Elicitação de Requisitos - Evidências de uma Problemática na Formação dos Estudantes de Computação *

Andréa Pereira Mendonça[1,2] Evandro de Barros Costa[1,3]
Dalton Dario Serey Guerrero[1]

[1]Departamento de Sistemas e Computação – Universidade Federal de Campina Grande (UFCG)

[2]Coordenação de Informática - Centro Federal de Educação Tecnológica do Amazonas (CEFET - AM)

[3]Instituto de Computação – Universidade Federal de Alagoas (UFAL)

andrea@dsc.ufcg.edu.br, ebc.academico@gmail.com, dalton@dsc.ufcg.edu.br

Abstract. Computing students have difficulties to deal with requirements elicitation. This problem was highlighted by two empirical research - a case study carried out with students in the beginning of the course and a survey research conducted with computing teachers and IT professionals. In this paper, we present these research results and propose a pedagogical approach that supports the development of problem specification skills and good practices on software development. The approach proposed will be evaluated in the context of novice programming students.

Keywords: Requirements Elicitation, Empirical Researches, Programming Learning.

Resumo. Estudantes de computação apresentam dificuldades em lidar com elicitação de requisitos. Esta problemática foi evidenciada por meio de duas pesquisas empíricas - um estudo de caso com estudantes de programação, além de uma pesquisa do tipo *survey* com professores de computação e profissionais de TI. Neste artigo, apresentamos os resultados destas pesquisas e propomos uma abordagem pedagógica que possibilita o desenvolvimento das habilidades para especificação de problemas, assim como das boas práticas de desenvolvimento de software. A abordagem proposta será avaliada no contexto de estudantes iniciantes de programação.

Palavras-chave: Elicitação de Requisitos, Pesquisas Empíricas, Aprendizagem de Programação.

*A primeira autora recebe bolsa da FAPEAM - Fundação de Amparo à Pesquisa do Estado do Amazonas.

1 Introdução

No processo de desenvolvimento de *software*, a especificação do problema não é uma atividade simples, principalmente se considerarmos que a declaração de um problema tipicamente contém ambigüidades, contradições, falta de informações e/ou informações irrelevantes.

Transformar declarações dessa natureza em uma definição clara do que deve ser feito exige um conjunto de habilidades não triviais, tais como: identificação de informações relevantes, estratégias para aquisição e registro de informações, modelagem, diálogo e análise das restrições do problema. No modelo tradicional dos currículos de computação, tais habilidades são desenvolvidas nas disciplinas de Análise, Engenharia de Software ou denominações similares [MEC/SESU 1999].

Desta forma, é comum que nas séries iniciais, os estudantes lidem com o domínio da solução, priorizando a construção de programas e apenas, posteriormente, passam a lidar com o domínio do problema, tratando, por exemplo, com elicitação dos requisitos. Esta estratégia curricular não apenas contraria a forma canônica do processo de resolução de problemas, no qual, primeiramente, entende-se o problema e somente depois soluciona-o, como também não possibilita o satisfatório desenvolvimento das habilidades para especificação de problemas.

Além de analisar relatos da literatura [Winbladh 2004], nós realizamos duas investigações que apontam deficiências dos alunos em elicitar e especificar problemas - (i) um estudo de caso com alunos iniciantes de programação [Mendonça 2008b] e (ii) uma pesquisa do tipo *survey* com professores de computação e profissionais de TI [Mendonça 2008a].

A partir dos resultados dessas investigações e da análise da literatura [Deek 1997, Janzen and Saiedian 2006, Winbladh 2004], propusemos uma abordagem pedagógica, a ser implementada no contexto de alunos iniciantes, que possibilita o desenvolvimento das habilidades para especificação de problemas em conjunto com as boas práticas de desenvolvimento de *software*, como por exemplo, interação com cliente e escrita de casos de testes.

Nas Seções 2 e 3, respectivamente, apresentamos os resultados obtidos com o estudo de caso e a pesquisa *survey*. Na seção 4 descrevemos a abordagem pedagógica proposta, assim como a metodologia para avaliá-la e posteriormente, apresentamos as considerações finais sobre a pesquisa em pauta.

2 Estudo de Caso com Alunos Iniciantes de Programação

No período de 18/02/2008 a 10/04/2008 foi realizado um estudo de caso com alunos iniciantes de programação, do curso de Ciência da Computação, da Universidade Federal de Campina Grande.

