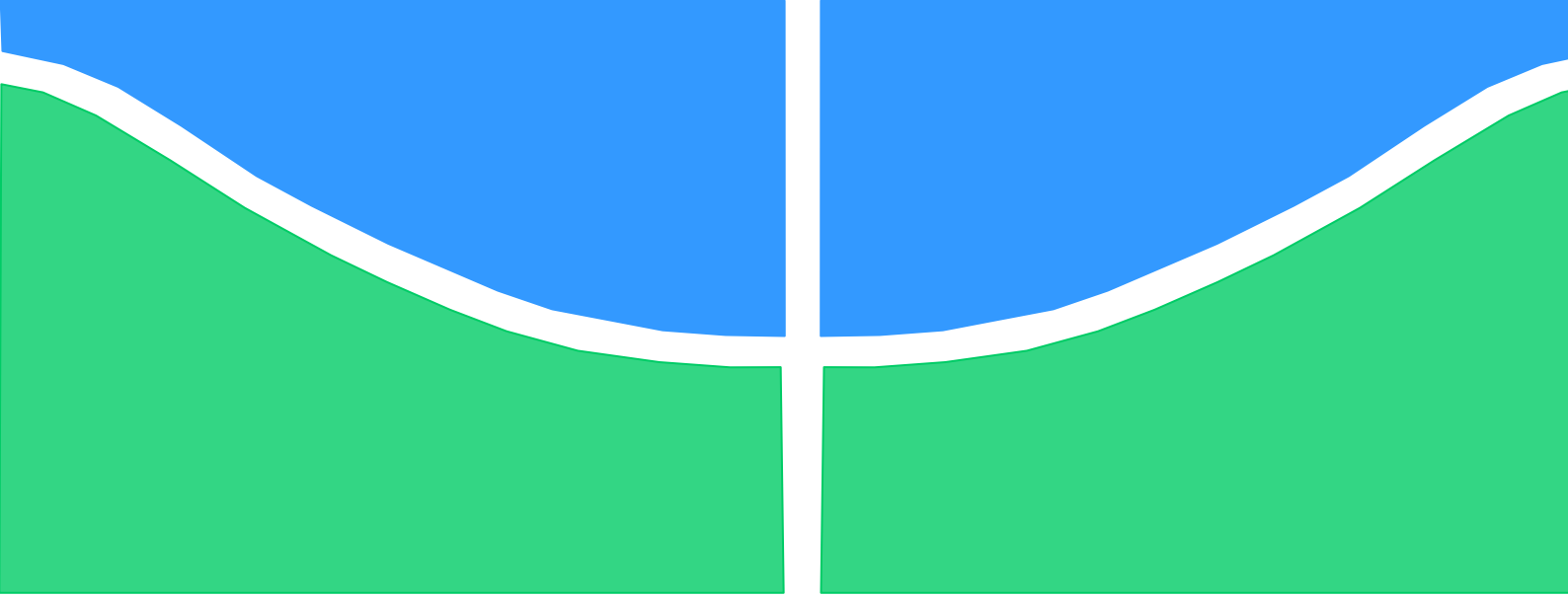

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Curso de Engenharia de Software

FATORES RELACIONADOS AO ENSINO DE PROGRAMAÇÃO NAS DISCIPLINAS INTRODUTÓRIAS NO CURSO DE ENGENHARIA DE SOFTWARE

Autor: Guilherme de Lima Calixto

Orientador: Dr. Wander C. M. Pereira da Silva

Brasília, DF

2015

GUILHERME DE LIMA CALIXTO

FATORES RELACIONADOS AO ENSINO DE PROGRAMAÇÃO NAS DISCIPLINAS INTRODUTÓRIAS NO CURSO DE ENGENHARIA DE SOFTWARE

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Orientador: Dr. Wander C. M. Pereira da Silva

Brasília, DF

2015

CIP – Catalogação Internacional da Publicação*

de Lima Calixto, Guilherme.

Fatores Relacionados ao Ensino de Programação: uma Análise das Disciplinas Introdutórias em Cursos de Engenharia / Guilherme de Lima Calixto. Brasília: UnB, 2015. 66 p. : il. ; 29,5 cm.

Monografia (Graduação) – Universidade de Brasília

Faculdade do Gama, Brasília, 2014. Orientação: Wander C. M. Pereira.

1. Aprendizagem de Programação 2. Computer Science Education I. C. M. Pereira, Wander. II. Dr.

CDU Classificação

Guilherme de Lima Calixto

Monografia submetida como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software da Faculdade UnB Gama - FGA, da Universidade de Brasília, em 10/12/2015 apresentada e aprovada pela banca examinadora abaixo assinada:

Prof. Dr. Wander C. M. Pereira da Silva, UnB/ FGA

Orientador

Profa. Dra. Edna Dias Canedo, UnB/ FGA

Membro Convidado

Profa. M.Sc. Elaine Venson, UnB/ FGA

Membro Convidado

Brasília, DF
2015

AGRADECIMENTOS

Queria agradecer aos meus amigos por me motivarem a continuar sempre buscando o melhor de mim mesmo. Queria agradecer também aos meus pais por trabalharem duro, sem nunca deixar faltar nada pra mim, e sempre me fornecerem a melhor educação possível. Queria agradecer também a minha namorada, que revisou esse texto e me ajuda e apoia todo dia.

Dedico esse trabalho a todo o meu caminho percorrido na faculdade, que foi muito árduo e desmotivador em vários momentos, mas recompensador em outros inúmeros. O conhecimento e aprendizado são de fato muito especiais.

O último agradecimento vai ao meu orientador Wander, por me guiar e ajudar. Além de não desistir de mim, o que é o mais importante.

RESUMO

O processo de ensino-aprendizagem de introdução à programação pode ser uma tarefa penosa para estudantes de graduação, gerando altos índices de reprovação e abandono de curso. Diante disso, o presente trabalho teve por objetivo identificar os principais fatores que estão relacionados ao processo de ensino-aprendizagem de programação em disciplinas introdutórias de cursos de graduação em engenharia de software. Para isso, foi realizado um estudo de caso na Faculdade UnB Gama (FGA), com 233 estudantes matriculados na disciplina introdutória de programação. Os resultados mostraram que de acordo com percepção dos estudantes, aprender a programar é sim uma tarefa complexa e difícil, que exige cuidado ao ser abordado tanto dentro da sala de aula quanto fora dela. Finalmente, concluiu-se que essa percepção em relação ao tema pode influenciar no número de estudantes que optam em seguir pela carreira de engenheiro de software.

