedade Brasileira de Computação

UM CICLO QUE SE FECHA

COM O ENCERRAMENTO DA ATUAL GESTÃO, UMA NOVA DIRETORIA DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO SERÁ NOMEADA DURANTE O CSBC.

Estamos concluindo mais um ciclo para a Sociedade Brasileira de Computação (SBC) com o encerramento das gestões 2011/2013 e 2013/2015, nas quais tive a honra de atuar como presidente ao lado de um grupo de colegas altamente qualificado.

Nesses quatro anos, a SBC promoveu atividades baseadas no planejamento estratégico da instituição e no planejamento de ações da Diretoria atual. Um exemplo foi a implantação da Representação Institucional em Brasília (DF), permitindo que a SBC fique mais próxima das decisões governamentais sobre o Programa de Ciência, Tecnologia e Inovação do País. Outra novidade foi a criação do programa de Conferencistas Seniores da SBC, com objetivo de mobilizar porta-vozes para contribuir com as discussões que movimentam a área de TIC. Destacamos ainda a homologação do novo estatuto da SBC, as atividades de comemoração dos 35 anos da instituição, as ações voltadas à renovação de associados, as novas formas de atração de alunos à área de Tecnologia, a implantação um novo modelo financeiro e a consolidação de uma relação mais próxima com empresas e indústria. Quero manifestar a minha imensa satisfação pelo trabalho realizado. E faço um agradecimento a toda a equipe envolvida e aos associados pela confiança. Por fim, convido todos a prestigiarem esta edição da revista, que aborda o impacto da Engenharia de Software no universo da Computação. Boa leitura!

COMO SE ASSOCIAR

Se você deseja renovar a anuidade ou se associar à SBC, confira o valor anual:

Categoria	Valor pago até 31.12.2014	Valor pago a partir de 01.01.2015
Efetivo/Fundador	R$ 150,00	R$ 160,00
Efetivo Associado à ACM	R$ 135,00	R$ 145,00
Estudante	R$ 58,00	R$ 62,00
Efetivo Associado à ACM	R$ 45,00	R$ 50,00
Estudante de Graduação Básico	R$ 14,00	R$ 15,00
Institucional	R$ 912,00	R$ 970,00
Assinante Institucional C	R$ 5.064,00	R$ 5.380,00
Assinante Institucional B	R$ 2.832,00	R$ 3.010,00
Assinante Institucional A	R$ 1.488,00	R$ 1.580,00

A anuidade da SBC vale pelo ano fiscal (janeiro a dezembro). Associados da SBMicro têm desconto.

28
03/ 2015

Computação Brasil

Revista da Sociedade Brasileira de Computação

www.sbc.org.br

Caixa Postal 15012
CEP: 91.501-970 – Porto Alegre/RS
Av. Bento Gonçalves, 9.500 - Setor 4 – Prédio 43412 – Sala 219
Bairro Agronomia - CEP: 91.509-900 - Porto Alegre/RS
Fone: (51) 3308.6835 | Fax: (51) 3308.7142
E-mail: comunicacao@sbc.org.br

Giornale Comunicação Empresarial
Fone: (51) 3378.7100 - www.giornale.com.br
Fotos: Arquivo SBC

Índice

AGOSTO 17 a 21 — **XX SBQS XIV**
Simpósio Brasileiro de Qualidade de Software (SBQS 2015)
Manaus - AM sbqs2015.com.br/

AGOSTO 24 a 26 — **I ERAD-RJ**
I Escola Regional de Alto Desempenho do Estado do Rio de Janeiro (ERAD-RJ 2015)
Petrópolis - RJ eradrj2015.lncc.br

AGOSTO 24 a 25 — **II BRASERO**
II Brazilian Workshop on Service Robotics (BRASERO 2015)
Salvador - BA www.acso.uneb.br/brasero2015

AGOSTO 26 a 28 — **II CARLA**
II Latin American High Performance Computing Conference (CARLA 2015)
Petrópolis - RJ www.ccarla.org

AGOSTO 26 a 29 — **XVIII SIBGRAPI**
XXVIII Conference on Graphics, Patterns and Images (SIBGRAPI 2015)
Salvador - BA sibgrapi2015.dcc.ufba.br/index.html

AGOSTO 26 a 28 — **VI ERI-MS**
VI ERI-MS 2015 Escola Regional de Informática 2015Symposium (LANOMS 2015)
Coxim - MS www.eri2015.ufms.br/

AGOSTO 27 a 29 — **IV ENCOSIS**
IV Encontro Regional de Computação e Sistemas de Informação (ENCOSIS 2015)
Manaus - AM www.encosis.com.br

AGOSTO 21 a 26 — **VI CBSoft**
VI Congresso Brasileiro de Software: Teoria e Prática (CBSoft 2015)
Belo Horizonte - MG www.cbsoft2015.dcc.ufmg.br/

SETEMBRO 1 a 3 — **VII LANOMS**
VIII Latin America Network Operations and Management Symposium (LANOMS 2015)
João Pessoa - PB www.lanoms.org/2015

SETEMBRO 13 a 16 — **XXX SBBD**
XXX Simpósio Brasileiro de Bancos de Dados (SBBD 2015)
Petrópolis - RJ dexl.lncc.br/sbbd2015/

SETEMBRO 18 a 21 — **XXVII SBAC-PAD 2015**
XXVII Simpósio Brasileiro de Arquitetura de Computadores – Processamento de Alto Desempenho (SBAC-PAD 2015)
Florianópolis - SC www2.sbc.org.br/sbac/2015/

SETEMBRO 19 a 21 — **II SMPINF**
II Simpósio de Informática do IFSUL (SIMPINF 2015)
Passo Fundo - RS simpinf.passofundo.ifsul.edu.br/

SETEMBRO 20 a 21 — **XIII SIRC**
XIII Simpósio de Informática da UNIFRA (SIRC 2015)
Santa Maria - RS www.sirc.unifra.br

SETEMBRO 21 a 23 — **X SBIAGRO**
X Congresso Brasileiro de Agroinformática (SBIAGRO 2015)
Ponta Grossa - PR www3.uepg.br/sbiagro2015/

SETEMBRO 21 a 23 — **VI Agile Brazil**
VI Conferência Brasileira em Métodos Ágeis (Agile Brazil 2015)
Porto de Galinhas - PE www.agilebrazil.com

SETEMBRO 26 a 30 — **IV CBIE**
IV Congresso Brasileiro de Informática na Educação (CBIE 2015)
Maceió - AL ic.ufal.br/evento/cbie_laclo2015/

SETEMBRO 27 a 30 — **III SBR – XII LARS**
III Simpósio Brasileiro de Robótica (SBR 2015) e XII Latin American Robotics Symposium (LARS 2015)
Uberlândia - MG eventosrobotica2015.wordpress.com/

APRESENTAÇÃO | Engenharia de Software

TUDO É SOFTWARE

QUAL É A IMPORTÂNICA DA ENGENHARIA DE SOFTWARE PARA O MERCADO E PARA A GERAÇÃO DE CONHECIMENTO?

por Sérgio Soares

A ENGENHARIA DE SOFTWARE (ES) lida com métodos, metodologias, ferramentas e disciplinas, que colaboram para o desenvolvimento de sistemas. Por isto, a ES acaba sendo uma ponte que integra ainda áreas da Computação, como Banco de Dados, Linguagens de Programação, Interface Homem Máquina, entre outras, e outras áreas que transcendem a Computação. As principais instituições de ensino e pesquisa no país têm departamentos e centros da área de Computação com fortes grupos da área de Engenharia de Software. Todos notamos que o software está cada vez mais presente em tudo que nos rodeia, daí a frase (meio provocativa, meio séria) que costumamos usar afirmando que "Tudo é Software".

Este número especial da Computação Brasil traz um conjunto de 11 artigos com reflexões que vão desde questões ligadas ao ensino, passando pela pesquisa na área, até visões de mercado e exemplo de tecnologias que estão nos influenciando, seja como professores, pesquisadores, desenvolvedores ou simplesmente usuários diretos e indiretos de software.

Os três primeiros artigos refletem sobre o ensino na área de ES. O Professor Silvio Meira (CIn/UFPE) fala especificamente da necessidade de reflexão sobre o ensino de ES frente às necessidades do mercado, trazendo questionamentos como se as "graduações são ultrapassadas pelo conhecimento e práticas dos negócios?". Uma discussão sobre as competências do Engenheiro de Software é feita pelo Professor Daltro Nunes (UFRGS), que traz mais uma reflexão sobre como os gestores acadêmicos deveriam proceder para definir os conteúdos dos cursos. Ainda na linha da discussão sobre ensino, a Professora Itana Gimenes (DIN/UEM) reflete sobre a necessidade de apropriar-se das teorias e práticas pedagógicas e fala de dilemas didáticos da disciplina de ES.

Os quatro artigos seguintes discutem a pesquisa em ES. A Professora Cláudia Werner (COPPE/UFRJ) traz pontos importantes para o estabelecimento de uma agenda de pesquisa e as perspectivas para jovens pesquisadores em ES. Em seguida, as dimensões de pesquisa em ES são discutidas pelo Professor Paulo Borba (CIn/UFPE), que aborda decisões importantes a serem tomadas nos projetos e na carreira de um pesquisador. No artigo seguinte, a Professora Thais Batista (DIMAP/UFRN) destaca as conquistas da Comissão Especial de Engenharia de Software (CEES) da Sociedade Brasileira de Computação (SBC) no desenvolvimento da ES no Brasil. O Professor Leonardo Murta (IC/UFF), atual Coordenador de Programa do Simpósio Brasileiro de Engenharia de Software (SBES), transmite um relato do passado, presente e futuro do SBES, que este ano chegará a sua 29ª edição. Um relato de parte dos resultados de um projeto de cooperação internacional na área de Sistemas de Sistemas é endereçada pelos Professores Augusto Sampaio (CIn/UFPE) e Juliano Iyoda (CIn/UFPE).

Por fim, finalizamos este número especial sobre Engenharia de Software da Computação Brasil com artigos mais na linha de mercado e do desenvolvimento de software em si. O Professor Julio Leite (PUC-Rio) faz uma reflexão sobre as responsabilidades e os desafios de quem constrói software e da possibilidade de o próprio software coletar dados para analisar seu comportamento. Rodrigo Senra (EMC) discute um tema muito falado hoje em dia, a Internet das Coisas (IoT). Por fim, Antônio Valença (CEO da Pitang) discute a representatividade no cenário mundial de inovação na área de TIC no Brasil. Valença mostra uma lista de ações estruturadoras que podem ajudar a melhorar a posição do Brasil neste cenário mundial.

SÉRGIO SOARES

É Mestre e Doutor em Ciência da Computação pela UFPE, Professor Adjunto do CIn/ UFPE, Diretor do ISI-TICs (Instituto SENAI de Inovação para Tecnologias da Informação e Comunicação), Coordenador Executivo do INES (Instituto Nacional de Ciência e Tecnologia para Engenharia de Software), Presidente da Comissão Especial de Engenharia de Software (CEES) da Sociedade Brasileira de Computação (SBC), Coordenador Geral do CSBC 2015 (Congresso da Sociedade Brasileira de Computação) e foi eleito Diretor de Articulação com Empresas da SBC para o Biênio Agosto 2015- Setembro 2017.

SISTEMAS DE INFORMAÇÃO E ENGENHARIA DE SOFTWARE: CADÊ AS ESCOLAS?

por **Silvio Meira**

NA NOSSA ÁREA, A IMPRESSÃO É DE QUE AS GRADUAÇÕES FORAM ULTRAPASSADAS PELO CONHECIMENTO E PRÁTICAS DOS NEGÓCIOS.

NUNCA TANTOS E TANTAS COISAS dependeram tanto de sistemas de informação (SI) como agora. Na prática, quase tudo depende de software, em todas as facetas da atividade humana. E não só na economia: o comportamento de cada um e o nosso, como grupo e sociedade, são performances sobre – às vezes dentro de – SI. Imagine a vida sem WhatsApp, para ter uma ideia do que estamos falando. No Brasil falamos de 13% de queda na receita das operadoras móveis em 2015, boa parte devido à mudança de comportamento de quem deixou de fazer ligações telefônicas, funcionalidade analógica que havia sido digitalizada, para interagir via texto, imagem, áudio e vídeo sobre uma plataforma digital por essência, que habilita um novo universo de conexões e relacionamentos. Telefone virou app...