A investigação foi realizada em duas fases: (i) estudo piloto realizado com 4 alunos e o estudo de caso, propriamente dito, realizado com 10 alunos. Cada aluno foi observado individualmente e cada sessão durava em média duas horas. Após a resolução de um problema de programação era realizada uma entrevista com o aluno.

O planejamento, execução e registro do estudo de caso seguiu as orientações de Kitchenham *et al.* [Kitchenham et al. 1995] e Yin [Yin 1984]. A descrição detalhada da investigação incluindo trechos de diálogos, versões de código e análise dos resultados obtidos estão descritos em Mendonça [Mendonça 2008b].

O estudo de caso teve por objetivo responder as seguintes questões de pesquisa: (Q1) Quais estratégias os alunos iniciantes empregam para especificar problemas? (Q2) Quais as dificuldades enfrentadas por estes alunos ao lidar com especificação de problemas? (Q3) Como os alunos iniciantes conduzem o diálogo para obtenção e esclarecimento das informações sobre o problema?

O procedimento empregado na investigação é descrito como segue: o aluno recebia um problema "mal definido"[1] e deveria solucioná-lo. A especificação completa do problema era de conhecimento apenas do investigador que fazia o papel de cliente. Para este problema foram definidos previamente 8 casos de testes. Caso o programa feito pelo aluno não passasse nos casos de testes previstos, o aluno deveria corrigi-lo. Cada entrega do programa ao investigador era contabilizada como uma versão. Não havia limitações quanto ao tempo e o aluno podia fazer perguntas a qualquer momento. A atividade era finalizada quando o programa passava em todos os casos de testes previstos.

2.1 Resultados da Investigação

Em média os alunos levaram 86 minutos para concluir o programa com aprovação em todos os casos de testes. Nenhum deles conseguiu elicitar todos os requisitos do problema na primeira versão do programa. Somente 20% dos alunos conseguiram finalizá-lo na segunda versão, sendo que 50% deles precisaram gerar mais de 4 versões para atender a todos os requisitos. Realizaremos a descrição dos resultados tomando como referência as questões de pesquisa.

- [Q1] - Quais estratégias os alunos iniciantes empregam para especificar problemas?

Verificamos na investigação que a maioria dos alunos, após a leitura do enunciado do problema, partiam diretamente para a construção do programa ou para escrita de um algoritmo. Assim, não realizavam uma análise exploratória do problema e a especificação do mesmo era construída no decorrer da codificação da solução.

- [Q2] - Quais as dificuldades enfrentadas pelos alunos ao lidar com especificação de problemas?

Os alunos apresentaram dificuldades em realizar leitura exploratória, interpretação do enunciado do problema, análise das restrições do problema, aquisição e registro das informações.

- Leitura Exploratória. Os alunos não exploravam previamente o enunciado do problema.

Os alunos apresentaram dificuldades em identificar informações faltantes, principalmente as que dizem respeito às entradas, saídas e restrições do problema. Em geral, questionavam apenas sobre as regras dos cálculos que deveriam ser feitos. Além disso, não exploravam o significado dos termos envolvidos no problema, mesmo que os desconhecessem.

---

1 Problema cuja descrição não é completa, podendo conter ambigüidades, contradições, falta de informações e/ou informações irrelevantes.

- Interpretação. A maioria dos alunos ignorava a existência de um cliente e atribuía um significado pessoal para a informação.

A maioria dos alunos construiu o programa seguindo uma interpretação pessoal do que devia ser feito, ignorando a perspectiva do cliente. Por exemplo, os alunos não questionavam sobre as saídas e em virtude disso duas situações ocorriam: ou imprimiam apenas o dado que era explicitamente descrito no enunciado, ou imprimiam as informações que eles próprios julgavam necessárias, impondo uma interpretação pessoal sobre o problema.

- Aquisição de informações. As dificuldades com leitura exploratória e interpretação comprometeram a aquisição de informações.

Obviamente, se o aluno não percebe a necessidade de informações ele também não desenvolve as estratégias para adquiri-las. As estratégias para aquisição de informações foram pautadas no diálogo e na submissão do programa. A maioria dos alunos submetiam o programa esperando que, quando houvesse erros, indicações do que precisava ser corrigido fossem apontadas. Essa é uma característica que, entre outras coisas, revela a falta de reflexão do aluno sobre o problema. Descreveremos as estratégias de diálogo posteriormente, em resposta a próxima questão de pesquisa.

- Análise das restrições do problema. Os alunos preocupavam-se com a verificação dos casos óbvios.