Palavras-chave: Aprendizagem de Programação. Engenharia de software.

ABSTRACT

The process of teaching-learning programming introduction can be a grueling task for graduate students, generating high rates of failure and course abandonment. Therefore, this study is aimed to identify the main factors that are related to the process of teaching and learning in introductory programming courses in software engineering degrees. For this, we conducted a case study at the Faculdade UnB Gama (FGA), with 233 students enrolled in an introductory programming course. The results showed that according to the student's perception, learning to program is rather a complex and difficult task, requiring caution to be addressed, both within the classroom and outside it. Finally, it was concluded that this perception of the theme can influence the number of students who choose to follow the software engineering career.

Keywords: Programming Learning, Software Engeneering

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

1.1. CONTEXTO

O aprendizado de programação é considerado parte fundamental nos cursos de graduação na área de computação. O sucesso nesse aprendizado é decisivo na formação do egresso, tornando-se um diferencial ao longo de sua carreira profissional.

A Sociedade Brasileira de Computação (SBC), de longa data, enfatiza que as disciplinas ligadas à aprendizagem de programação, como por exemplo: *Linguagens de programação, estruturas de dados, e projeto e análise de algoritmos* deveriam ser obrigatórias nos cursos de graduação em computação (SBC, 1999).

No Projeto de Resolução que institui as Diretrizes Curriculares Nacionais para os cursos da área de Computação, constante no Parecer CNE/CES nº 136/2012[1], a necessidade de aprender a programar fica bastante evidente quando aborda o perfil desejado dos alunos egressos dos cursos de computação:

> "**Art. 5º** Os cursos de bacharelado e licenciatura da área de Computação devem formar egressos que revelem pelo menos as competências e habilidades comuns para:
>
> I – identificar problemas que tenham solução algorítmica;
>
> II – (...)
>
> III – resolver problemas usando ambientes de programação;" (BRASIL, 2012, p. 22).

Considerando a centralidade e a importância de aprender a programar, se pode afirmar que parte expressiva da vida acadêmica do estudante de graduação em computação será aplicada na criação ou interpretação de código, programando soluções para determinados problemas. E, mesmo quando não se está programando diretamente, ter domínio sobre programação é fundamental na complementação de outros conceitos explorados nesses cursos.

[1] As DCNs abrangem 05 cursos: Bacharelado em Ciência da Computação, Bacharelado em Engenharia da Computação, Bacharelado em Engenharia de Software, Bacharelado em Sistemas de Informação, e Licenciatura em Computação

Por outro lado, os avanços da sociedade atual – denominada por Drucker (1999) de *sociedade da informação e do conhecimento*, é grandemente alimentado por ferramentas baseadas em software. Como afirma Silveira (2004, p. 24)

> "[…] com sua penetração em todos os segmentos do cotidiano que utilizam processamento e transmissão de informações, o software adquiriu uma grande importância como catalisador da possibilidade de se concentrar ou desconcentrar riqueza e poder."

Na sociedade do conhecimento os ativos mais valiosos de uma organização, de negócios ou não, serão seus trabalhadores do conhecimento e sua produtividade (DRUKER, 1999). Para sobreviver nesse novo cenário as empresas necessitam de bons desenvolvedores de soluções baseadas em algoritmos. Portanto, aprender a programar em alto nível confere ao profissional um alto valor de mercado.

No Brasil, o mercado de trabalho para programadores tem apresentado crescimento acentuado. Um estudo do governo brasileiro de 2005 mostrou que existiam 17 mil vagas de trabalho não preenchidas na indústria nacional de software (FIGUEIREDO e cols. 2010). Esse levantamento aponta ainda a necessidade de se aumentar não só a quantidade de profissionais, como melhorar a sua qualidade.

No entanto, a despeito dessa demanda de mercado crescente por profissionais de TI, formar bons profissionais que saibam programar em alto nível ainda é uma tarefa desafiadora. A literatura da área tem evidenciado o quanto é árdua a tarefa de ensinar e aprender a programar (ROBINS et. al., 2003).

Em torno da questão do ensino aprendizagem de programação, surgiram duas grandes perspectivas de pesquisas: (1) a da Engenharia de Software, e (2) àquelas ligadas à Psicologia e Educação. A perspectiva da engenharia de software tem focado nos profissionais experientes que já estão, de certa forma, envolvidos em trabalhos profissionais, e a da psicologia/educação que focaliza nos estudantes iniciantes de disciplinas introdutórias de programação (ROBINS et. al., 2003).

Na perspectiva psicológica e educacional, que é o contexto do presente trabalho, é possível agrupar as pesquisas em três grandes blocos: (1) as que evidenciam o quão difícil é a tarefa de aprender a programar nas disciplinas introdutórias; (2) àquelas que investigam os requisitos cognitivos para a aprendizagem, e (3) as que buscam identificar quais fatores estão relacionados ao sucesso ou insucesso do processo de ensino-aprendizagem em programação.

Jens Bennedsen e Michael Caspersen (2007) em 63 instituições de ensino superior ao redor do mundo, indicam que uma média de 33% dos alunos não obtém sucesso no curso de introdução a programação.

Soloway et al. (1983) encontraram que apenas 38% dos alunos de um curso de computação foram capazes de escrever um programa simples para calcular a média de um conjunto de números. Essa tarefa é considerada o cerne da capacidade de um estudante que concluiu um semestre de programação.

McGettrick et al. (2004) observaram que os professores citam falhas em cursos introdutórios de programação e/ou desencanto com a programação como principais fatores subjacentes ao abandono de estudantes pobres em programas de licenciatura em computação.

Weinberg (1971, *apud* de CASTRO, 2011) afirma que a aprendizagem de programação é uma atividade cognitiva que requer alto nível de raciocínio abstrato. E, essa necessidade de abstração, com frequência provoca dificuldades nos alunos de graduação em computação, nas disciplinas introdutórias de programação, transformando a aprendizagem no tema em uma experiência penosa e desmotivadora.

Mas, além de ser uma atividade cognitiva complexa, quais fatores poderiam estar relacionados à dificuldade de programar? Nos últimos anos foram publicados diversos estudos indicando diferentes fatores que poderiam influenciar a aprendizagem de estudantes novatos em programação.