Para quase todo mundo, WhatsApp é um app, está no nome. Para nós, é um sistema de informação, de uma classe que poucos apostavam que existiria em futuro próximo. WhatsApp tem 1 bilhão de usuários, 700 milhões diários, enviando 30 bilhões de mensagens/dia. Imagine-se professor ou aluno de um curso de graduação ou pós, na cadeira de desenho, desenvolvimento, implantação, operação e evolução de sistemas (multimídia) interativos em tempo quase real para centenas de milhões de usuários simultâneos... Imaginou? Imagine também porque ninguém deu tal ementa ou fez tal tese em lugar nenhum.

É provável que a disciplina ou tese não existiram (e talvez não existam) porque ninguém, em nenhuma faculdade, tenha se dedicado a estudar, sistematizar e compartilhar os

fundamentos teóricos e práticos que levariam alguém a se preparar para fazer SI do grau de complexidade de um WhatsApp. E o mesmo vale para a arquitetura do software e hardware por trás deste e outros SI usados por bilhões de usuários. E isso é uma pena.

Por muitas razões, fragmentamos o ensino de informática em verticais como SI (de onde raramente sai alguém que entenda um ERP), Ciência da Computação (de onde quase nunca sai alguém que entenda como Google funciona, quanto mais pensar e fazer o que viria depois dele) e Engenharia de Software (ES), de onde nunca vi sair ninguém que – a partir de conhecimento adquirido no curso – conseguisse agregar valor num ambiente real de desenvolvimento de software. E essas são só três da miríade de formações que há por aí, incluindo algumas que nunca deveriam ter existido.

Na nossa área, a impressão é de que as graduações foram ultrapassadas pelo conhecimento e práticas dos negócios. Academias em empresas como InfoSys e Accenture educam colaboradores no ritmo da mudança das plataformas teóricas e práticas do negócio, ao tom de milhares de horas formais no início da carreira e outras centenas de aprendizado explícito por ano, sem contar oportunidades e aprendizados implícitos. Em muitos casos, trata-se de formação no ensino médio (há vários casos no Porto Digital), até porque – e ainda bem – ninguém precisa de uma graduação para exercer profissões de informática no Brasil. Pelo menos até o país piorar a tal ponto, o que não é de todo improvável.

Em negócios intensivos em conhecimento – e há poucos mais dinâmicos do que TICs – parece que a universidade é im-

portante em ensino e pesquisa, mas deixou de ser relevante, essencial para os negócios. E nem sempre foi assim. Porque programar, há algumas décadas, era tão complexo (e raro) que quase só se aprendia nas universidades. Uma boa graduação, nos anos 70/80, formava programadores competentes, mestres nas estruturas de dados e algoritmos mais conhecidos. Uma graduação espetacular adicionava sistemas operacionais e bancos de dados. E tal "formado" dava conta das expectativas de mercado de seu tempo. Não mais.

De resto, programação deveria ser assunto do ensino médio há tempos, uma das linguagens essenciais, como Matemática, Lógica e Física.

Ninguém precisa fazer uma graduação em Informática, hoje, para aprender a programar. De resto, programação deveria ser assunto do ensino médio há tempos, uma das linguagens essenciais, como Matemática, Lógica e Física. Por outro lado, a fragmentação dos currículos de graduação em termos de assuntos, e não sua unificação ao redor de problemas e projetos que se enfrentará na prática, criou uma colcha de retalhos que raramente dá resultado. Ou, quando dá, é porque o que o estágio ensinou a escola não estragou.

Precisamos fazer uma revisão do que e como se cria oportunidades para aprender informática, em especial SI e ES. É quase uma irresponsabilidade que os futuros profissionais dessas áreas façam os cursos que fazem hoje, e que saiam deles, em quase qualquer instituição, tão ignorantes da vida real e tão despreparados para ela, se sua formação ficar só a critério dos cursos de graduação (e pós, em muitos casos). Ainda bem que os alunos parecem ser muito mais espertos

do que os professores e aprendem quase tudo por fora. Por quanto tempo ainda vamos deixar que seja assim?...
Se SI e ES fossem irrelevantes para a sociedade e economia, até que se poderia deixar pra lá e fingir que se ensina e os alunos aprendem. Mas – felizmente – não é o caso. Teremos que resolver o problema de onde e como aprender SI e ES apropriadamente, pois a incompetência de nossa formação, nessas áreas, é um duplo problema econômico: recursos investidos no ensino dessas competências e habilidades se perdem em sua quase totalidade e, por causa disso, são as empresas que têm, na prática, que investir para formar seu capital humano. ●

SILVIO MEIRA | Engenheiro Eletrônico (ITA, 77), MSc (UFPE, 81) e Ph.D. (Kent, UK, 85) em Computação, nasceu em Taperoá, PB, é batuqueiro de Maracatu, fundador e ex-presidente da SBC. Professor Titular (aposentado) de Engenharia de Software do CIN/UFPE, foi Fellow do Berkman Center, Harvard e Cientista-Chefe do CESAR. Professor Associado da FGV DIREITO RIO e Coordenador do INES.org.br.

Competências do Engenheiro de Software

por Daltro Nunes

ESSAS COMPETÊNCIAS SÃO ESTABELECIDAS PELAS DIRETRIZES CURRICULARES DE COMPUTAÇÃO E INCLUEM ANÁLISE, AVALIAÇÃO, INTEGRAÇÃO E GERENCIAMENTO.

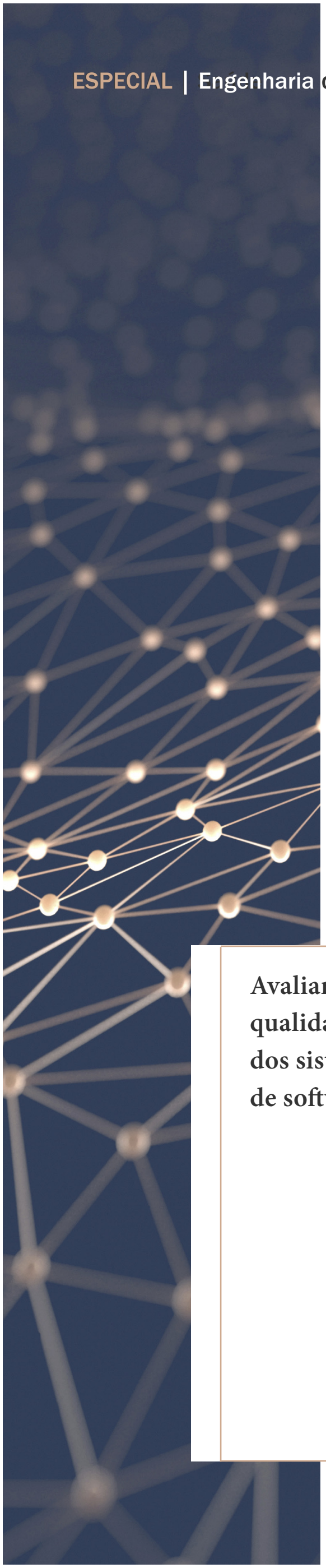
O TERMO “ENGENHARIA DE SOFTWARE” foi introduzido no final da década de 60 como uma resposta à chamada crise de software que surgiu com as experiências associadas ao projeto, desenvolvimento e manutenção de sistemas de software confiáveis de larga escala. Os sistemas de software eram construídos como torres de Babel: sem planejamento, sem comunicação, sem documentação etc. Muitos projetos foram abortados e a história tem inúmeros casos de prejuízos causados. Naquela década, a Engenharia de Software foi definida como o estabelecimento e a utilização de sólidos princípios de engenharia para produzir economicamente software confiável e que rodasse eficientemente em máquinas reais. Como os princípios de engenharia eram necessários para projetar e construir sistemas físicos complexos, surgiu a ideia de que eles poderiam ser necessários também para construir software de larga escala. A Engenharia de Software passou a ter presença na academia e no mundo industrial. A Engenharia de Software tem claramente raízes na Ciência da Computação. A lista, não exaustiva, dos princípios de engenharia inclui: *estudo, especificação, projeto, planejamento, realização de estudos de viabilidade técnico e econômica, gerência de processos, desenvolvimento, manutenção, coordenação e supervisão de equipes de trabalho, vistoria, perícia e avaliações, emissão de laudos e pareceres.* Transversalmente, atenção à legislação, às normas técnicas e de segurança nacionais e internacionais, tudo dentro dos princípios da ética científica e profissional. Como a solução de um problema é um produto, a aplicação dos princípios de engenharia garante a construção do produto dentro dos prazos e custos planejados, qualquer que seja o produto (material ou imaterial). Qualquer curso que usa os princípios de engenharia é um curso de engenharia, mesmo não tendo engenharia no seu nome. Parado-

Como a solução de um problema é um produto, a aplicação dos princípios de engenharia garante a construção do produto dentro dos prazos e custos planejados...

xalmente, as Diretrizes Curriculares Nacionais de Engenharia tem ainda um ranço dos antigos currículos mínimos, chamando de curso de engenharia aqueles que têm na sua grade curricular um conjunto de conteúdos curriculares como: *fenômenos dos transportes, resistências dos materiais, química etc.*

"Casa de ferreiro, espeto é de pau": é assim que se pode dizer da comunidade científica de computação. A comunidade aplica os princípios de engenharia para produzir software, mas não para criar/reformar cursos de Engenharia de Software. Na produção de software, os primeiros e mais importantes passos estão na definição de requisitos, nas fases finais na implementação. Entretanto, quando da criação/reforma de um curso de Engenharia de Software (ou outro qualquer), os gestores acadêmicos começam pela grade curricular (implementação) e não pela especificação do profissional.

A fase de especificação inclui o desenho do profissional com suas competências sociais e técnicas, ou seja, as capacidades que devem possuir para realizar atividades de desenvolvimento de software. As competências sociais estabelecem as relações entre o exercício profissional e a sociedade que, devidamente instanciadas, valem para qualquer curso. As competências técnicas do Engenheiro de Software são estabelecidas pelas Diretrizes Curriculares de Computação. Algumas delas são: Analisar e selecionar tecnologias adequadas para a construção de software; Avaliar a qualidade de sistemas de software; Integrar sistemas de software; Gerenciar projetos de software conciliando objetivos conflitantes, com limitações de custos, tempo e com análise de riscos; Aplicar adequadamente normas técnicas; Qualificar e quantificar seu trabalho baseado em experiências e experimentos; Exercer múltiplas atividades relacionadas a software, como: desenvolvimento, evolução, consultoria, negociação, ensino e pesquisa; Conceber, aplicar e validar princípios, padrões e boas práticas no desenvolvimento de software; Analisar e criar modelos relacionados ao desenvolvimento de software; Identificar novas oportunidades de negócios e desenvolver soluções inovadoras.



Os gestores acadêmicos devem, a partir das competências, encontrar conteúdos que habilitem os alunos a ter essas competências. No âmbito do Fórum de Engenharia de Software (FEES), realizado no contexto do Simpósio Brasileiro de Engenharia de Software (SBES), um grupo de pesquisadores e professores construíram o que foi chamado de "refinamento de competências". Este processo, que exige acadêmicos muito especializados da área de engenharia de software, parte das competências técnicas e, por refinamentos sucessivos, procuram chegar aos conteúdos. A ideia é que, se a grade curricular de um curso incluir os conteúdos encontrados, os alunos terão as competências estabelecidas. Este processo, que teve a participação, inclusive, da Universidade de Goiás, de Brasília e de várias universidades gaúchas (UFRGS, PUC-RS, UNIPAMPA, LASSALE), já tem quatro anos e ainda não foi concluído. A comunidade de pesquisadores e professores de Engenharia de Software pretende encerrar o processo na próxima edição do FEES, devendo os resultados serem publicados em Anais do evento.

A seguir é mostrado um exemplo de refinamento de uma competência técnica dos profissionais de Engenharia de Software *"Avaliar a qualidade de sistemas de software"*.

Avaliar a qualidade dos sistemas de software	Entender quais são os atributos de qualidade de produto de software e sua utilidade	Conteúdo: atributos de qualidade de produto de software	
	Aplicar mecanismos de medição da qualidade de produto de software	Conteúdos: métricas de produto de software, técnicas de avaliação de produto	
	Aplicar técnicas e procedimentos de validação, verificação e teste	Aplicar técnicas e procedimentos de validação e verificação estáticos	Conteúdos: técnicas de análise dinâmica de artefatos de software
		Aplicar técnicas e procedimentos de validação e verificação dinâmicos	Conteúdos: técnicas de revisão e análise estática de artefatos de software

Com base nos conteúdos produzidos pelo refinamento de competências, os gestores acadêmicos podem, adequadamente, compor disciplinas e distribuí-las na grade curricular, incluindo aí as demais disciplinas básicas necessárias, como matemática, ciência da computação, etc., as disciplinas eletivas e as atividades complementares. Eles devem também determinar como as disciplinas devem ser trabalhadas: à distância, presencial, seminários, em grupos etc., e as necessidades de estágios. É um processo que pode durar meses, até anos. A construção de um Currículo de Referencia pela SBC vai auxiliar muito a construção de Projetos Pedagógicos de Cursos de Engenharia de Software.