Os alunos apresentaram dificuldades em especificar as restrições do problema. Os testes, que poderiam ajudar nessa descoberta, eram concebidos para os casos óbvios e não exploravam as restrições e situações de erros no programa. Na maioria dos casos, os alunos concebiam dois ou três testes e o faziam no final da implementação.

- Registro das Informações. Os alunos não re-elaboravam a definição do problema.

Os registros das novas informações eram feitos através de pequenas anotações em papel sem preocupação com estilo e organização. As anotações utilizavam abreviações, simbologia matemática e setas relacionando blocos de informação. Em função do mau registro da informação, as dúvidas se repetiam em diferentes momentos da resolução de problemas. Os alunos não tinham a consciência de que o registro dos requisitos estabelecem um "contrato" definindo *o que* deve ser feito.

- [Q3] - Como os alunos iniciantes conduzem o diálogo para obtenção e esclarecimento das informações sobre o problema?

No geral, o diálogo ocorria sob demanda, com poucas perguntas sendo feitas inicialmente e as demais sendo realizadas à medida que o aluno codificava o programa. Nem todas as perguntas feitas eram direcionadas à especificação do problema, algumas delas visavam esclarecer aspectos de implementação. Em virtude de não perceberem as informações que precisavam ser esclarecidas, os alunos questionavam pouco. Dada a quantidade de perguntas imprecisas, constatamos a inabilidade dos alunos em elaborar perguntas objetivas para o esclarecimento das informações.

3 *Survey* com Professores de Computação e Profissionais de TI

Uma investigação mais abrangente foi realizada por meio de uma pesquisa do tipo *survey* no período de 06/04 a 08/05/2008. O *survey* foi do tipo *case control* [Kitchenham and Pfleeger 2002a], destinado a professores de computação e profissionais de TI. A

coleta de dados ocorreu por meio de questionários, não supervisionados, disponibilizados na Web. A investigação contou com a participação de 205 professores e 196 profissionais de TI de todas as regiões do país. O planejamento, execução, análise e registro dos resultados seguiu as orientações de Kitchenham and Pfleeger [Kitchenham and Pfleeger 2002a, Kitchenham and Pfleeger 2002b, Kitchenham and Pfleeger 2003] e foram descritos detalhadamente em Mendonça [Mendonça 2008a].

3.1 Resultados e Discussão

Ao avaliarem os alunos egressos dos cursos de computação com relação ao desenvolvimento das habilidades para especificação de problemas, 51,5% dos profissionais acreditam que poucos saem com essas habilidades e 58,6% dos professores acreditam que apenas parte dos alunos saem com essas habilidades.

Do total de professores, 53,9% deles afirmaram que os alunos freqüentemente apresentam dificuldades em compreender os problemas propostos na disciplina. Do mesmo modo, 52,6% dos profissionais, afirmaram que freqüentemente os desenvolvedores de software com os quais trabalham apresentam dificuldades em compreender os problemas propostos pelos clientes.

As sugestões deixadas pelos professores concentram-se em três aspectos: (i) a necessidade de conscientização dos próprios professores para o problema em questão; (ii) descrição das dificuldades percebidas nos alunos com relação à interpretação de textos, expressão oral e escrita; (iii) a necessidade de orientação pedagógica que auxilie o professor a lidar com esse problema no contexto de sala de aula.

Verificamos com a pesquisa que, tanto no contexto acadêmico quanto no profissional, são freqüentes as dificuldades com especificação de problemas. A investigação demonstrou ainda que professores e profissionais de TI consideram que as habilidades para entendimento de problemas não foram suficientemente desenvolvidas na graduação, haja vista a avaliação dos egressos feita por eles. Tal fato, evidencia a necessidade de melhorias na formação dos estudantes de computação.

4 Proposta de uma Abordagem Pedagógica

Nesta seção descrevemos uma abordagem pedagógica que tem por objetivo desenvolver as habilidades para especificação de problemas e as boas práticas de programação. A abordagem pressupõe que o entendimento do problema ocorre de maneira iterativa e incremental em duas fases que chamaremos de *entendimento inicial* e *refinamento da compreensão* (Figura 1). Problemas "mal definidos" devem ser propostos a fim de permitir aos estudantes o desenvolvimento das habilidades de leitura exploratória, análise, interpretação e diálogo.