A literatura aborda questões relacionadas à: metodologia de ensino (ESTEVES et al., 2008); fatores internos e externos aos alunos (ROUNTREE et al., 2002); linguagem a ser ensinada (CHUN e RYOO, 2010); particularidades da disciplina programação em si (JENKINS, 2002); e fatores cognitivos (MARTINS et

al., 2010). Outros são focados em estudos de caso muito específicos, por exemplo, se uma determinada linguagem ou paradigma é mais indicado para o ensino (BHATTACHARYA et al., 2011); se as metodologias utilizadas ou ambientes computacionais criados facilitam a aprendizagem (GOMES e Mendes, 2010).

Outros estudos abordam a relação entre as características dos alunos e suas chances de falha ou sucesso em um curso introdutório a programação. A pesquisa investiga se há relação entre as experiências prévias do aluno (como cursos ou uso diário de computadores) e seu sucesso na disciplina (ROUNTREE et al*.,* 2002; BYRNE e LYONS, 2001).

No artigo intitulado *On the Difficulty of Learning to Program*, Tony Jenkins (2002) comenta sobre a existência de um grande número de pesquisas em computação educacional com foco em maneiras novas e pouco tradicionais para ensino de programação, contudo, apesar de considerar essas ideias, o autor questiona se elas são realmente mais eficientes do que a tradicional.

Finalmente, no Brasil esse tema ainda é pouco explorado na área de computação. No levantamento realizado para este trabalho não foi encontrada nenhuma pesquisa que objetivasse a validação dessas hipóteses levantadas pelos estudos citados na literatura internacional.

Dados com estudantes brasileiros seriam interessantes – principalmente pelas peculiaridades do ensino e a realidade brasileira, no sentido de avaliar se os fatores encontrados nos estudos internacionais são ou não válidos para os estudantes de programação no Brasil.

1.2 PROBLEMA

Diante desse contexto, o presente trabalho de conclusão de curso, tem como questão central de pesquisa a seguinte pergunta:

Quais fatores estão relacionados ao processo de ensino-aprendizagem de programação nas disciplinas introdutórias em um curso de graduação em engenharia de software?

1.3 OBJETIVOS

1.3.1 OBJETIVO GERAL

O objetivo desse trabalho é identificar os principais fatores relacionados ao processo de ensino-aprendizagem de programação em disciplinas introdutórias de cursos de graduação em engenharia de software, independente da metodologia e linguagens utilizadas.

1.3.2 OBJETIVOS ESPECÍFICOS

- Identificar na literatura os principais fatores de dificuldade envolvidos no processo de ensino-aprendizagem de programação;
- Sistematizar esses fatores em um questionário a ser aplicado e validado;
- Analisar como o caso estudado (alunos de disciplinas introdutórias em programação na Faculdade do Gama-FGA), se comporta em relação à questão.

1.4 ORGANIZAÇÃO DO DOCUMENTO

O presente trabalho encontra-se organizado nos seguintes capítulos:

- Fundamentação Teórica: um resumo teórico do que se encontra na literatura atual a respeito dos fatores relacionados à aprendizagem de programação, revisão sistemática, e construção de instrumentos de pesquisa;
- Metodologia: contém as explicações da metodologia aplicada no desenvolvimento das pesquisas teóricas e quantitativas;
- Análise dos Resultados: reune todos os resultados obtidos após as pesquisas exploratórias e explanatórias;
- Conclusão: conclusão dos resultados investigados no trabalho;

2. FUNDAMENTAÇÃO TEÓRICA

2.1 PROGRAMAR É DIFÍCIL?

As disciplinas iniciais de programação nos cursos de graduação em computação tem representado um desafio para os alunos que nela se inscrevem. Este desafio mostra-se em vários sentidos, pois além dos alunos que não obtém sua aprovação no fim do semestre, existem aqueles que abandonam a matéria, e os que, apesar do sucesso na aprovação, continuam sem saber programar.

Como já citado, Weinberg (1971, *apud* de CASTRO, 2011) afirma que, por ser uma atividade cognitiva que requer alto nível de raciocínio abstrato, a aprendizagem de programação transforma as disciplinas introdutórias de programação em uma experiência penosa e desmotivadora para os alunos iniciantes.

Os números apresentados em diversos estudos evidenciam que aprender a programar é uma dificuldade para grande parte dos estudantes iniciantes. Por exemplo, Jens Bennedsen e Michael Caspersen (2007) promoveram uma pesquisa com o intuito de verificar se o consenso geral a respeito dos cursos de introdução a programação possuírem um alto índice de abandono e falha é verídico, já que poucos estudos abrangentes parecem se basear em estatítiscas oficiais. Sendo assim os autores enviaram e-mails solicitando dados a respeito dos índices de abandono e aprovação para mais de 500 faculdades ao redor do mundo. Um total de 63 instituições responderam suas solicitações, e os autores verificaram que, em média, 33% dos alunos não obtém sucesso no curso de introdução a programação.

Corroborando esses números, Soloway et al. (1983) encontraram que apenas 38% dos alunos de um curso de computação foram capazes de escrever um programa simples para calcular a média de um conjunto de números.

McGettrick et al. (2004) observaram que os professores citam falhas em cursos introdutórios de programação e/ou desencanto com a programação como principais fatores ao abandono de estudantes pobres em programas de licenciatura em computação.

Guzdial (2010) mostra que os estudantes que já estão cursando a matéria introdutória de programação ou até mesmo aqueles que já a completaram possuem grande dificuldade em ler, entender e escrever código.

Mendes et al. (2012) apontam que a Universidade de Coimbra, e muitas outras instituições de ensino, sofrem de altos índices de reprovação e abandono em seus cursos iniciais de programação.

Apesar de haver abandonos em qualquer momento nos cursos de graduação em computação (por exemplo, em decorrência dos atritos e dificuldades apresentadas ao longo da graduação), o índice é muito superior durante o primeiro ano de curso, de acordo com Bhattacharya et al. (2011). Muitos estudantes vêem a parte prática de ciência da computação como tediosa, chata, e irrelevante, desenvolvendo medo em decorrência de enxergar essa atividade como algo inacessível e intangível (BHATTACHARYA, 2011).

A comunidade acadêmica de computação sempre almejou entender porque programar é uma tarefa difícil e como as metodologias adotadas em cursos introdutórios influenciam no desempenho dos estudantes nessa tarefa. “Parece haver um consenso entre os professores e pesquisadores no que diz respeito à aprendizagem de programação como uma atividade não trivial, já que a mesma introduz uma nova série de requisitos cognitivos nas rotinas dos estudantes (...)” (MARTINS et al., 2010, p. 1317).[2]

2.2 QUAIS FATORES ESTÃO RELACIONADOS À DIFICULDADE DE APRENDER PROGRAMAÇÃO?

Toda a importância dada à aprendizagem de programação a torna objeto de assunto de inúmeros textos sobre suas particularidades e relevância, tornando o tema um dos mais abordados na *Computer Science Education* (SCE). Nesse sentido, um dos principais interesses é saber quais fatores estão relacionados à dificuldade de programar e qual o papel de cada um deles.