De imediato, os refinamentos de competências podem ser usados na verificação da completude dos conteúdos do currículo. Se o currículo de um curso de engenharia de software incluir os conteúdos do refinamento, então os alunos terão as competências esperadas.

DALTRO NUNES | É Professor titular e Secretário de Avaliação Institucional da UFRGS. É Doutor em Informática pelo Instituto de Informática da Universidade de Stuttgart-Alemanha, Mestre em Informática pela PUC/RJ e graduado em Engenharia Elétrica-Eletrônica pela Escola de Engenharia, UFRGS. Suas principais área de pesquisa são Engenharia de Software, métodos formais e semântica formal. Mais informações: http://lattes.cnpq.br/1830250331989097.

OS DILEMAS DIDÁTICOS DA ENGENHARIA DE SOFTWARE

por Itana Gimenes

A ENGENHARIA DE SOFTWARE, COMO DISCIPLINA DA COMPUTAÇÃO, DEVE APROPRIAR-SE DAS TEORIAS E PRÁTICAS PEDAGÓGICAS CONTEMPORÂNEAS E CONSCIENTIZAR-SE DO PERFIL DE SEUS ESTUDANTES.

ESTAMOS DIANTE DE UM CONTEXTO educacional que questiona drasticamente as formas de aprendizagem. Não é o caso de ignorar as teorias de aprendizagem existentes, mas sobretudo de ressignificá-las. Podemos citar como elementos contextuais: (i) as modalidades de educação – presencial, a distância e híbrida; (ii) o inesquecível tripé da Universidade: ensino, pesquisa e extensão; (iii) as Tecnologias de Informação e Comunicação (TIC) contemporâneas; (iv) as mudanças nos espaços físicos de aprendizagem; (v) a aprendizagem autônoma e em rede; e, (vi) a diversidade cultural.

Diante desses elementos contextuais, a Engenharia de Software, como disciplina da Computação, deve apropriar-se das teorias e práticas pedagógicas contemporâneas e conscientizar-se do perfil de seus estudantes, para planejar as práticas de ensino-aprendizagem com o uso eficiente de ferramentas de TIC (Gimenes et al., 2012) (Computação Brasil vol. 22). Isto considerando a experiência de seus quase 50 anos de existência, nos quais passou de arte a métodos tradicionais; a exigência de flexibilidade que levou aos métodos ágeis; ao reconhecimento da influência das questões sociais no desenvolvimento de software; e a evolução evidente das ferramentas de apoio ao desenvolvimento de software (Fuggeta, 2014).


Trabalho em rede

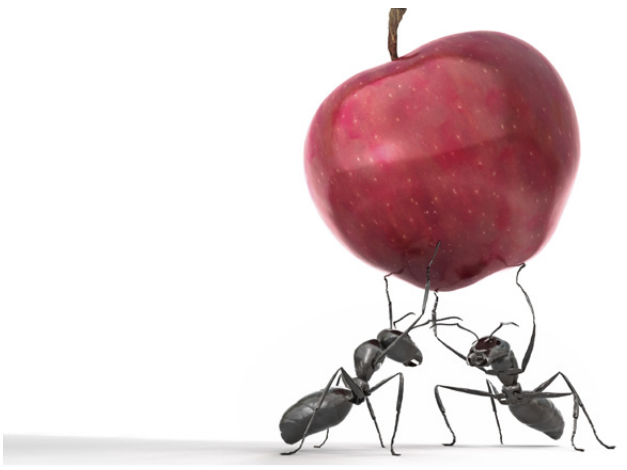
Destacamos assim, pelo menos 7 (sete) dilemas didáticos envolvidos na disciplina de ES:

1. **Teoria versus prática:** a Engenharia de Software é uma disciplina inerentemente prática; seus princípios levam diretamente a construção de produtos que são utilizados pela sociedade. Porém, produtos de software "podem" ser facilmente construídos sem o adequado uso de princípios de ES, ainda como arte, por pessoas que têm intuição ou por empresas que não seguem princípios e arriscam sua reputação.

2. **Abstração e Modelagem:** a Engenharia de Software é fortemente baseada no poder de abstração de conceitos e representação desses em modelos computacionais, **Porém**, os alunos começam a aprender esses conceitos ainda muito imaturos, muitas vezes não conhecem os ambientes em que os sistemas funcionam. É necessário interdisciplinaridade.

3. **Rápida evolução da tecnologia:** a tecnologia usada na ES evolui muito rapidamente (ex.: linguagens, frameworks, ferramentas, hardware etc.). Porém, os professores não conseguem se apropriar dos recursos tecnológicos com a mesma rapidez para utilizar em suas aulas.

4. **Ensino de Engenharia de Software em currículos de computação:** os currículos de computação têm pouco espaço para ES. Isto implica uma ou duas disciplinas genéricas de ES que usam livros textos clássicos (ex.: Pressman, Sommerville) e não há espaço para aulas práticas ou projetos. Não é raro encontrar professores de outras subáreas da computação que negligenciam o conteúdo de ES, ex., especificam software de suas áreas específicas com diagramas informais, e não fazem a devida relação entre as disciplinas básicas e a ES nos currículos, tratam sempre de problemas fechados com enunciados predefinidos, enquanto os projetos reais são problemas em aberto com mudanças continuas de requisitos.



Trabalho em grupo

5. **Ensino de ES orientado pelo modelo em cascata:** a distribuição das disciplinas de Engenharia de Software segue o modelo em cascata (requisitos, análise, projeto ...), isto **Implica que** o aluno demora muito para programar sistemas. Isto é incompatível com o que ele "ouve" da prática na indústria, por exemplo, sobre métodos ágeis. O protelamento da prática de desenvolvimento de software desmotiva o aprendizado do conteúdo de ES.

6. **Práticas conteudistas:** os professores são conteudistas, isto **Implica que** não trabalham as soft-skills como comunicação, liderança, resolução de conflitos e dinâmica de grupo. Essas habilidades são imprescindíveis para engenheiros de software.

7. **Qualidade de software:** existem padrões de qualidade de software nacionais e internacionais, **Porém**, usuários e clientes ainda aceitam sistemas ineficientes de bancos, passagens aéreas, órgãos governamentais, etc. Ainda aceitam facilmente desculpas de "o sistema está fora do ar, volte outra hora", "os dados foram perdidos", "digite novamente". Por outro lado, clientes têm dificuldade de aceitar pagar mais para um desenvolvimento mais seguro.

A criação dos Cursos de ES é uma grande contribuição para a solução dos problemas acima (dilema 4), mas os educadores precisam: inovar para atacar a dilema 5, os currículos também precisam ser ágeis; precisam explorar o eixo de extensão universitária para se aproximar da indústria e da comunidade, para assim oferecer oportunidade para que os estudantes vivenciem problemas reais, atacando os problemas apresentados pelos dilemas 1, 2 e 7; precisam promover espaços dinâmicos de aprendizagem para que os alunos possam interagir em rede e realizar projetos conjuntos, com tecnologias inovadoras, discutir as questões éticas, assim atacando os dilemas 3 e 6.

Novas tecnologias

O curso de ES demanda um corpo docente de perfil misto e qualificado, no momento em que o tripé universitário se volta demais para a pesquisa teórica e a busca cega e individualista por índices de produtividade em pesquisa, o curso sofre, se distancia do mercado e dos desejos de muitos estudantes.

REFERÊNCIAS

FUGGETTA, A.; DI NITTO, E. Software process. Proceedings of the Future of Software Engineering (FOSE), International Conference on Software Engineering (ICSE 2014), Hyderabad, India, ACM Press, 2014, p. 1-12.

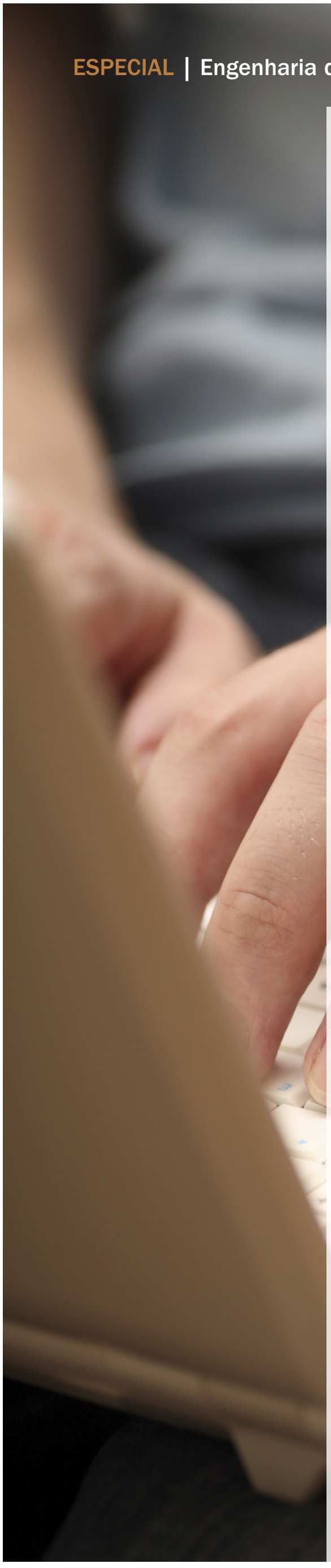
GIMENES, ITANA M. S. ; BARROCA, LEONOR ; BARBOSA, E. F. . The future of human resources qualifications in Software Engineering meeting demands from industry and benefiting from educational and technological advances. In: 26o. Simpósio Brasileiro de Engenharia de Software (SBES) - Trilha Especial, 2012, Natal, RN. Anais do 26o. Simpósio Brasileiro de Engenharia de Software (SBES) - Trilha Especial. USA: IEEE, 2012. v. 1. p. 1-5.

ITANA MARIA DE SOUZA GIMENES | Professora titular da Universidade Estadual de Maringá (UEM), Paraná. Atualmente é Pró-Reitora de Extensão e Cultura da UEM. Fez estágio de pós-doutorado sênior na Open University, Reino Unido (2011) em pesquisa relacionada a projeto de aprendizagem aplicada à Engenharia de Software. Em 2005, fez pós-doutorado na School of Computer Science, University of Waterloo, ON, Canadá, em pesquisa sobre linha de produto de software. Tem Ph.D. em Ciência da Computação pela University of York, UK (1992).

JOVENS PESQUISADORES EM ENGENHARIA DE SOFTWARE: AGENDA DE PESQUISA E PERSPECTIVAS

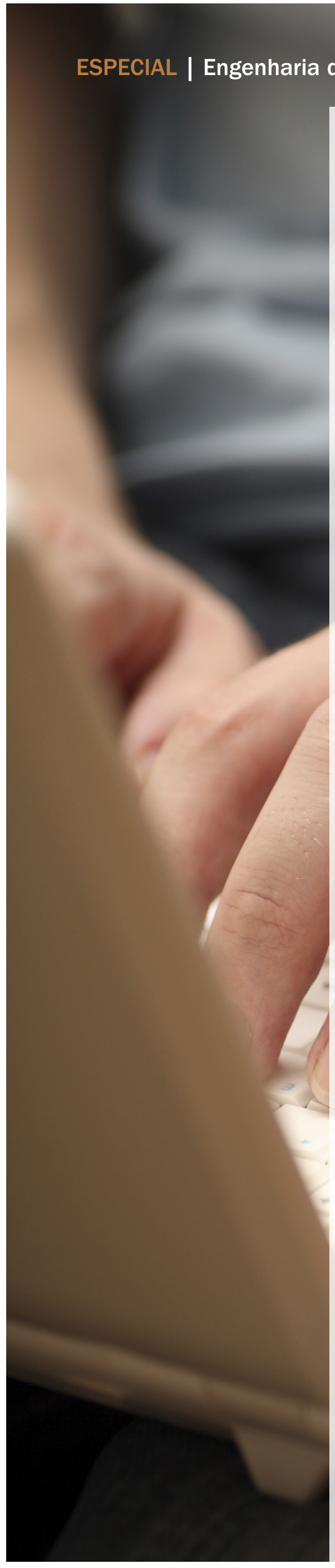
por Claudia Werner

DIANTE DE TAMANHA DIVERSIDADE, MUITOS PESQUISADORES, EM ESPECIAL OS JOVENS, TÊM DÚVIDA QUANTO AO TÓPICO DE PESQUISA AO QUAL DEVEM SE DEDICAR NA ÁREA DE ENGENHARIA DE SOFTWARE.