Na fase de *entendimento inicial* o estudante toma conhecimento do enunciado do problema e deve produzir um artefato, que chamamos de *especificação inicial*. Este artefato deve conter os requisitos desejados pelo professor (cliente), os quais são elicitados por meio de diálogo, obedecendo o processo natural de perguntas e respostas.

A intenção nessa fase é que aluno vivencie um contexto mais próximo do mundo real e exercite as atividades de elicitação e especificação de requisitos. O documento de especificação não obedece um *template*. Os alunos devem expressar a especificação do problema em um formato textual, expandindo o enunciado anteriormente apresentado pelo professor (cliente). A opção por adotar um documento textual sem um formato

pré-definido justifica-se pela simplicidade nessa etapa. Orientações quanto ao estilo e organização do documento de especificação são fornecidos aos alunos.

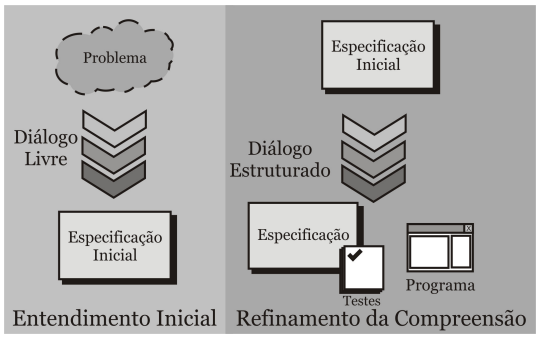

Figura 1 – Abordagem Pedagógica.

Como foi observado no estudo de caso, os alunos apresentaram dificuldades na elaboração de perguntas. Na fase de entendimento inicial, na qual o diálogo é livre, é importante que o aluno faça questionamentos com qualidade. Para isto, estratégias de diálogo foram desenvolvidas com o objetivo de orientar os estudantes na reflexão sobre o problema, na habilidade de expressar claramente as dúvidas e na aquisição de autonomia com relação a própria aprendizagem. A estratégia de diálogo está estruturada em três dimensões: *propósito*, *forma* e *conteúdo* (Figura 2) e foi inspirada no trabalho de Lane [Lane 2004].

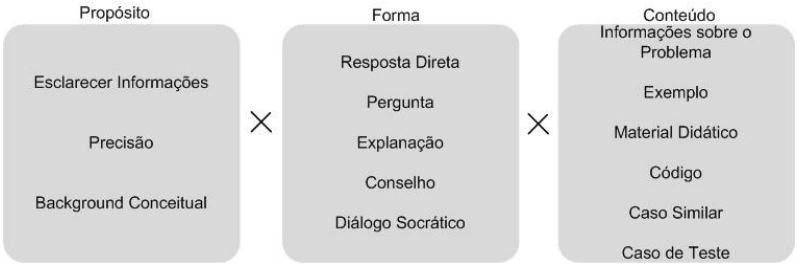

Figura 2 – Estratégias de Diálogo.

O *propósito* diz respeito ao objetivo pedagógico que se deseja alcançar. A *forma* indica a estrutura da interação e o *conteúdo* refere-se ao teor da interação. Por exemplo, no caso em que o aluno faz uma pergunta sobre um determinado requisito, o *propósito* do diálogo é o de esclarecer informações, isto pode ser feito na *forma* de uma resposta direta com informações sobre o problema (*conteúdo*).

Quando o aluno consegue estabelecer um acordo com o professor (cliente) sobre o que deve ser feito, expressando-o por meio do documento de *especificação inicial*, ele passa para a fase de *refinamento da compreensão* que ocorre gradativamente à medida que o aluno vai construindo a solução do problema. Nesta fase, o diálogo continua sendo a "força motriz" do processo de elicitação de requisitos, porém ele deve, agora, processar-se de maneira mais estruturada, na forma de *casos de testes*. Testes, portanto, são entendidos como uma pergunta, escritos em uma sintaxe estruturada, que questiona o professor (cliente) quanto aos requisitos e restrições do problema.

Nesta fase, motivamos a prática de mais uma atividade da engenharia de *software* - testes. Entretanto, não a utilizamos seguindo os rigores de TDD (*Teste Driven Development*) [Beck 2002], isto é, que os testes sempre precedam a codificação. Nesta abordagem, incentivamos a criação de casos de testes de maneira contínua e antecipada, porém não adotamos o rigor de que os mesmo sejam concebidos sempre antes da codificação. O mais importante nesta fase é que os alunos utilizem testes como

práticas efetivas que auxiliam na confirmação e descoberta de requisitos, na reflexão sobre corretude e situações de erro que podem ocorrer no programa.