[2] There seems to be a consensus among teachers and researchers with regard to programming learning as a non-trivial activity, since it introduces a new series of cognitive requirements in the routines of students […]

A seguir serão descritos os principais fatores que são listados na literatura como relacionados ao processo de ensino aprendizagem de programação.

2.2.1 AS ESTRATÉGIAS OU FERRAMENTAS DE ENSINO

A literatura da área apresenta trabalhos que abordam maneiras diferentes e inovadoras, como alternativa às formas tradicionais de ensinar programação.

São artigos que tratam de estratégias alternativas, e que se utilizam de jogos ou linguagens visuais na tentativa de despertar o interesse da classe (GIANNOTTI, 1987), além de relatos que apresentam ferramentas desenvolvidas com objetivo de auxiliar o professor nas aulas de programação – como a utilização de um ambiente 3D virtual (FUNABIKI et al., 2013; ESTEVES et al., 2008).

Guzdial (2010), também propõe a utilização de um paradigma visual, mas menciona que essa abordagem possui desvantagens. Uma das desvantagens é a de que um indivíduo sempre levará mais tempo para compreender o que está escrito em linguagem gráfica do que para compreender o mesmo em linguagem textual, independente da experiência da pessoa, conforme proposto por Thomas Green e Marian Petre (*apud* GUZDIAL, 2010).

Nessa mesma linha dois estudos apontam que o uso de estratégias e ferramentas alternativas não alteraram significativamente o padrão alcançado com a metodologia tradicional.

Um deles é relatado por Jenkins (2002), que afirma que apesar do *feedback* positivo dos alunos frente a essas novas abordagens, não há evidências de que estas metodologias ou ferramentas propiciem uma alta na taxa de aprovação ou uma diminuição no número de desistentes. Há menos evidência ainda de que os alunos de fato aprendam mais ou com maior qualidade através do uso desses artifícios.

O outro é o artigo intitulado *Algorithm Animator: A Tool For Programming Learning* de Giannotti (1987). Ele adota uma ferramenta que se utiliza da linguagem visual no ensino de programação, mas relata que após algumas aulas apenas uma minoria dos estudantes se manteve interessada em explorar corretamente o uso da

ferramenta, e que para os que não mantiveram essa postura o aproveitamente em termos de aprendizagem foi muito baixo, até mesmo para os conceitos mais básicos.

É importante ressaltar, no entanto, que os últimos estudos citados foram realizados já há algum tempo, podem ser considerados antigos, e isso pode ter influenciado suas conclusões. É aceitável afirmar que as estratégias e ferramentas de ensino de programação evoluíram e dispõem hoje de tecnologias mais avançadas. Mesmo assim, não foram encontrados estudos recentes conclusivos sobre a questão.

Levando essas informações em conta, no presente estudo optou-se por não incluir o fator ferramentas de ensino na pesquisa. Além do mais a adoção desse fator demandaria um estudo mais sofisticado, com um planejamento experimental envolvendo diversos recursos financeiros e materiais, reduzindo a viabilidade de sua aplicação em um Trabalho de Conclusão de Curso.

2.2.2 A LINGUAGEM DE PROGRAMAÇÃO

Em relação à questão se um tipo particular de linguagem de programação pode causar mais dificuldades para programadores iniciantes, Milne e Rowne, (2004) apontam que existem poucos trabalhos para que se possam obter conclusões sobre tal questão.

Dos estudos sobre as linguagens de programação, o que parece estar consolidado na literatura é que a compreensão do programa utlizado afeta a escolha da linguagem de programação e que a adoção de diferentes notações facilitam a compreensão de diferentes tipos de informações encontradas em programas (Wiedenbeck, 1999a; Wiedenbeck, 1999b; Good et al., 1997 *apud* Milne e Rowne, 2004).

Martins et al. (2010), afirmam que o maior problema da aprendizagem de computação não está na linguagem ou no paradigma de programação, mas sim na dificuldade em desenvolver competências que façam o estudante ser capaz de contextualizar o seu próprio conhecimento com o objetivo de resolver problemas. Esses autores defendem que é mais importante fazer o estudante perceber que programar é, acima de tudo, um exercício das habilidades mentais em um contexto que suporta o desenvolvimento de diversas habilidades cognitivas.

Dentre os argumentos que sustentam essa afirmação, existem os que Gomes e Mendes (2007) descrevem em seu texto *Learning to program – difficulties and solutions*. São eles:

- Ensinar não é algo personalizado, não existindo garantias de que todos os alunos vão se adaptar a metodologia de ensino escolhida e as estratégias utilizadas pelo docente.
- Os alunos não possuem antecedentes e aptidões padronizados, podendo possuir deficiências em matemática ou lógica.

Os trabalhos encontrados na literatura da área não permitem estabelecer se há relação, e se há, qual seria essa relação, entre a aprendizagem de programação e os temas “estratégias ou ferramentas de ensino” e o “tipo de linguagem de programação”.

Novamente, considerando essas informações e ainda as dificuldades metodológicas na operacionalização dos mesmos, a presente pesquisa também optou por não incluir o fator no seu escopo.

2.2.3 A MOTIVAÇÃO

Segundo Jenkins (2002) uma variedade de fatores motivacionais leva alguém a se matricular em um curso de computação. Essa motivação pode ser classificada em dois tipos: intrínseca ou extrínseca. A motivação intrínseca é aquela apresentada por aqueles que realmente possuem interesse no assunto computação, e a extrínseca é a apresentada por aqueles que desejam por meio da computação obter uma carreira lucrativa por exemplo.

A motivação e seu tipo parecem ser um fator decisivo, pois já foi mostrado que alunos que fazem um grande esforço para programar têm maiores chances de possuírem uma motivação extrínseca (JENKINS, 2002). Esse tipo de motivação, porém, não pode ser trabalhada, já que sua definição se dá antes do ingresso no curso.

No artigo intitulado *Predictors of Success and Failure in a CS1 Course,* Rountree et al. (2002) realizaram um questionário focado nos antecedentes dos alunos matriculados no curso introdutório de programação oferecido pela

Universidade de Otago. Esse questionário foi realizado durante dois semestres, e dentre as conclusões que os autores chegaram está a de que os alunos que possuíam a expectativa de tirar nota máxima no curso foram os que apresentaram a maior proporção de aprovados, e também a maior proporção com notas iguais ou acima de 70%.