Desde as primeiras edições do Simpósio Brasileiro de Engenharia de Software (SBES), em 1987, temos acompanhado as pesquisas em diversos temas como gerência de projetos, qualidade de software, processos de desenvolvimento, métodos formais, linguagens de programação, ambientes e ferramentas de desenvolvimento de software e reutilização de software, dentre outros. Em 2010, com a criação de uma conferência especificamente voltada para a área de software (Conferência Brasileira em Software: Teoria e Prática - CBSoft), aumentaram-se ainda mais as possibilidades de temas de pesquisa, agregando, além do SBES, o Simpósio Brasileiro de Componentes, Arquiteturas e Reutilização de Software (SBCARS) e o Simpósio Brasileiro de Linguagem de Programação (SBLP) – e o Simpósio Brasileiro de Métodos Formais (SBMF) um ano depois –, e vários eventos satélites (i.e. workshops). Diante de tamanha diversidade, muitos pesquisadores, em especial os jovens, têm dúvida quanto ao tópico de pesquisa ao qual devem se dedicar na área de Engenharia de Software (ES).

Com o objetivo de passar dicas valiosas para pesquisadores interessados em ES, foi realizado o I Workshop de Diretrizes para Jovens Pesquisadores em ES, coordenado por Eduardo Almeida (UFBA) e Paulo Masiero (USP-São Carlos), em Salvador, no âmbito do CBSoft 2010 – iniciativa essa similar ao "New Software Engineering Faculty Symposium" realizado na "International Conference on Software Engineering" (ICSE) em 2001, em Toronto, Canadá, que vem se repetindo anualmente neste contexto. Esta única edição do evento resultou na publicação de um livro, "A carreira de Pesquisador em Engenharia de Software: princípios, conceitos e direções", editado pelos coordenadores (bit.ly/CarreiraES). Dentre os tópicos tratados, temos: agenda de pesquisa, estratégia de publicação, ética profissional e educação em ES.



É fundamental que propiciemos às novas gerações meios para que elas possam dar continuidade ao que se iniciou décadas atrás.

Neste ano, em julho, foi realizada a Segunda Escola Latino-Americana em Engenharia de Software, coordenada por Ingrid Nunes (UFRGS) e Francisco Dantas (UERN), em Porto Alegre. Fui convidada a proferir uma palestra exatamente sobre o estabelecimento de uma agenda de pesquisa e as perspectivas para jovens pesquisadores em ES. Tal tarefa não é trivial e está fadada à obsolescência dos temas apresentados em função da dinâmica da área, assim como, a previsão de um futuro para a carreira do engenheiro de software (ou pesquisador) corre o risco de estar baseada apenas no que acontece no presente ou ocorreu no passado. Mesmo assim, aceitei o desafio e destaco aqui dois pontos que considero importantes para estarmos atentos:

1) A fim de identificar novas possibilidades (temas) de pesquisa, a leitura de artigos das principais conferências na área (i.e., ICSE, SBES etc.), assim como a participação em workshops ou trilhas que se proponham a discutir novas ideias e áreas emergentes são fundamentais. Além disso, problemas reais existem e é importante que pesquisadores estejam sempre atentos aos interesses industriais; e

2) A decisão por seguir uma carreira acadêmica ou se inserir em pesquisa na indústria, por exemplo, em parques tecnológicos, é difícil e precisa ser tomada com cautela. Algumas qualidades são esperadas para se ter um futuro promissor nesta área, tais como criatividade, independência, capacidade de

comunicação e colaboração, dentre outras. Atividades como a supervisão de alunos/subordinados, a procura por financiamento de projetos e o ensino/treinamento de novos engenheiros de software fazem parte do dia a dia de um pesquisador.

Sabemos que o desenvolvimento da área de ES no Brasil e o sucesso de seus pesquisadores está diretamente relacionado ao quanto eles são capazes de garantir a evolução da pesquisa e o engajamento em ações que mantenham a comunidade ativa no pais. Neste sentido, é fundamental que propiciemos às novas gerações meios para que elas possam dar continuidade ao que se iniciou décadas atrás. Neste sentido, a discussão de guias e diretrizes para jovens pesquisadores é salutar e será sempre bem-vinda! ●

CLAUDIA WERNER | Doutora pela COPPE/UFRJ (1992) e professora do Programa Engenharia de Sistemas e Computação desta instituição desde 1994, onde lidera a linha de pesquisa em Engenharia de Software. Atua na área há mais de 20 anos, em temas como Reutilização, Visualização e Ecossistema de Software e Educação em Engenharia de Software. É pesquisadora do CNPq e FAPERJ. lattes.cnpq.br/9719247117370600.

DIMENSÕES DE PESQUISA EM ENGENHARIA DE SOFTWARE

por **Paulo Borba**

DECISÕES IMPORTANTES A SEREM TOMADAS NOS PROJETOS E NA CARREIRA DE UM PESQUISADOR EM ENGENHARIA DE SOFTWARE.

AO LONGO DA CARREIRA, um pesquisador toma decisões que são determinantes para o sucesso dos seus projetos e resultados de pesquisa. Entender as dimensões envolvidas nessas decisões e as escolhas disponíveis é importante para avaliar as consequências e refletir sobre aonde se quer chegar como pesquisador. Com o objetivo de ajudar jovens pesquisadores a terem maior consciência no direcionamento da sua carreira, discutimos neste artigo algumas dessas dimensões no contexto da Engenharia de Software.

Uma das mais discutidas dimensões é a aplicabilidade esperada dos resultados gerados pela pesquisa. Apesar de existir forte tendência pela busca de maior aplicabilidade, é justificável e necessário investir também em pesquisa com aplicação indireta, desde que haja preocupação em estabelecer os fundamentos da área. Esse é o caso, por exemplo, de teorias lógicas que definem a base matemática de processos e ferramentas de refatoração; pode ser também o caso das teorias sociais de equipes de desenvolvimento de software, apesar de que essas podem também ter aplicação mais direta.

O maior risco na escolha do posicionamento nessa dimensão é perder o foco na contribuição científica ao tentar maximizar a aplicabilidade. Pode-se acabar resolvendo problemas concretos de desenvolvimento em empresas, mas sem gerar conhecimento científico novo. Consequentemente, não há evidência dos benefícios da solução proposta, nem comparação adequada com soluções alternativas. Similarmente, o foco em tópicos da moda como cidades inteligentes e computação em nuvem, para citar dois mais recentes, pode ser arriscado. A falha em identificar claramente os problemas centrais de Engenharia de Softwa-

re envolvidos pelas fortes camadas de marketing e aplicações inovadoras associadas a esses tópicos pode levar a resultados de pesquisa não originais ou com contribuição limitada. Pior, pode nem gerar benefícios socioeconômico industriais.

Outro risco importante nessa dimensão é se deixar enganar por aplicações utópicas da pesquisa, seja pelo foco em problemas mal definidos ou que não existem na prática, seja por soluções que são baseadas em assunções inválidas sobre a realidade de equipes de desenvolvimento de software. Por exemplo, um método de escalonamento de tarefas de desenvolvimento que assume a execução sequencial de tarefas não vai ser usado por equipes, nem por quem desenvolve um sistema sozinho, já que nesse caso a preocupação com gerenciamento é menor.

O maior risco na escolha do posicionamento nessa dimensão é perder o foco na contribuição científica ao tentar maximizar a aplicabilidade.

Além da aplicabilidade dos resultados, outra dimensão importante é a natureza do processo de pesquisa adotado. O ideal é que o pesquisador seja capaz de adotar métodos de pesquisa adequados para o problema a ser considerado, o que pode envolver a combinação de métodos empíricos (com forte uso de estatística e projeto de estudos e experimentos), analíticos (com sólida base de lógica e matemática) e computacionais (com simulação e prototipação).

O maior risco aqui é o pesquisador se limitar a usar apenas um pequeno subconjunto dos métodos existentes, o que limita também os tipos de problemas a serem atacados, estudos a serem realizados e perguntas a serem respondidas. Pior, essa escolha pode também inviabilizar a solução efetiva dos problemas

e a obtenção de respostas sob diferentes perspectivas, levando a resultados superficiais. Por exemplo, um pesquisador que basicamente usa surveys como método de pesquisa corre o risco de ter uma carreira baseada em publicações que respondem a perguntas de pesquisa sobre assuntos desconexos, com perspectivas restritas e sem efetivamente progredir com a solução de problemas da área.

O pesquisador deve procurar se alinhar com os padrões e critérios de qualidade, elegância e publicação da sua comunidade internacional de pesquisa.

Escolhas sobre outros fatores que definem o processo de pesquisa são menos importantes. Por exemplo, tanto o foco em melhorias pontuais em ferramentas e técnicas de desenvolvimento de software quanto o foco em maiores desafios ou processos e ambientes de desenvolvimento podem levar a resultados relevantes sem maiores riscos. No primeiro foco, esperam-se evidências fortes de vantagens e desvantagens em relação a soluções existentes. No segundo, esperam-se soluções mais originais e não incrementais. De qualquer forma, é necessário cuidado para não acarretar soluções mínimas, no primeiro caso, nem soluções não usáveis, no segundo caso.

Outra dimensão a ser considerada é a visibilidade e o impacto dos resultados de pesquisa. Para maximizar esses aspectos, o pesquisador deve procurar se alinhar com os padrões e critérios de qualidade, elegância e publicação da sua comunidade internacional de pesquisa. Infelizmente, na prática, às vezes é necessário se preocupar também com as regras locais de avaliação e promoção de pesquisadores, que podem adotar critérios não alinhados com a visibilidade. Por exemplo, a avaliação local

pode ser quantitativa e baseada apenas em artigos, enquanto a comunidade internacional pode valorizar também resultados como ferramentas e benchmarks de qualidade. Enfim, buscar a excelência internacional pode se tornar um objetivo mais difícil do que deveria.

Por fim, várias outras dimensões estão menos relacionadas a riscos, e normalmente as opções podem ser combinadas em um dado projeto: foco em qualidade ou produtividade de software; foco em aspectos técnicos ou humanos da Engenharia de Software; pesquisa que tenta prevenir ou remediar problemas; resultados que ajudam desenvolvedores ou pesquisadores. O importante é estar consciente das escolhas feitas em cada dimensão e refletir sobre as possíveis consequências. ●

PAULO BORBA | Professor Titular de Desenvolvimento de Software do Centro de Informática da UFPE, onde lidera o Grupo de Produtividade de Software. Seus principais interesses de pesquisa são nos seguintes temas e na integração entre eles: modularidade de software, linhas de produtos de software e refatoração. Paulo é Doutor em Computação pela Universidade de Oxford, e Mestre e Bacharel em Computação pela UFPE.

O PAPEL DA COMISSÃO ESPECIAL DE ENGENHARIA DE SOFTWARE (CEES) NO DESENVOLVIMENTO DA ÁREA NO BRASIL

por Thais Batista

COM O OBJETIVO DE ATUAR EM DIVERSAS VERTENTES, AS COMISSÕES ESPECIAIS DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO GERAM QUALIDADE PARA AS ATIVIDADES ACADÊMICAS DE CADA ÁREA DE ESPECIALIDADE.

A SOCIEDADE BRASILEIRA DE COMPUTAÇÃO (SBC), através das suas Comissões Especiais (CEs), organiza a comunidade de Computação do Brasil de acordo com áreas de interesse. Atualmente, há mais de 25 Comissões Especiais, abrangendo diversas áreas da Computação. As CEs têm como missão atuar em várias vertentes, em especial promover e garantir a qualidade das atividades acadêmicas em cada área de especialidade, colaborar com a definição de diretrizes curriculares, apoiar na definição de agendas de pesquisa na área, incentivar a interação da academia com a indústria e manter uma lista de discussões entre os membros. Uma das principais ações das CEs é promover a realização de eventos (Congressos, Simpósios, Workshops) para possibilitar a disseminação de trabalhos científicos e a interação entre os membros da comunidade. As Comissões Especiais também são responsáveis por listar os eventos nacionais e internacionais da sua área, que são usadas pelo Perfil-CC da SBC. Essa lista tem sido usada pela CAPES, desde 2009, como base para o Qualis-CC. Vale ressaltar que a SBC não realiza classificação dos eventos, essa é uma prerrogativa exclusivamente da CAPES.