Para estabelecer uma continuidade nas atividades desenvolvidas pelos alunos, eles são motivados a expressarem em casos de testes as declarações de requisitos presentes no documento de especificação inicial. Isto contribui para que o aluno passe a dialogar de maneira mais estruturada e que continue especificando novos requisitos.

Nessa fase, a compreensão do problema é mensurada pelo número de testes cobertos pelo aluno. Caso ele submeta o programa a avaliação do professor sem que todas as funcionalidades tenham sido atendidas, um novo ciclo de *refinamento da compreensão* é iniciada. Nesta fase, além dos testes, o aluno deve complementar o documento de especificação e produzir o programa.

Nas duas fases da abordagem, o estudante é motivado a refletir sobre o problema antes de simplesmente codificá-lo. Para produzir o documento de especificação inicial, por exemplo, o estudante deverá explorar o enunciado, detectar as informações faltantes, contraditórias ou ambíguas, deve esclarecer tais informações por meio do diálogo e registrá-las de maneira organizada no documento de requisitos. Na segunda fase, além de refinar essas atividades, o estudante desenvolverá a habilidade de testes e, conseqüentemente, a análise das restrições do problema. Acreditamos que essa intervenção pedagógica irá minimizar as deficiências apresentadas pelos alunos, conforme descritas no estudo de caso (Seção 2).

4.1 Avaliação

A avaliação da abordagem proposta será feita por meio de dois estudos de caso: o primeiro, com o objetivo de avaliar os efeitos da abordagem, as dificuldades de aplicação, a adequação das variáveis de controle e das formas de medir os resultados. Esta investigação funcionará como um estudo piloto que orientará a formalização do segundo estudo de caso, a ser realizado com dois grupos - um de controle e outro experimental.

O primeiro estudo de caso será realizado com a turma de alunos iniciantes do Curso de Ciência da Computação da Universidade Federal de Campina Grande no período de setembro de 2008 a fevereiro de 2009. Python será a linguagem de programação utilizada e os testes serão feitos usando expressões *assert*, próprias da linguagem.

5 Considerações Finais

Neste artigo apresentamos evidências de uma problemática na formação do estudante de computação - falta de habilidades para especificação de problemas. Estas evidências foram obtidas por meio de duas investigações: um estudo de caso com alunos iniciantes de programação e um *survey* realizado com professores e profissionais da área. A dificuldade com elicitação de requisitos, entretanto, não é característica de estudantes brasileiros. Outros autores relatam problemas semelhantes, em diferentes países, inclusive, alertando para o *gap* entre a formação dos estudantes e as exigências da indústria de software [Winbladh 2004, O'Leary et al. 2006, Garg and Varma 2008].

Embora os resultados obtidos não possam ser, facilmente, generalizados em termos estatísticos, eles contribuíram para uma discussão inicial sobre o tema e permitiram observar comportamentos e algumas dificuldades dos alunos ao realizarem atividades de especificação. A observação desses fenômenos é especialmente importante no contexto de computação porque grande parte dos problemas são resolvidos fora do

horário de sala de aula e em virtude disto, os professores realizam pouco acompanhamento do processo de resolução, tendo mais acesso ao produto final que é entregue pelo aluno na forma de um programa ou algoritmo.

A fim de contribuir para a melhoria do processo ensino-aprendizagem, propomos uma abordagem pedagógica que possibilita ao aluno, já na série inicial, conviver com problemas "mal definidos" e desenvolver habilidades para solucioná-los, exercitando atividades da engenharia de software, tais como, testes, elicitação e especificação de requisitos.

O diferencial desta abordagem reside no fato de congregar diferentes atividades de desenvolvimento de *software* e conceber o entendimento do problema como uma atividade iterativa e incremental, no contexto de alunos iniciantes. Outros trabalhos já enfatizaram o uso de testes [Janzen and Saiedian 2006] e especificação [Brown 1988] com alunos iniciantes, mas não o fizeram de forma conjunta. Deek [Deek 1997] propôs um modelo para resolução de problemas e desenvolvimento de programas que congrega todas as etapas de desenvolvimento de *software*. Entretanto, o modelo proposto por ele assemelha-se ao desenvolvimento em cascata, em que o entendimento e solução dá-se de maneira seqüencial e progressiva, contrariando a perspectiva incremental e iterativa do processo de desenvolvimento de *software*. Além disso, na fase de formulação do problema, o autor baseia-se na estratégia de extrair do enunciado os elementos relevantes do problema. Nossa perspectiva é de que o entendimento baseia-se não somente na extração de informações do enunciado, mas também em um processo de interação com o cliente (professor) por meio do diálogo.