Outro estudo interessante apresentado sobre o fator motivacional foi o realizado por Mendes et al. (2012) e discutido no artigo *Increasing Student Commitment in Introductory Programming Learning*. O estudo é a respeito do curso introdutório de programação na Universidade de Coimbra, curso este que antes era ministrado em turmas compostas por alunos não só de computação, mas também por alunos graduandos de outras áreas.

Mudanças foram introduzidas na tentativa de responder a uma pergunta, que segundo os autores do texto era recorrente nos alunos de anos anteriores: por que eu deveria aprender a programar?

Percebeu-se que muitos dos alunos não possuíam qualquer motivação para enfrentar as dificuldades inerentes a aprendizagem de programação. As mudanças feitas na estrutura do curso foram em sua grande maioria de natureza organizacional, e suas consequências de natureza pedagógica. O contexto do aprendizado era mais adequado às características dos indivíduos, resultando em maior motivação e um envolvimento mais profundo dos alunos (MENDES et al., 2012). Os resultados foram positivos, e são apresentados de maneira resumida a seguir:

- O índice de aprovados (aprovados/matriculados) aumentou em mais de 100%;
- Os alunos avaliaram o curso de maneira positiva em relação ao seu próprio aprendizado e a qualidade do curso;
- Alunos que já haviam feito o curso e falhado avaliaram a nova abordagem de maneira positiva.

A contextualização como forma de motivação também é um dos tópicos do trabalho de Guzdial (2010). Em sua investigação sobre os fatores que influenciam na aprendizagem, ele menciona o caso ocorrido na universidade *Georgia Institute of*

Technology, que a partir de 1999 começou a exigir introdução à computação de todos os cursos de graduação como arquitetura, artes e administração.

Em 2003 um experimento foi iniciado utilizando um curso de programação contextualizado em manipulação de mídia, cujo foco eram alunos de comunicação e artes, mas contendo todos os tópicos presentes no curso tradicional de introdução a computação (FORTE e GUZDIAL, 2010). Os resultados foram muito positivos, observando-se aumento nos índices de aprovação geral entre outros benefícios.

Da mesma maneira que é possível estimular o aluno de maneira a motivá-lo, estratégias devem ser adotadas de maneira a não causar o efeito oposto. Dependendo da abordagem de ensino adotada pelo professor, é possível que o aluno diminua sua motivação ao longo do curso; como quando se pede a um principiante em programação que inicie o desenvolvimento de um programa subitamente, sem modelos ou exemplos para o aluno se apoiar. Esse tipo de situação pode facilmente causar frustrações, o que leva a diminuição em sua motivação para aprender (CHANG et al., 2000).

Santos et al. (2011) e Zhang (2009) alegam que a diferença de ritmo e estilo de aprendizagem de cada aluno também é um fator importante a se levar em consideração, dado que as classes são heterogêneas, o ensino de programação deve ocorrer de maneira a respeitar essas diferenças.

Outro fator encontrado na literatura que pode gerar desmotivação do estudante é o apresentado por Castro et al. (2008), que indica que os alunos raramente aprendem técnicas de resolução de problemas em seus cursos introdutórios, e que os estudantes encontram grande dificuldade em aplicar seus conhecimentos anteriores. Isso acaba se tornando uma fonte de medo e frustração, resultando em evasão dos alunos. Esteves (2008) também cita a falta de habilidades em solução de problemas como um fator desmotivador.

No texto *Encouraging Programming Learning for Novices with Grouping and Convincing Opinions,* Phuong et al. (2009) argumentam sobre como, sem supervisão adequada disponível para auxiliar os principiantes nos momentos de dúvida e dificuldade, e prover *feedbacks* na resolução de exercícios; a motivação do aluno vai se degradando, podendo até levar ao abandono da classe.

Para os autores, quando o aluno em processo de aprendizagem necessita de ajuda, mas não a obtém, tem sua confiança prejudicada, pois não sabe para onde seguir. “A ajuda nas práticas iniciais é extremamente significante, pois o aprendiz não perde sua motivação em aprender, mas consegue solucionar suas dificuldades com a ajuda” (PHUONG et al., 2009, p.283, tradução nossa)[3].

Phuong et al. (2009) propõem então duas estratégias para solucionar o problema de falta de supervisão, e que podem ser aplicadas em qualquer tipo de curso de introdução a alguma linguagem de programação. As estratégias são promover o estudo em grupo, e o compartilhamento de conhecimento entre os alunos matriculados, pois de acordo com os autores essas simples abordagens promovem a homogeneização do conhecimento, e a melhora nos índices de motivação geral da turma.

A falta de respostas e *feedback* constantes como elemento desmotivador é um problema discutido também por Funabiki et al (2013), Zhang (2009), Alves et al. (2012), e Fernandez-Medina et al. (2013). Iniciantes possuem tendência a serem sem foco, e a falta de objetivos claros pode ser extremamente prejudicial ao autoaprendizado (ZHANG, 2009).

Martins et al. (2010) argumentam sobre a importância do papel do professor como agente motivador através da supervisão dos alunos. A supervisão do professor não deveria surgir apenas em momentos relacionados à prática de programação, mas, como educador, o docente deveria ficar atento às necessidades e problemas de seus alunos, intervindo e aconselhando quando necessário.

A sensibilidade do professor é essencial, sua capacidade de identificar as fontes de frustrações dos alunos, e reservar momentos das aulas para discutir e trabalhá-las pode ser um fator decisivo na manutenção da motivação dos estudantes (MARTINS et al., 2010).

Todos esses argumentos apontam para uma direção em comum, a de que alunos motivados tendem a se sair melhor nos cursos introdutórios de programação. As pesquisas indicam que o professor responsável pela matéria ao fazer com que os

[3] The help on the initial practices is extremely significant, because the student does not lose it’s motivation to learn, but instead can solve his difficulties with the help.

alunos compreendam que os obstáculos são transponíveis e que as habilidades exigidas podem ser desenvolvidas e aprimoradas, encorajando-os a progredir em uma série de competências acadêmicas para que eles possam se aperfeiçoar em seus estudos e em sua vida profissional futura.

Ainda há dificuldades em encontrar uma maneira que motive os estudantes em se envolver com o curso apesar de suas dificuldades, e a não desistir de tentar superá-las (MARTINS et al., 2010).