A Comissão Especial de Engenharia de Software (CEES) é uma das comissões especiais da SBC com maior número de membros, responsável por fomentar o desenvolvimento da área de Engenharia de Software no Brasil.

A CEES é responsável pela organização de vários eventos da área. Há 29 anos teve início o mais tradicional evento de Engenharia de Software no Brasil, o Simpósio Brasileiro de Engenharia de Software (SBES), que é um tema explorado em outro artigo desta edição. A CEES também organiza o Simpósio Brasileiro de Componentes, Arquiteturas e Reutilização de Software (SBCARS), o Simpósio Brasileiro de Qualidade de Software (SBQS) e apoia uma série de Workshops relacionados com a área. Ao reconhecer a grande amplitude da área e a sua interseção com diversas outras áreas correlatas, em 2010, a CEES criou o Congresso Brasileiro de

A CEES vem apoiando as publicações dos sócios da SBC no JSERD ...

Software: Teoria e Prática (CBSoft), em parceria com a Comissão Especial de Métodos Formais (CEMF) e a Comissão Especial de Linguagens de Programação (CELP). O CBSoft visa agregar a comunidade de ES, proporcionando um fórum que reúne os principais eventos da área. Sua primeira edição foi realizada em Salvador, em 2010, incluindo dois simpósios tradicionalmente organizados pela CEES, o SBES e o SBCARS, e dois simpósios de outras comissões especiais, o Simpósio Brasileiro de Métodos Formais (SBMF) e o Simpósio Brasileiro de Linguagem de Programação (SBLP). Na perspectiva de estimular a interação da academia com a indústria, a CEES promove, no contexto do CBSoft, a Trilha da Indústria.

No seu papel de fomentar as pesquisas na área, a CEES criou, em 2012, em parceria com a Springer, o *Journal of Software Engineering Research and Development (JSERD)*. Esse é um marco importante para a comunidade, materializando o anseio de se ter um periódico internacional da área de Engenharia de Software e dar visibilidade aos artigos da área. O JSERD surgiu de uma iniciativa articulada de pesquisadores da área de Engenharia de Software, liderado pela Professora Itana Gimenes (UEM), com apoio da SBC através da sua parceria com a editora Springer. A CEES vem apoiando as publicações dos sócios da SBC no JSERD e convoca a comunidade a participar ativamente da consolidação desse periódico, submetendo artigos para manter a edição continuada e possibilitar a indexação internacional.

A CEES procura manter laços estreitos com as atividades de outras sociedades científicas na área de Engenharia de Software. Há alguns anos a CEES indica um representante na *IFIP (International Federation for Information Processing)*, que participa das reuniões onde são tomadas decisões de abrangência mundial sobre a área.

A CEES também prima pela valorização de pesquisadores que têm se destacado por contribuir para o avanço da área de Engenha-

ria de Software no Brasil. Anualmente, desde 2004, a CEES concede uma distinção para um professor pela sua contribuição relevante para a área. O primeiro professor homenageado pela CEES foi o Prof. Carlos José Pereira de Lucena (PUC-Rio). Desde então, já foram homenageados 12 (doze) professores de diferentes instituições brasileiras, incluindo PUC-Rio, ICMC-USP São Carlos, UFRJ, UFRGS, UFRN, UEM e UFPE.

A ação política é outra vertente de atuação da CEES. Diversos pesquisadores que fazem parte da CEES têm atuado em órgãos governamentais, conselhos e fóruns científicos, participando ativamente de discussões e de proposições de políticas nacionais e motivando a necessidade de editais de fomento para a área. Um exemplo bem-sucedido da atuação da comunidade junto a órgãos de fomento foi o edital do CNPq, em 2007, para bolsas de doutorado específicas para a área de Engenharia de Software.

A CEES que temos hoje é resultado do trabalho persistente de gerações de pesquisadores da área, que construíram uma comunidade bem organizada, com eventos sólidos e um periódico em fase consolidação. Somos herdeiros desse legado e convocamos todos os estudantes, pesquisadores e profissionais da área a se juntarem a CEES e a realizarem esforços para a continuidade das conquistas. Os desafios são diversos e o envolvimento da comunidade é fundamental para se avançar ainda mais, acompanhando o desenvolvimento internacional da área. A página web da CEES está disponível em comissoes.sbc.org.br/ce-es. ●

THAIS BATISTA | Professora Associada da Universidade Federal do Rio Grande do Norte (UFRN), onde atua na área de Engenharia de Software e Sistemas Distribuídos. Thais foi Diretora de Secretaria Regionais da Sociedade Brasileira de Computação (SBC) no período de 2010 a 2013 e coordenadora da Comissão Especial de Engenharia de Software (CEES) da SBC nos períodos de 2011-2012 e 2013-2014.

SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE:

PASSADO, PRESENTE E FUTURO

por **Leonardo Murta**

NESTE BREVE ARTIGO APRESENTAMOS UM POUCO DA HISTÓRIA DO SBES, OS PREPARATIVOS PARA A SUA 29ª EDIÇÃO E TENDÊNCIAS FUTURAS.

O Simpósio Brasileiro de Engenharia de Software (SBES) é um dos simpósios mais tradicionais SBC e o principal evento técnico-científico de Engenharia de Software da América Latina. Ele tradicionalmente congrega pesquisadores, praticantes e estudantes de graduação, mestrado e doutorado. Na sua programação há palestras convidadas, sessões técnicas e sessões de ideias inovadoras, assim como uma variada gama de atividades que ocorrem em paralelo, como sessões da indústria, sessões de ferramentas, fórum de educação, workshop de teses e dissertações, painéis, tutoriais, minicursos e workshops em temas específicos. Essas atividades discutem processos, métodos, técnicas e ferramentas para o desenvolvimento, manutenção e evolução de software. Neste breve artigo apresentamos um pouco da história do SBES, os preparativos para a sua 29ª edição e tendências futuras.

PASSADO

Em 1987 nascia o SBES, em Petrópolis, RJ, sob a coordenação da Professora Ana Regina Rocha (COPPE/UFRJ). Nesse momento a Engenharia de Software tinha recém completado a sua maioridade, já que o termo foi cunhado por Margaret Hamilton, integrante do projeto espacial Apollo, em 1968. Além disso, a principal conferência internacional da área, a International Conference on Software Engineering (ICSE), tinha somente 13 anos de existência naquele momento.

Desde 1995, o SBES vinha sendo realizado em conjunto com o Simpósio Brasileiro de Banco de Dados (SBBD). Porém, em 2010 o SBES se juntou aos Simpósios Brasileiros de Linguagens de Programação (SBLP), Métodos Formais (SBMF) e Componentes, Arquiteturas e Reutilização de Software (SBCARS) para formar o Congresso Brasileiro de Software: Teoria e Prática (CBSoft). Nesse período, o SBES teve relevante participação na formação de pesquisadores e encubação de novos eventos. Por

exemplo, tanto o SBMF quanto o Simpósio Brasileiro de Qualidade de Software (SBQS) tiveram as suas primeiras edições como workshops do SBES.

Em 2011 o SBES fazia seu 25º aniversário, e para celebrar a data o então coordenador, Prof. Alessandro Garcia (PUC-Rio), organizou uma trilha especial chamada "SBES é 25". Nessa trilha diversos trabalhos discutiram a história da Engenharia de Software no Brasil, que se mistura com a própria história do SBES. Dentre esses trabalhos, cabe um destaque para o artigo de Silveira Neto et al. (2013), que resgatou com maestria dados históricos do SBES até aquele momento. Além disso, em um trabalho conjunto dos Profs. Francisco Dantas (UFRN) e Alessandro Garcia (PUC-Rio), todas as publicações anteriores do SBES foram resgatadas e disponibilizadas na Biblioteca Digital Brasileira de Computação.

PRESENTE

Neste ano, de 21 a 26 de setembro, ocorre a 29ª edição do SBES em Belo Horizonte (MG). Para esta edição o SBES recebeu 122 submissões de artigos de 9 países diferentes, sendo 92 válidas. Esses artigos foram avaliados por 89 membros do comitê de programa e 45 avaliadores externos de 10 países diferentes. Cada um dos artigos recebeu de 3 a 4 pareceres, o que implica mais de 350 pareceres no total. Os cinco tópicos de interesse de maior popularidade dentre esses artigos são: (1) métodos, técnicas, linguagens e ferramentas para Engenharia de Software; (2) Engenharia de Software experimental; (3) processos de software (incluindo métodos ágeis); (4) verificação, validação e teste de software; e (5) engenharia de requisitos.

Neste ano, de 21 a 26 de setembro, ocorre a 29ª edição do SBES em Belo Horizonte (MG).

Para o SBES 2015 contaremos com excelentes palestras convidadas, tratando de temas inovadores e intrigantes. Na quarta-feira, 23/9,

o pesquisador Thomas Zimmermann (Microsoft Research, EUA) irá discutir sobre o surgimento de cientistas de dados na indústria de software. Esses cientistas de dados apoiam as equipes de desenvolvimento na compreensão de grandes quantidades de dados sobre o processo de desenvolvimento e utilização do software. Na quinta-feira, 24/9, o Professor Prem Devanbu (UC Davis, EUA) fará um paralelo entre linguagem natural e linguagem de programação, visando mostrar que boa parte do código que escrevemos é repetitivo e previsível, podendo ser alvo dos mesmos modelos estatísticos usados para processamento de linguagem natural. Na sexta-feira, o Professor Sílvio Meira (UFPE) irá discutir sobre a relevância da comunidade acadêmica de Engenharia de Software.

FUTURO

Em um trabalho conjunto com os Professores Arilo Dias Neto (UFAM), Rafael Prikladnicki (PUCRS) e Márcio Barros (UNIRIO), aceito para publicação pelo Journal of the Brazilian Computer Society (JBCS), fizemos um levantamento sobre a nova geração de doutores formados nos últimos dez anos em Engenharia de Software no Brasil. Para tal, utilizamos o método snowball sampling (Goodman 1961) visando identificar, dentro da própria comunidade de Engenharia de Software, quem são os principais pesquisadores jovens da área e quais são seus perfis de atuação e publicação.

O SBES 2015 é uma excelente oportunidade para todos os envolvidos com pesquisa e prática em Engenharia de Software.

Dentre os principais achados, descobrimos que o SBES é o fórum mais procurado pela nova geração de pesquisadores em Engenharia de Software: 80% deles já publicaram nele. Além disso, apesar de a formação desses pesquisadores ter se concentrado no sudeste e nordeste, eles foram contratados por universidades de todas as cinco regiões do país de forma bas-

tante equilibrada, o que tornará a Engenharia de Software ainda mais inclusiva no Brasil. Por fim, os cinco tópicos de pesquisa com maior popularidade dentre esses jovens pesquisadores são: (1) teste de software; (2) desenvolvimento orientado a aspectos; (3) Engenharia de Software experimental; (4) linguagens de programação; e (5) qualidade de software. Apesar de os termos seguirem uma taxonomia diferente da utilizada nos tópicos de interesse SBES, é possível notar uma grande intercessão com os principais tópicos de interesse da edição de 2015.

O SBES 2015 é uma excelente oportunidade para todos os envolvidos com pesquisa e prática em Engenharia de Software. Para praticantes, é o momento de se atualizarem nos assuntos mais badalados da Engenharia de Software. Para pesquisadores, é uma oportunidade ímpar de estabelecerem novas parcerias de pesquisa e, no caso de professores, captarem novos alunos. Para alunos, é uma vitrine para divulgarem as suas pesquisas e ganharem visibilidade nacional. Neste ano, em Belo Horizonte, cidade de fácil acesso e com povo acolhedor, teremos o ambiente propício para mais uma excelente edição do SBES. Contamos com a sua participação! ●

LEONARDO GRESTA PAULINO MURTA | Professor do Instituto de Computação da Universidade Federal Fluminense (UFF), Doutor e Mestre em Engenharia de Sistemas e Computação pela COPPE/UFRJ, e Bacharel em Informática pelo IM/UFRJ. É bolsista de Produtividade em Pesquisa nível 2 do CNPq desde 2009 e Jovem Cientista da FAPERJ desde 2012. Seus principais campos de atuação são Gerência de Configuração, Evolução de Software, Arquitetura de Software e Proveniência. www.ic.uff.br/~leomurta.

ENGENHARIA DE **SISTEMAS DE SISTEMAS**

Por Augusto Sampaio e Juliano Iyoda

ESPECIFICAÇÃO, MODELAGEM E ANÁLISE DE SISTEMAS QUE INTEGRAM SISTEMAS COMPLEXOS E INDEPENDENTES.