Acreditamos que a abordagem proposta traz três contribuições significativas: (i) proporcionar ao aluno iniciante o desenvolvimento de outras habilidades além da codificação, consolidando, já na série inicial, algumas práticas da engenharia de *software*; (ii) sistematizar um procedimento de ensino-aprendizagem que pode ser replicado em outras universidades; (iii) colaborar por meio dos resultados obtidos para uma reflexão futura sobre a integração gradativa do currículo, na qual os alunos não precisam mais esperar até as disciplinas de Análise ou Engenharia de Software para desenvolver habilidades de especificação.

Cabe por fim ressaltar que a programação é considerada um dos sete grandes desafios em educação para computação [McGettrick et al. 2004]. Dentre os desafios citados estão o de desenvolver estudos sistemáticos para melhorar a compreensão das habilidades humanas neste domínio e proporcionar metodologias de ensino que desenvolvam as habilidades necessárias para o desenvolvimento de programas. Acreditamos que com a avaliação da abordagem proposta estaremos colaborando para a construção de soluções e superação desses desafios.

Referências

Beck, K. (2002). Test-Driven Development By Example. Addison Wesley.

Brown, D. A. (1988). Requiring CS1 students to write requirements specifications: a rationale, implementation suggestions, and a case study. In SIGCSE '88: Proceedings of the nineteenth SIGCSE technical symposium on Computer science education, pages 13–16, New York, NY, USA. ACM.

Deek, F. P. (1997). An integrated environment for problem solving and problem development. PhD thesis, New Jersey Institute of Technology, Newark.

Garg, K. and Varma, V. (2008). Software engineering education in india: Issues and challenges. IEEE 21st Conference on Software Engineering Education and Training (CSEET '08), pages 110–117.

Janzen, D. S. and Saiedian, H. (2006). Test-driven learning: intrinsic integration of testing into the cs/se curriculum. In SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education, pages 254–258, New York, NY, USA.

Kitchenham, B. and Pfleeger, S. L. (2003). Principles of survey research part 6: data analysis. SIGSOFT Softw. Eng. Notes, 28(2):24–27.

Kitchenham, B., Pickard, L., and Pfleeger, S. L. (1995). Case studies for method and tool evaluation. IEEE Softw., 12(4):52–62.

Kitchenham, B. A. and Pfleeger, S. L. (2002a). Principles of survey research part 2: designing a survey. SIGSOFT Softw. Eng. Notes, 27(1):18–20.

Kitchenham, B. A. and Pfleeger, S. L. (2002b). Principles of survey research: part 3: constructing a survey instrument. SIGSOFT Softw. Eng. Notes, 27(2):20–24.

Lane, H. C. (2004). Natural Language Tutoring and the Novice Programmer. PhD thesis, University of Pittsburg. Department of Computer Science.

McGettrick, A., Boyle, R., Ibbett, R., Lloyd, J., Lovegrove, G., and Mander, K. (2004).

Grand challenges in computing education. Technical report, British Computer Society.

MEC/SESU (1999). Diretrizes curriculares de cursos da Área de computação e informática. Technical report.

Mendonça, Andréa P. (2008a). Entendimento de problemas - um survey com profissionais de TI, professores e alunos de computação. Technical Report DSC/004/2008, Departamento de Sistemas e Computação. Coordenação de Pós-Graduação em Ciência da Computação. Universidade Federal de Campina Grande.

Mendonça, Andréa P. (2008b). Entendimento de problemas: Uma investigação exploratória com alunos iniciantes de programação. Technical Report DSC/005/2008, Departamento de Sistemas e Computação. Coordenação de Pós-Graduação em Ciência da Computação. Universidade Federal de Campina Grande.

O'Leary, C., Lawless, D., Gordon, D., Haifeng, L., and Bechkoum, K. (2006). Developing a software engineering curriculum for the emerging software industry in china. In CSEET '06: Proceedings of the 19th Conference on Software Engineering E-ducation & Training, pages 115–122, Washington, DC, USA. IEEE Computer Society.

Winbladh, K. (2004). Requirements engineering: closing the gap between academic supply and industry demand. Crossroads, 10(4):4–4.

Yin, R. K. (1984). Case Study Reseach Design and Methods. Sage Publications.