A motivação do aluno decorre, então, da relação de diversos fatores, tendo raízes na educação escolar e familiar do indivíduo. Porém, é um fator que pode ser trabalhado em sala de aula tanto no início quanto ao longo do curso, de maneira a atender as individualidades de cada aluno. Existem, por exemplo, os alunos que precisam se sentir contextualizados, outros precisam da sensação de que estão sendo supervisionados de maneira adequada, enquanto outros precisam perceber que estão evoluindo no assunto para ganharem confiança.

2.2.4 A ATIVIDADE DE PROGRAMAR

A atividade de programar em si apresenta especificidades próprias que dificultam o seu aprendizado. Trata-se de uma atividade que depende de conhecimentos correlatos como lógica e matemática, habilidade na resolução de problemas e capacidade de abstração.

Programar é um processo complexo, possuindo tarefas que devem ser desenvolvidas em diferentes níveis, exigindo uma série de habilidades como: leitura e compreensão, pensamento crítico e sistemático, etc. Aprender conceitos e métodos para construção de programas computacionais não é uma tarefa trivial, exigindo uma alta dose de raciocínio abstrato.

Eckerdal et al. (2005) relatam um estudo realizado em um curso para engenheiros na Suécia, que apesar de não possuir foco em programação, contemplava matérias de introdução a programação. O tema do estudo era o de tentar compreender o que pode ser considerado como “pensar programação”, já que registros na literatura indicavam que aprender a programar necessita de uma abordagem própria, diferente das demais disciplinas.

Dos alunos matriculados, 14 se voluntariaram para participar de uma entrevista com perguntas a respeito do que seria programar, qual sua utilidade, qual a diferença entre aprender a programar e aprender outras disciplinas, entre outras. Dentre o relatado pelos autores pode se destacar a conclusão de que na percepção dos entrevistados “programar é uma habilidade, mas que também requer uma compreensão profunda de conceitos abstratos” (ECKERDAL et al., 2005, p.140, tradução nossa)[4].

Para ser bem sucedido em um curso introdutório de programação, o aluno deve aprender um conjunto de conceitos básicos, a sintaxe de uma linguagem de programação, e desenvolver habilidades em solução de problemas usando a linguagem de programação para expressar soluções (SANTOS et al., 2013).

Aprender a programar não é apenas aprender a transcrever a solução de um problema para um texto de código usando estruturas de uma linguagem de programação, mas é também a habilidade de ler e compreender programas escritos por outros (JADZGEVIČIENĖ & URBONIENĖ, 2012).

Matthews et al. (2012) defendem que normalmente estudantes acham uma tarefa fácil aprender conceitos de programação e sintaxe individualmente, contudo problemas surgem quando diversos conceitos são colocados em conjunto com objetivo de escrever um programa.

Por outro lado, Jenkins (2002) considera que programar é um tema difícil por causa das seguintes características:

- Múltiplas habilidades: programar não é uma única habilidade, e não é também um simples conjunto de habilidades; as habilidades formam uma hierarquia;
- Múltiplos processos: programar envolve processos distintos. Sendo o mais difícil deles o primeiro, que é transformar uma especificação em algoritmo. O segundo seria mapear os passos desse algoritmo, criar uma espécie de receita. E o terceiro é passar esse mapeamento para a linguagem desejada (JENKINS, 2002).

[4] Programming is a skill, but requires also a deep understanding of abstract concepts

Zhang (2010) e de Barros et al. (2005) citam pesquisas no campo da psicologia da programação que estão de acordo com essas idéias. De acordo com essas pesquisas, são dois os desafios que um aprendiz em programação deve lidar:

- Aprender uma nova linguagem: o estudante deve aprender uma nova sintaxe e semântica;
- Aprender como programar uma solução dado um problema: o estudante deve aprender como transformar uma solução em um programa computacional (ZHANG, 2010).

Mesmo os alunos que possuem conhecimentos suficientes e sabem como aplicá-los de maneira a construir um algoritmo dado um problema, apresentam dificuldades em passar esse algoritmo para uma determinada linguagem (ESTEVES et al., 2008).

Alguns alunos que não possuem familiaridade com a língua inglesa também podem encontrar dificuldades, pois este é o idioma adotado pela maioria das linguagens de programação, na construção dos *statements* e nas palavras reservadas, por exemplo (CHUN; RYOO, 2010).

Devido à natureza abstrata dos conceitos de ciência da computação, os atuais cursos introdutórios de computação impedem o aprendizado do iniciante, diminuindo o interesse em um aprendizado mais profundo (WU, 2011).

De acordo com Godwin-Jones (2005), a única maneira de o estudante conseguir desenvolver uma mentalidade que consiga contemplar as especificidades próprias da atividade de programar, é ao entender como na sociedade atual a programação interconecta quase todas as atividades realizadas diariamente, tanto no profissional, quanto no particular.

Para Mark Guzdial (2010), o desafio enfrentado por aqueles que estão aprendendo a programar acontece pelo fato de que as linguagens de programação não são linguagens naturais. Ao contrário do que se está acostumado ao lidar no dia-a-dia, “programar é a manipulação de uma linguagem artificial inventada para um uso particular, um propósito relativamente não natural – dizer a um agente não

humano (computador) exatamente o que fazer" (GUZDIAL, 2010, p.114, tradução nossa)[5].

Todas as especificidades relacionadas à atividade de programar, e todo processo cognitivo que está por trás dessa atividade, tornam-a difícil de ser estudada e praticada pelos que estão iniciando sua prática (ECKERDAL et al., 2005; FORTE e GUZDIAL, 2005; PHUONG et al. 2009). Isso se deve parte entre as confusões que o aluno possui entre as etapas envolvidas na concepção de um programa, que exigem metodologias e abordagens diferentes ao serem exercitadas (PHUONG et al. 2009).

Finalmente, há que se considerar que as linguagens de programação foram desenvolvidas para fins profissionais, e não com enfoque no aprendizado em si. A complexidade alta de suas sintaxes difere do modo de pensar e da maneira com que os indivíduos se comunicam uns com os outros.

2.2.5 A RESOLUÇÃO DE PROBLEMAS

A resolução de problemas é uma atividade que regularmente oferece dificuldades ao estudante de programação computacional, principalmente aos novatos, observa Shadiev et al. (2013). Isso decorre da ausência de conhecimentos e habilidades prévias.

Essa opinião encontra apoio no trabalho de Santos et al. (2011), que argumentam que frequentemente os estudantes apresentam baixa capacidade de abstração e habilidades na solução de problemas.