A ENGENHARIA DE SOFTWARE e, de forma mais geral, a Engenharia de Sistemas, que envolve hardware e software, possui atividades bem definidas, delineadas em termos de processos, para a construção de aplicações. Atividades como concepção, desenvolvimento, teste e manutenção são exemplos do cotidiano de um engenheiro. Estes fundamentos já estão bem estabelecidos para sistemas individuais e com propósitos bem definidos, como, por exemplo, o desenvolvimento de um novo telefone celular ou de um novo modelo de carro. Processos existentes auxiliam a sistematização das atividades de desenvolvimento, verificação (incluindo testes) e evolução de tais sistemas. Porém, existem sistemas que não são projetados de forma tradicional: eles emergem como consequência da integração de outros sistemas previamente existentes. Estes *Sistemas de Sistemas, ou SoS (do Inglês, System of Systems)*, são sistemas cujas partes são, por si só, complexas e independentes, desenvolvidas sem uma perspectiva futura de serem integradas a outros sistemas.

O sistema de saúde é um exemplo de Sistema de Sistemas: hospitais, clínicas, profissionais de saúde, laboratórios, planos de saúde, governos e serviços de emergência (ambulâncias, bombeiros e polícia) trabalham todos juntos para prover a infraestrutura necessária ao atendimento de um cidadão. Este sistema não foi desenvolvido e projetado de forma centralizada e como parte de um projeto específico, mas foi sendo construído ao longo do tempo através da integração de sistemas já existentes, independentes e distribuídos.

Outro exemplo de um Sistema de Sistemas em menor escala é o de equipamentos domésticos interagindo entre si: celulares, roteadores, sistemas de som, televisões, tablets, notebooks, relógios e até mesmo caixas de som sem fio interagem para prover algum serviço integrado. Por exemplo, o sistema de som

da casa pode receber músicas do celular e enviar para as caixas de som, com todas as comunicações acontecendo através do roteador sem fio. O resultado deste Sistema de Sistemas é uma funcionalidade nova: música do celular saindo nas caixas de som da casa. Esta funcionalidade não foi projetada de antemão por nenhum sistema, mas emergiu a partir da cooperação dos sistemas individuais. Cenários similares a este serão comuns no futuro com a chegada da internet das coisas, em que quaisquer objetos (eletrônicos ou não) estarão conectados entre si e à internet.

As disciplinas tradicionais de Engenharia assumem que um sistema será independente, terá uma arquitetura estável, uma base tecnológica estática, um ambiente relativamente controlado e estará sujeito a mudanças incrementais no futuro. Estas suposições influenciam diretamente as atividades do desenvolvimento de um sistema individual. Sistemas de Sistemas, por sua vez, violam estas suposições e, consequentemente, exigem da engenharia uma nova perspectiva.

Outro exemplo de um Sistema de Sistemas em menor escala é o de equipamentos domésticos interagindo entre si...

O caso de equipamentos domésticos interagindo entre si ilustra algumas situações em que o desenvolvimento tradicional atinge seus limites. Quando vários equipamentos participam de uma interação, um equipamento deve assumir o papel de líder para coordenar as comunicações entre todos e suas ações. As regras de eleição do líder é um algoritmo que é executado de forma descentralizada por cada equipamento. Ao final da eleição, todos os equipamentos devem atingir um mesmo estado consistente em que exatamente o mesmo líder foi eleito (não pode haver ausência de líder, nem mais de um líder). Como o algoritmo é descentralizado, o processo de eleição não conta com uma autoridade preestabelecida e centralizada. A complexidade surge da necessidade de cada equi-

pamento, executando um mesmo procedimento, sem uma coordenação central, chegar sempre a um mesmo líder.

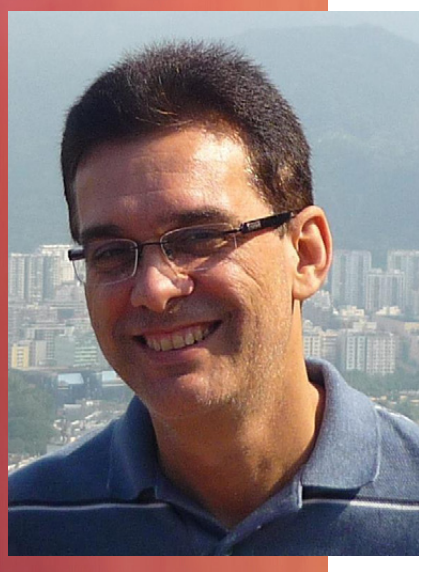
Alguns esforços internacionais de cooperação, envolvendo universidades e empresas, têm investigado este tipo de desafio e têm tentado propor teorias, processos, técnicas e ferramentas que suportem o desenvolvimento de Sistemas de Sistemas. Por exemplo, o projeto COMPASS (www.compass-research.eu) é uma iniciativa nesta direção. Particularmente, o foco do COMPASS, com o qual tivemos a oportunidade de contribuir, foi nas fases de requisitos, modelagem, projeto (design) e verificação do correto funcionamento de tais sistemas. A verificação inclui testes, mas também técnicas mais elaboradas de análise automática dos modelos construídos, que garantem que certas propriedades de interesse são preservadas. No caso de um Sistema de Sistemas, é essencial este tipo de verificação, dado que a integração de sistemas independentes tende a gerar vários problemas potenciais, como problemas de comunicação e sincronização, resultando em problemas clássicos da Computação, como situações de impasse (*deadlock*), onde a execução é impossibilitada de progredir.

O número de situações a considerar em uma análise global de tal Sistema de Sistemas é maior do que a quantidade de átomos do universo.

No contexto do COMPASS, a partir dos requisitos de uma aplicação, um modelo diagramático do Sistema de Sistemas é construído e, em seguida, tal modelo é automaticamente traduzido para um modelo em uma notação formal, passível de verificação, utilizando-se técnicas e ferramentas de análise. Para dar um exemplo de um grande desafio e importante resultado obtido no contexto do COMPASS, considere a análise de *deadlock* de um sistema de eleição de um líder, em um cenário com 32 dispositivos domésticos interagindo, como explicado anteriormente. O número de situações a considerar em uma análise global de tal Sistema de Sistemas é maior do que a quantidade

de átomos do universo. Portanto, uma análise exaustiva dessas possibilidades é, claramente, inviável. Para lidar com este problema, técnicas de análise foram desenvolvidas, permitindo que uma análise global (com custo exponencial) de uma propriedade do sistema fosse inferida a partir de um número bem mais discreto de análises locais (por exemplo, envolvendo apenas a interação entre dois dispositivos de cada vez), com custo linear.

A Engenharia de Sistemas de Sistemas é uma área ainda muito recente da Computação. De forma geral, o desenvolvimento de Sistemas de Sistemas confiáveis, dentro de um cronograma e custos previsíveis, escaláveis, utilizando padrões e explorando técnicas de reuso, é um desafio significativo e bastante interessante para as comunidades de Ciência e Engenharia da Computação. ●



AUGUSTO SAMPAIO | É Professor Titular do Centro de Informática da UFPE e Pesquisador 1B do CNPq. Possui graduação e mestrado em Computação pela UFPE e doutorado pela Oxford University. Em 2010, foi agraciado com o título de Comendador da Ordem Nacional do Mérito Científico. Atua em Engenharia de Software, com ênfase em Métodos Formais, incluindo semântica, refinamento, transformação e análise de especificações, modelos e programas concorrentes e orientados a objetos.

JULIANO IYODA | É Professor Adjunto em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco (CIn-UFPE). Juliano Iyoda é bacharel e mestre em Ciência da Computação pela UFPE e Ph.D. pela Universidade de Cambridge, Inglaterra. Durante o doutorado, Juliano desenvolveu um sintetizador de hardware utilizando o provador de teoremas HOL4. Desde então, Juliano tem se dedicado à área de Métodos Formais e Testes.

SOFTWARE CONSCIENTE

por Julio Cesar Sampaio do Prado Leite

A RESPONSABILIDADE DE QUEM CONSTRÓI SOFTWARE É ENORME, FACE AO IMPACTO QUE ESSAS MÁQUINAS DE DIFERENTES PROPÓSITOS TÊM NA SOCIEDADE.

A ENGENHARIA DE SOFTWARE é uma área de conhecimento jovem que procura disponibilizar métodos, técnicas e ferramentas para a construção de software. Construir software é fundamentalmente construir máquinas abstratas, que tomam por base máquinas físicas e as transformam em máquinas de diferentes propósitos. Essa capacidade do software torna possível que carros andem sem motorista, que aviões sejam pilotados sem piloto, carros sejam fabricados por máquinas e que máquinas de comunicação pessoal possam desempenhar tarefas tanto de comunicação por áudio, como por imagem ou por texto, além de processarem imagens.

Portanto a responsabilidade de quem constrói software é enorme, face ao impacto que essas máquinas de diferentes propósitos têm na sociedade. No entanto, a demanda por escritores de software é maior que a capacidade das universidades de produzirem engenheiros de software. Portanto, muitas máquinas de diferentes propósitos são construídas por escritores sem uma formação sólida na disciplina. Esse fato impacta na sociedade como um todo, face tanto ao custo de construção como a qualidade do produto.

Fundamentalmente o escritor de software deve lidar com três tipos de conhecimento básicos: 1) conhecimento da máquina computacional, 2) conhecimento do contexto e 3) conhecimento da engenharia de software. Esse triângulo do conhecimento é fundamental para que a máquina abstrata transforme a máquina física na máquina desejada pelo contexto, ou seja o ambiente e as suas necessidades. É grande o desafio de formar pessoas com conhecimento básico desse triângulo. O que se observa é que muitas máquinas de software são escritas sem o equilíbrio entre os vértices do triângulo.

Com a variabilidade das máquinas computacionais e dos contextos, a tarefa de prover um conhecimento em engenharia que dialogue com esses dois vértices é um grande desafio. A Engenharia de Software, como disciplina, tem feito grandes avanços no sentido de contornar tais desafios; desde métodos orientados a qualidade, como linguagens de diferentes níveis de abstração para facilitar a ta-

refa de escrita de software, assim como diferentes técnicas de escrita e automação através de ferramentas.

De maneira similar à Engenharia Mecânica, que hoje utiliza máquinas de software para desenhar suas máquinas (Desenho Apoiado por Computador), a Engenharia de Software tem pesquisado como apoiar a construção de máquinas de software usando máquinas de software. Esse desenvolvimento tem gerado diferentes camadas de abstração que se estratificam no vértice da máquina computacional. Portanto, cada vez mais, se agregam à máquina física máquinas abstratas (por exemplo: firmware, sistemas operacionais, linguagens de programação), aumentando a produtividade do escritor, mas aumentando a dependência em escritas anteriores.

Como o software é produto da escrita, reescrever é tarefa acessível ao escritor.

Como o software é produto da escrita, reescrever é tarefa acessível ao escritor: isso torna o processo de construção de máquinas de software muito próximo do processo de desenho, já que o ciclo de produção é simplesmente um processo de copia. Portanto, o processo de construção de software é fortemente baseado em evolução e com isso sua natureza flexível acarreta sérios problemas para a disciplina de construção. Ou seja, a criação confunde-se com a construção, algo inexistente nas engenharias clássicas.

Além dos diferentes níveis de abstração agregados à máquina computacional, máquinas de software têm apoiado o desenho e a montagem de diversas configurações de componentes de máquinas de software. Como o desenho/construção de software é cada vez mais uma tarefa colaborativa, torna-se fundamental que a escrita colaborativa seja a mais transparente possível, de modo que a máxima de Raymond (“Com um número de olhos suficientes, os defeitos veem à tona”) possa ser verdadeira. No entanto, construir software transparente é um desafio.

Muitas vezes, o número suficiente de olhos humanos é um recurso escasso. Nesse caso, máquinas de software devem ir além da ajuda

ao humano, mas também ter um papel mais ativo. Portanto, é importante investigar a possibilidade de que o software seja consciente, ou seja, que o próprio software saiba sobre seu comportamento e que possa saber analisar implicações desse comportamento.

O estudo da consciência de software passa pela aquisição de informações através do sensoriamento do comportamento e registro de situações (aprendizado) e posterior análise face a objetivos pré-definidos. Desta forma a Engenharia de Software tem que aprender com sistemas de controle, já amplamente utilizados nas Engenharias Elétrica e Mecânica. O estudo do sensoriamento do software pelo software é fortemente ligado a transparência, porque é necessário desenhar ambientes de sensoriamento. Esses ambientes devem ter situações transparentes que possam ser lidas por sensores, e essas leituras devem prover análises face a objetivos e as situações anteriores (aprendizado).