No artigo *The Analysis of a Case Study for Group Programming Learning*, Castro (2008) argumenta sobre como a ausência de habilidades em solução de problemas pode gerar frustração.

Em um estudo realizado e detalhado no artigo *Studies and Proposals about Initial Programming Learning,* Gomes e Mendes (2010) descrevem sobre o experimento com estudantes de Ciência da Computação que falharam na aprovação do primeiro curso de programação.

[5] Programming is the manipulation of an artificial language invented for a particular, relatively unnatural purpose – telling a nonhuman agent (a computer) exactly what to do.

Os autores aplicaram dois testes nesses alunos: um com objetivo de diagnosticar as competências relacionadas à programação, e o outro com objetivo de diagnosticar as competências relacionadas a matemática.

O resultado mostrou que os alunos apresentavam baixa habilidade em programação, e que grandes números de estudantes possuíam deficiências em conhecimentos matemáticos básicos, esperados em alunos que se encontram na graduação.

Também foi encontrado que os estudantes avaliados possuíam dificuldades em compreender o problema descrito, dificuldades em transformar o problema textual em formula matemática, baixos níveis de abstração e raciocínio lógico.

Além da literatura escassa, para se captar diretamente a percepção do que seria a influência da resolução de problemas na aprendizagem de programação de um determinado indivíduo, se faz necessário que esse indivíduo seja submetido a testes em relação as suas habilidades de raciocínio, não sendo recomendada a aplicação de um instrumento de pesquisa (CASTRO, 2008). Sendo assim esse não será um fator aprofundado no presente trabalho.

2.3 REVISÃO SISTEMÁTICA

Como parte do processo de compreensão do estado da arte da questão de pesquisa do presente trabalho, foi realizada uma revisão sistemática da literatura.

A revisão sistemática (RS) é uma forma de pesquisa que se utiliza de fontes secundárias de pesquisa, tais como artigos e outros textos publicados em repositórios e bancos indexados, que abordam determinado tema. Segundo Sampaio e Mancini (2007), a revisão sistemática disponibiliza um resumo das evidências relacionadas a um tema específico, mediante a aplicação de métodos explícitos e sistematizados de busca, apreciação crítica e síntese da informação selecionada.

Usualmente representa uma atividade inicial de uma pesquisa, e é uma ferramenta útil para alcançar a qualidade das informações. O objetivo é que por meio dela a pesquisa se mostre mais abrangente e menos tendenciosa possível.

Iniciada na medicina teve seu uso expandido para outras áreas das ciências, pois se utiliza do paradigma baseado em evidência.

As principais razões para se realizar uma revisão sistemática, de acordo com Biolchini et al. (2005) são:

- Sumarizar evidências existentes sobre um fenômeno;
- Identificar lacunas na pesquisa atual;
- Fornecer um arcabouço para posicionar novas pesquisas;
- Apoiar novas hipóteses.

E o principal objetivo é construir numa síntese de pesquisa existente que “não tendenciosa”; “seja rigorosa”; “aberta”, e “objetiva”. (BIOLCHINI et al., 2005).

2.4 INSTRUMENTOS DE PESQUISA

Para entender como os sujeitos envolvidos no caso estudado pela presente pesquisa percebem os fatores que dificultam a aprendizagem de programação será construído e aplicado um instrumento de pesquisa. Instrumentos de pesquisa investigam traços de comportamento de uma determinada população ou amostra de pessoas, e são capazes de auxiliarem a identificação de características dos sujeitos em estudo (DE OLIVEIRA et al., 2005).

O ponto central por trás da concepção de um instrumento de pesquisa é o chamado princípio da parcimônia, em que um pequeno número de variáveis hipotéticas, denominadas de fatores ou componentes, podem explicar um grande número de variáveis observadas (LAROS, 2012).

Os instrumentos de pesquisa não possuem uma taxonomia que seja um consenso entre os autores que debatem o tema, sendo necessário entretando algum tipo de classificação de maneira a criar um mínimo de organização (PASQUALI, 1999).

No livro Instrumentos Psicológicos: Manual Prático de Elaboração, Pasquali (1999) apresenta uma classificação simplificada dos tipos de instrumentos, usando como base para sua hipótese as diferentes técnicas utilizadas na construção e aferição dos seus parâmetros de validade e fidedignidade.

Na classificação proposta pelo autor encontram-se os testes referentes a construtos, que são desenvolvidos com base na teoria psicológica e não em dados empíricos (PASQUALI, 1999). Um construto é um traço ou característica que se supõe existir, sendo abstraídos de uma diversidade de condutas que possuem significância educacional (VIANNA, 2003).

Nesse contexto o levantamendo de dados por amostragem, ou *survey*, assegura melhor representatividade e permite generalização para uma população mais ampla. Seu objetivo não está em testar a habilidade do respondente, mas sim em medir sua opinião, seus interesses, aspecto de personalidade e informação biográfica.

2.4.1 Construção de um Instrumento de Pesquisa: Survey

Utilizando como base o esquematizado na figura 01 verifica-se que os objetivos de uma pesquisa levam necessariamente à relação conceito/item e a relação população alvo/amostra.

Figura 1 - Estágios principais de um survey Fonte: Pasquali (1999)

A construção de um teste que utiliza medidas psicométricas obedece, de acordo com Kline (1995), á seguinte sequência:

1. Fundamentação teórica do teste: momento em que se realiza uma revisão bibliográfica cuidadosa sobre o objeto que se deseja avaliar e também estudos exploratórios, especialmente quando a literatura técnica não possui uma produção suficiente para fundamentar a formulação de itens;
2. Formulação de itens do teste: aqui se elabora um número de itens superior ao que se espera obter nas escalas que comporão o teste;

3. Análise preliminar de dificuldade de itens: emprego da técnica de juízes;
4. Análise da fidedignidade: busca-se verificar a consistência interna do teste;
5. Validação do conjunto final de itens do teste: na tentativa de demonstrar que o fator não é apenas uma abstração aleatória, mas um construto que permite uma compreensão do objeto em análise;
6. Padronização: momento em que se descreve o processo de aplicação, avaliação e interpretação do teste.

2.4.1.1 NÚMERO DE ITENS E RESPONDENTES

No que diz respeito à quantidade de itens e respondentes, Pasquali (1999) indica que pra cada fator a ser medido uma média de 100 pessoas devem responder ao questionário, ele também indica que uma média de 15 itens é suficiente para se abordar um tema que possuem um ou dois fatores envolvidos.