A construção de software consciente tem por objetivo principal automatizar a coleta/análise de informações sobre o software. Isto permitirá que o software evolua com base em objetivos predefinidos, aumentando a produtividade dos engenheiros de software e possibilitando um aumento do conhecimento sobre o próprio processo de construção/uso. Inventar métodos, técnicas e ferramentas de apoio para a construção de software consciente é uma área promissora de estudo – e necessária. Sua importância é função da necessidade de saber, mais, como esse produto abstrato, que gera tantas e diferentes máquinas, evolui. ●

JULIO CESAR SAMPAIO DO PRADO LEITE | É Ph.D., Professor Associado PUC-Rio, Membro do Grupo de Trabalho 2.9 (Engenharia de Requisitos) da IFIP, Membro da ACM, Membro da IEEE, Sócio Fundador da SBC. www.inf.puc-rio.br/~julio

ABRACADABRA: IOT!

por Rodrigo Senra

O MERCADO JÁ ESTÁ À BEIRA DE UMA INUNDAÇÃO DE PRODUTOS E PROJETOS DE INTERNET DAS COISAS E AINDA TEM DIFICULDADES EM DIFERENCIÁ-LA.

EM 1999, o funcionário britânico da Procter & Gamble Kevin Ashton cunhou o termo *Internet of Things (IoT)*, profetizando um mundo futurista em que dispositivos interconectados propiciariam economia de tempo e dinheiro. Depois de 2^4 anos, o futuro chegou.

O mercado já está à beira de uma inundação de produtos e projetos de IoT, e ainda temos dificuldade em defini-la. É qualquer coisa que possa se conectar à Internet?

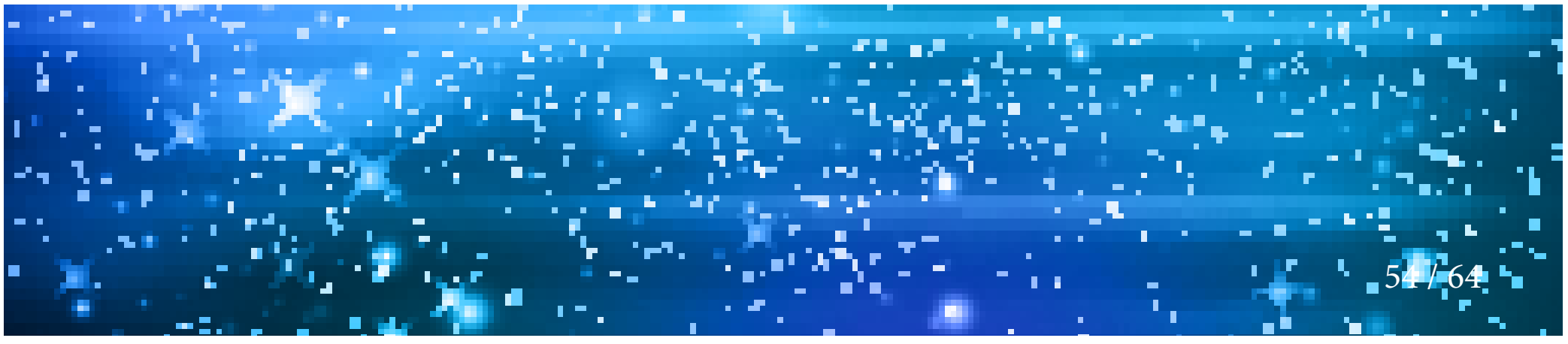
A Wikipedia define IoT como sendo: a rede de objetos físicos (coisas) que incorporam componentes eletrônicos, software, sensores, e atuadores, que agregam valor pela troca de dados entre fabricantes, operadores e outros dispositivos interoperando pela infraestrutura existente da Internet.

É mais fácil se dermos exemplos.

IoT é o *Apple Watch* lançado em 2015. Um relógio de pulso que troca mensagens, monitora sua saúde, aprende sobre a sua atividade física e até mostra as horas, atrasando no máximo 50ms em relação ao horário mundial padrão. Dick Tracy* ficaria com inveja.

IoT é a geladeira inteligente *ChillHu*b da GE, criada por hackers e para hackers. Tem 8 portas USB, Wi-fi e roda aplicações iOS. É possível obter controle direto sobre a geladeira através de uma API ou customizar a geladeira criando acessórios através de uma impressora 3D. Por exemplo, o *Milky Weigh* é uma balança programável que avisa quando o leite está acabando. Quem vai gostar é Arnold Schwarzenegger, pois no filme *The 6th Day* (2000) sua geladeira "inteligente" deixou o leite acabar. Hoje, custa apenas U$ 2,999.00, uma bagatela se comparada com os U$ 20,000.00 que era o preço da precursora Internet Digital DIOS da LG em 2000. Os early adopters entraram numa fria.

IoT é sobretudo fitness. Quantos passos percorreu? Quantas



calorias queimou? Dispositivos como *FitBit*, *FuelBand*, *LumoBack* e *Jawbone* surfam a onda de wearable computing, ficando de olho em você 24x7 e nas cores da moda. A maioria destes produtos interopera com smartphones para apresentar: estatísticas de desempenho, recomendações de postura, exercícios e dieta alimentar.

E quem veste a camisa da IoT, veste a *OMsignal Biometric Smartwear* que é uma camisa capaz de monitorar sua respiração e seus batimentos cardíacos. Além disso, gerencia a troca de umidade do seu corpo com o meio, eliminando odores. Resiste a chuva e ao suor, e pode ser lavada normalmente!

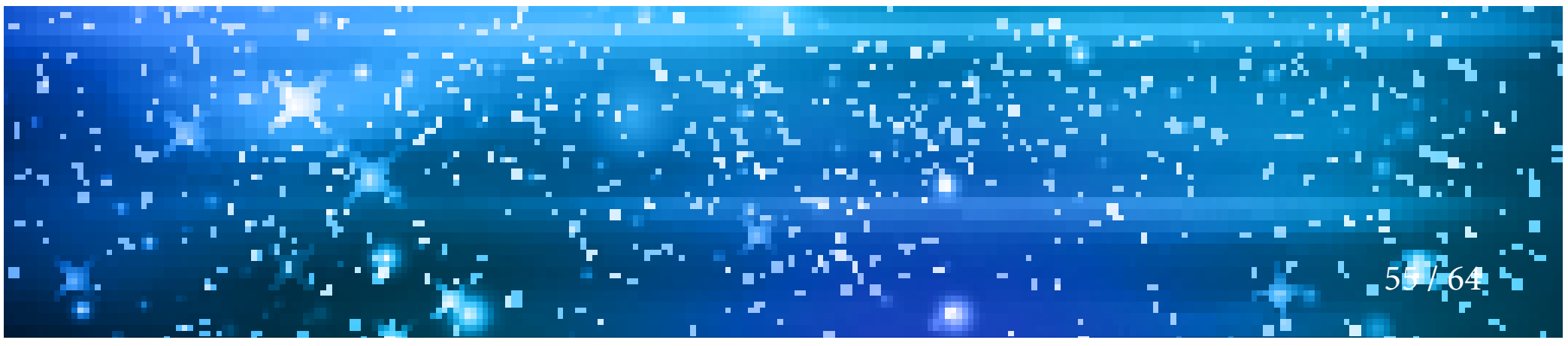
Há produtos mesmo para quem quer chutar a IoT para o alto, como a *miCoach Smartball* da Adidas, que é uma bola de futebol com um acelerômetro triaxial e fala Bluetooth. A bola permite um treinamento de precisão, medindo a velocidade, rotação, trajetória e ponto de impacto.

Mesmo quem não quer suar a camisa, mas não descuida da beleza, não foi esquecido pela IoT.

Voltado para o público feminino, o *WAY* é um consultor pessoal de estética facial que parece uma rosquinha. O WAY monitora os níveis de radiação UV e umidade do ambiente, e detecta o nível de humidade e oleosidade de camadas subcutâneas da pele, que são melhores indicadores da saúde dérmica.

As aplicações de IoT são pervasivas para todo o cenário urbano, municiando a revolução denominada SmartCities. Propiciando o monitoramento de vagas para veículos, o acompanhamento da saúde de edifícios através de vibração, o mapeamento da poluição sonora e eletromagnética, a detecção do acúmulo de lixo para uma coleta mais eficaz, e a otimização no controle de tráfego de veículos e pedestres.

Por outro lado, IoT transcende o perímetro urbano e chega à



zona rural, alavancando a agricultura e pecuária de precisão. Já existem casos de sucesso no controle de microclimas em estufas para otimizar a qualidade da produção de frutas e vegetais, e no monitoramento da umidade do solo em vinícolas, a fim de calibrar o teor de açúcar nas uvas. Na pecuária de precisão, a preocupação é o constante monitoramento dos rebanhos permitindo intervenções na hora certa. Reside nas iniciativas de IoT na agricultura a esperança de aumentar em 70% a produção de alimentos até 2050. Senão, segundo a FAO/UN, não será possível suprir a demanda de alimentos para a população (estimada) de 9.6 bilhões de pessoas em 2050.

As aplicações de IoT são pervasivas para todo o cenário urbano...

Nas áreas de Indústria e Comércio, as aplicações são tantas que seria necessário um artigo exclusivo para uma enumeração superficial de casos de uso.

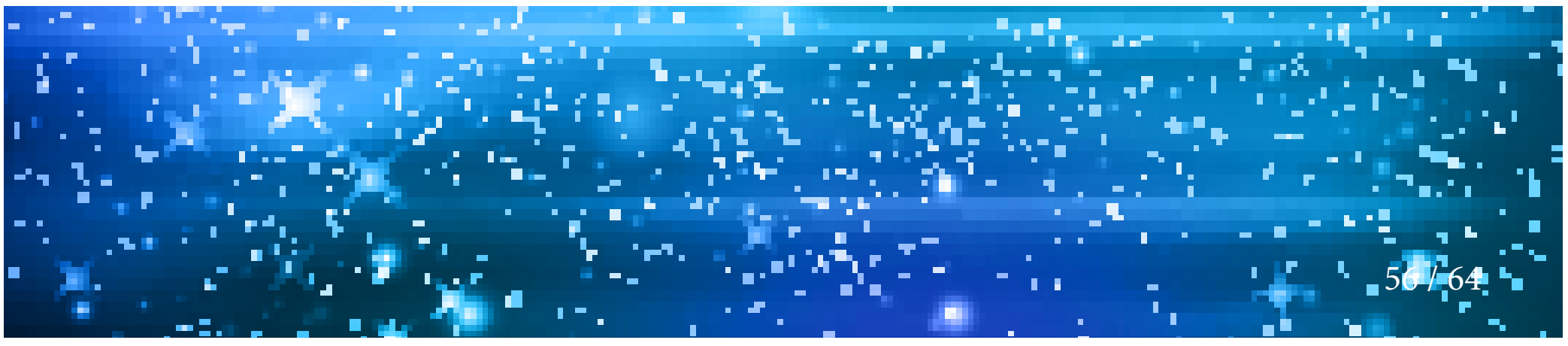
Fica patente que a excitação acerca de IoT tem um respaldo na expectativa de um vultoso retorno financeiro.

A previsão é de que o faturamento atinja U$ 300 bilhões em 2020. Segundo pesquisa do grupo Gartner, em 2020 haverá entre 26 e 30 bilhões de dispositivos IoT, e apenas 7,3 bilhões de PCs, smatphones e tablets.

O que justifica esse crescimento acelerado da IoT? Gostaríamos de arriscar 4 fatores: o barateamento do custo de chips wi-fi, *open-source, open-hardware* e *crowdsourcing*.

Para um dispositivo com custo a partir U$ 3, já não há justificativas para deixar de se incluir a conectividade sem fio.

No mundo open-source, muitas ferramentas de desenvolvimento embarcado têm favorecido a linguagem de programação Lua, que é um produto genuinamente brasileiro oriundo da PUC/TecGraf-RJ. Lua é uma linguagem dinâmica eficiente, que conso-



me pouquíssima memória, tornando-a ideal para sistemas embarcados. Não é por menos que foi adotada nas plataformas de IoT: Mihini, Koneki e Paho.

Sem dúvida alguma, um dos grandes viabilizadores do crescimento do IoT é a plataforma Arduíno de prototipização eletrônica de hardware aberto. A plataforma Arduíno tem por objetivo a criação de ferramentas de baixo custo, flexíveis e fáceis de se usar até por quem não é profissional de tecnologia, como artistas e amadores. O Arduíno gerou uma multitude de derivados, como: *Microduíno, Pyduíno, Yún, BeagleBone, Flutter, Raspeberry Pi.*

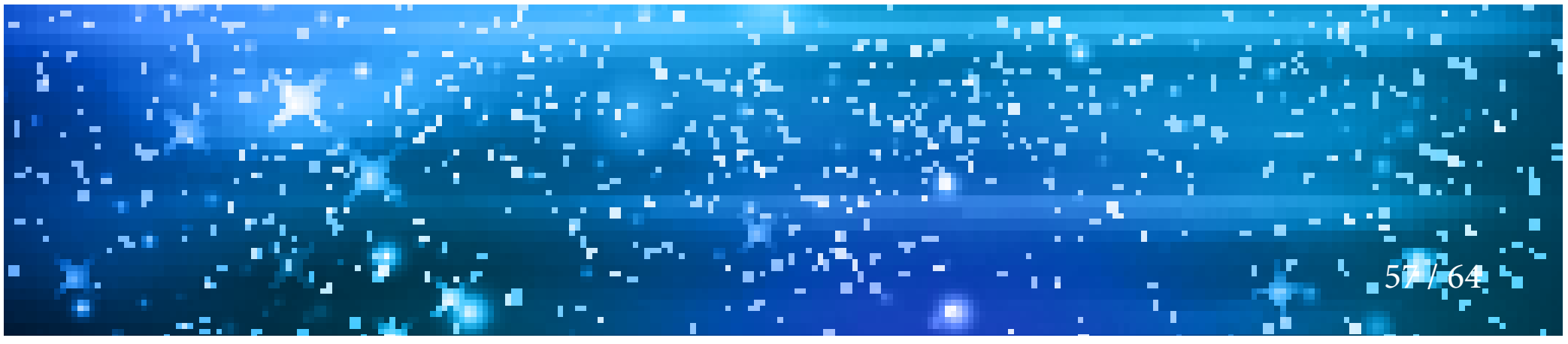
O último fator é o fenômeno de crowdsourcing, nada mais natural para unir investidores e empreendedores em IoT do que uma solução que já nasceu na Internet. Há inúmeros sites proeminentes de crowdsourcing, tais como: KickStarter, CrowdRiser, IndieGoGo, Spot.us ou Profounder. Estes sites são responsáveis pelos principais sucessos de empreendedorismo em IoT.

O Pebble Smartwatch, por exemplo, é um relógio inteligente lançado em 2013. Possui tela de LCD monocromática, CPU programável, Bluetooth, motor de vibração, magnetômetro, sensor de luz e acelerômetro. Como seus criadores não conseguiram financiamento no mercado, apelaram para o KickStarter. Nas primeiras 2 horas no ar, o projeto já havia batido a meta de U$ 100,000. Depois de 6 dias, havia se tornado o recordista de arrecadação, tendo levantado U$ 4.7 milhões. Até 2014, já havia vendido um milhão de unidades.

Outro exemplo mais modesto é o alimentador de gatos *Bistro* financiado pelo *IndieGoGo* em 2014.

O Bistro atingiu U$ 240,680 em um mês de campanha, o dobro do almejado, com o suporte de 1420 investidores.

Existem várias armadilhas inerentes às iniciativas IoT que remetem a problemas relevantes na Ciência da Computação. Para



enumerar apenas alguns: Como identificar o contexto em que está inserido o usuário na sua interação com dispositivos de IoT? Como evitar que a correlação entre vários serviços de dados interoperantes não viole a anonimidade garantida pelos serviços individualmente? Como garantir a escalabilidade de produtos desenvolvidos localmente em um mercado cada vez mais globalizado?

Em 1973, Arthur Clarke escreveu **"Qualquer tecnologia suficientemente avançada é indistiguível da magia"**, talvez inspirado em outra frase célebre de Leigh Brackett (1942) **"Witchcraft to the ignorant, ... simple science to the learned"**.

Se o mundo da IoT chegar rápido demais, antes que tenhamos tempo de educar a população mundial nos fundamentos tecnológicos em que está baseada a IoT, então verdadeiramente viveremos em um mundo de magia. O deslumbramento (devido à ignorância) de muitos andará lado a lado com a conquista intelectual de poucos.

*Dick Tracy é um detetive das tiras de quadrinhos e um personagem popular na cultura pop norte-americana.

RODRIGO DIAS ARRUDA SENRA
Engenheiro de Computação, Mestre e Doutor pelo Instituto de Computação da Unicamp. Consultant Research Scientist no Brazil & Research Development Center da EMC no Rio de Janeiro. Membro Fundador da Associação Python-Brasil.

INOVAÇÃO NO MERCADO DE SOFTWARE BRASILEIRO

por Antônio Valença

EMBORA O BRASIL SEJA A 5ª ECONOMIA DE TIC DO MUNDO, TEMOS BAIXÍSSIMA REPRESENTATIVIDADE NO CENÁRIO MUNDIAL QUANDO FALAMOS DE INOVAÇÃO EM NOVOS PRODUTOS E SERVIÇOS DE TIC. NESTE ARTIGO NOS PROPOMOS A ANALISAR ESTA QUESTÃO.

A TENDÊNCIA QUE TEMOS AO DISCUTIRMOS este assunto é de olhamos para os resultados de uma face do problema, a dos produtos/serviços disruptivos que o mercado brasileiro de software consegue produzir e consolidar. Ao olharmos por esta face vislumbramos um número muito pequeno de exemplos positivos. Se puxarmos pela memória talvez recuperemos facilmente o exemplo, do Easy Taxi, que informa ter uma rede de 120 mil taxistas em 30 países e se diz líder global de mercado; mas depois deste exemplo que outro nos vem à mente? Por que temos tão poucos expoentes nesta face da questão? E na outra face, a de pesquisa e inovação, a que produz tecnologia de base para a produção de produtos e serviços? Esta também sofre, pois via de regra temos poucos pesquisadores de uma forma geral [1] , temos um número ainda menor de pesquisadores nas empresas [2] , nossa produção científica não corresponde ao nosso potencial e à nossa representatividade econômica no cenário mundial [3] . Mudar este cenário requer um esforço conjunto de vários atores: o governo, o empresariado e os pesquisadores em si.

1. A proporção de pesquisadores na população mostra que o Brasil está longe da média mundial, que é de mais de 1.000 pesquisadores para cada milhão de habitantes. Em 2007, o país contava com pouco mais de 500 pesquisadores por milhão de habitantes, segundo o Relatório Unesco sobre Ciência 2010.
2. Cerca de 70% dos cientistas brasileiros estão nas universidades (fonte: artigo “Brasil, o País dos papers” publicado pelo Jornal do Comércio em março/2013). Em 2012, dados da Financiadora de Estudos e Projetos (Finep) diziam que 37% dos pesquisadores do Brasil atuavam em empresas, mais de um terço. Porém, na realidade, esses cientistas atuam praticamente apenas em empresas estatais ou ex-estatais (Eletrobrás, Vale do Rio Doce, Embraer, etc.)
3. O Brasil ocupa a 15ª posição entre os países de maior produtividade científica (segundo os dados do portal Scimago Journal & Country Rank entre 1996 e 2011). A posição fica aquém da colocação do país em termos econômicos - o Brasil é atualmente a 7ª economia do mundo (Banco Mundial, Abril/2014) e o 5º mercado de TIC do mundo (fonte: BRASSCOM, ABES e ASSESPRO – 2014)

Dado este cenário de operação para todos os atores envolvidos e considerando a distância que separa o Brasil dos países de vanguarda mundial, vemos que há muito trabalho a ser feito. Considero, entretanto, que temos todas as condições de encarar e superar este desafio. Para isto proponho algumas ações estruturadoras que nos ajudariam a reduzir a distância citada.

1. Revisão da Lei de Informática, Lei de Inovação, Fundos Específicos, dentre outras que demonstram exaustão e desalinhamento com os novos modelos de negócios de escala global;

2. Focar na melhoria da "qualidade da educação", onde o Brasil aparece no 126º lugar entre 144 países em estudo do WEF [4] ;

3. Aumentar os investimentos brasileiros em P, D&I, onde nosso país tem um investimento ainda abaixo da média dos países da OCDE (Brasil com cerca de 1,1% do PIB e países da OCDE perto de 1,6% [5]);

4. Alterar a legislação de financiamento de bancos públicos para permitir que patentes e planos de negócios inovadores possam servir de garantia para contratos de financiamento. Este é um grande empecilho para empresas de TI, que normalmente têm muito pouco a oferecer de garantias reais;

4. The Global Competitiveness Report 2014–2015 – World Economic Forum
5. OECD Science, Technology and Industrial Outlook 2014

5. Melhoria do ambiente de negócios com foco na redução da alta carga tributária direta e indireta, aumentando nossa competitividade internacional (o Brasil tem baixíssima participação no cenário mundial de TICs como exportador de produtos e serviços, não alcançando 2% do total do mercado [6]);

6. Mudança dos índices de avaliação dos pesquisadores pela Capes, que termina tendo uma ótica apenas "publicista" (quantitativa) e foca menos na qualidade dos trabalhos gerados. Outra forma de melhorar esse cenário é a inclusão da produção técnica na avaliação dos programas e cursos de pós-graduação;

7. Ausência de planejamento público e privado de longo prazo termina nos direcionando à discussão de questões menores e apenas conjunturais da economia e da indústria, tirando o nosso foco das questões macroeconômicas e estruturais que realmente importam. O governo pode, por exemplo, escolher alguns setores para focar o seu desenvolvimento tecnológico. No passado a Coreia do Sul escolheu os setores de eletrônica e naval, e hoje podemos ver na prática os impactos desta escolha na Samsung, LG, entre outras. Já a China obrigou as empresas que se instalaram lá – incluindo as multinacionais – a ter centros de Pesquisa & Desenvolvimento, aproveitando a mão de obra chinesa;

8. Repensar as regras burocráticas que desmotivam relações entre as universidades e as empresas. Os professores e pesquisadores acabam sendo impedidos de participar de projetos fora das universidades pela exclusividade exigida pelas instituições de ensino – regra que não existe em países de alto desenvolvimento tecnológico como os Estados Unidos;

6. UNCTAD, 2010

9. Aumentar os limites de faturamento para cálculo dos encargos tributários das startups e pequenas empresas que estão nos primeiros passos;

10. Reduzir a burocracia para o registro de patentes, que no Brasil leva, em média, onze anos para ser concluído [7]! Além disso, no Brasil a patente pertence exclusivamente à universidade ou à empresa em que o pesquisador trabalha. Esta situação pode representar um desincentivo ao mesmo;

11. Criar meios de incentivo ao investimento privado em P, D&I, visto que 90% das inovações nascem nas empresas [8];

Certamente este é um tema bastante amplo e que suscita muitas polêmicas, mas deve ser enfrentado e discutido por toda a sociedade, visto que é decisivo para a inclusão do Brasil no clube dos países prósperos neste século 21.

7. http://anprotec.org.br/site/2014/04/brasil-ocupa-penultima-posicao-em-ranking-de-patentes/

8. E. Mansfield, Contributions of new technology to the economy , in Technology, R&D and the Economy , ed. Bruce Smith e Claude Barfield. P. 125 (The Brookings Institutions, Washington, DC (1996)

ANTÔNIO VALENÇA | É Mestre em Ciência da Computação pelo Centro de Informática da Universidade Federal de Pernambuco e certificado como Gerente de Projetos pelo PMI. Atua na indústria de software desde 1986 e é CEO da Pitang (www.pitang.com.br) desde 2007.

AMPLIE SUAS CONEXÕES COM O CONHECIMENTO.

A **Association for Computing Machinery** é a maior sociedade educacional e científica do mundo na área da Computação, com mais de 100 mil membros em todo o planeta. Faça parte desse universo com vantagens exclusivas para os associados da **SBC**.

6 MESES **GRÁTIS** DE ASSOCIAÇÃO À ACM*

DESCONTOS NAS ANUIDADES SBC E ACM**

Faça sua inscrição diferenciada em **www.acm.org/membership/SBC_application**

PROGRAMA **ACM DISTINGUISHED SPEAKERS** PARA O BRASIL

Para conhecer os tópicos e os palestrantes, acesse **dsp.acm.org/**

STUDENTS E **PROFESSIONAL CHAPTERS ACM/SBC**

Participe de grupos especiais de desenvolvimento, por região ou por área de interesse. Em breve!

Association for Computing Machinery

Sociedade Brasileira de Computação

SBC

*Desconto válido para novos associados à ACM que estejam em dia com a SBC. **Veja a tabela de anuidades na Loja da SBC.