Por outro lado, no entanto, Laros (2012) considera que não existe uma regra clara ou consenso para a definição do número de itens ou quantidade de respondentes, pois esses números vão sofrer influência da natureza da pesquisa, e da qualidade do instrumento, mas que o ideal quando se está formulando e testando uma nova hipótese é que o número seja no mínimo próximo de 200 sujeitos.

De acordo com Pasquali (1999) a principal razão que leva a uma determinada pessoa que responda a instrumento de pesquisa de maneira honesta é uma ‘troca social’. Sendo assim o autor chega a conclusão de que para se maximizar a quantidade de respondentes em relação a amostra três princípios precisam ser levados em conta (PASQUALI, 1999):

- Recompensar o respondente: considerada a mais efetiva, a recompensa pode ser um brinde, voucher ou até mesmo acadêmica;
- Reduzir o custo de responder: reduzindo o esforço físico e mental, eliminando custo financeiro, fazendo com que a tarefa pareça breve;
- Estabelecer confiança: oferecer apreciação antecipada, identificar a importância da pesquisa e as credenciais dos envolvidos.

2.4.1.2 DEFINIÇÃO DE UMA MEDIDA DE ESCALA

Para se estabelecer uma relação entre uma hipótese (item) e a opinião do respondente, é necessária a definição de uma escala de medição. Isso ocorre de maneira que a variação na escala indica uma variação na ocorrência do evento (ideias, sentimentos, planos, crenças, etc), geralmente de maneira linear (PASQUALI, 1999).

Dentre as escalas utilizadas nas ciências sociais, a Escala Likert é considerada a mais popular, especialmente no levantamento de opiniões (PASQUALI, 1999). Dentre suas características está à possibilidade de através da mesma medida permitir ao respondente emitir uma opinião concordante, discordante, ou de indiferença. Essa flexibilidade é fundamental, pois através da opção de resposta com valor intermediário, o respondente tem a possibilidade emitir um parecer sobre uma hipótese na qual desconhece ou não possui opinião formada (PASQUALI, 1999).

Como a escala *likert* constrói uma padronização das respostas em um conjunto conciso de variáveis, cria-se a oportunidade de se efetuar uma análise estatística, tanto descritivas quanto inferenciais, dos itens após a aplicação do instrumento (PASQUALI, 1999). Outra vantagem dessa padronização é que ela reduz o custo de resposta, tornando-a mais alinhada aos princípios de maximização de respondentes (PASQUALI, 1999).

As escalas likert podem possuir diferentes níveis de graduação, sendo possível encontrar na literatura escalas com três, cinco, sete, ou até onze níveis de graduação diferentes. Dentre elas a de cinco níveis além de amplamente utilizada possui como vantagem não cansar o respondente em questionários que possuem mais de uma dezena de itens, mantendo uma quantidade suficiente de graduações que de fato representem níveis diferentes e interpretáveis de concordância/discordância, sem prejudicar a confiabilidade (CUMMINGS e GULLONE, 2000). Cummings e Gullone (2000) propõem uma escala cujos valores e enunciados correspondentes estão de acordo com o quadro 01.

Quadro 1 - Valores adotados na escala likert

1	2	3	4	5
Concordo totalmente	Concordo parcialmente	Indiferente	Discordo parcialmente	Discordo completamente

2.4.2 Validação do Instrumento de Pesquisa

É através da validação do instrumento de pesquisa que se torna possível determinar a precisão de inferência específica. Validar não é apenas comprovar o valor do instrumento, mas é também um processo de investigação (RAYMUNDO, 2009).

Pasquali (1999) e Raymundo (2009) propõem que a validação de um instrumento pesquisa deve ocorrer em duas etapas: a primeira é a validação teórica, e segunda é a validação analítica.

2.4.2.1 VALIDAÇÃO TEÓRICA

Criar um instrumento conduzido através de itens é desenvolver uma hipótese de que o que está sendo questionado no instrumento representa de maneira adequada um cenário real (PASQUALI, 1999).

A validação teórica de um instrumento consiste em submetê-lo ao julgamento e opiniões de diferentes examinadores, promovendo uma reflexão sobre o conteúdo das hipóteses (itens). Não é expressa por meio de coeficientes, ou seja, possui um caráter mais subjetivo (RAYMUNDO, 2009). Essa fase da validação ocorre antes da aplicação do instrumento junto ao público-alvo, e é dividida em duas etapas:

- Análise de Conteúdo ou Análise dos Juízes: seu objetivo é avaliar a adequação dos itens em relação aos fatores que estão sendo estudados, ou seja, tentar garantir que existam correlações entre os itens e que o teste possui significado (RAYMUNDO, 2009);

- Análise Semântica dos Itens: seu objetivo é tentar garantir que todos os itens são de fácil compreensão para o público-alvo da pesquisa, eliminando os possíveis erros de português e as ambiguidades nas sentenças (PASQUALI, 1999);

2.4.2.2 VALIDAÇÃO ANALÍTICA

A validação analítica de um instrumento ocorre após a coleta dos dados empíricos (aplicação) junto ao público-alvo. Seus objetivos principais são verificar as relações e a fidedignidade de um instrumento (PASQUALI, 1999). Duas maneiras distintas e complementares de se abordar um conjunto de dados, de modo a verificar suas relações, são a análise fatorial e a análise de *clusters*. A lógica fundamental de ambos os procedimentos é uma classificação através da homogeneidade (KREBS; BERGER; FERLIGOJ, 2000).

A homogeneidade no que diz respeito à análise fatorial tem foco nos valores atribuídos as variáveis pelos respondentes, e seu resultado é uma escala de medida para os fatores comuns intrínsecos que de alguma maneira controlam as variáveis originais (MAROCO, 2003). Seu objetivo é "[...] estabelecer uma base teórica causal de relação entre os indicadores (itens) e uma variável latente (fator ou dimensão)" (KREBS; BERGER; FERLIGOJ, 2000, p. 148)[6].

No caso da análise de clusters, homogeneidade significa que os sujeitos, sejam eles indivíduos ou grupos de indivíduos, são classificados em grupos no que diz respeito à sua semelhança em relação às variáveis; também sendo possível a aplicação do mesmo princípio as próprias variáveis, relacionando-as entre si (KREBS; BERGER; FERLIGOJ, 2000). Seu objetivo é "encontrar uma classificação empírica ou uma estrura de agrupamentos teoricamente definina a priori" (KREBS; BERGER; FERLIGOJ, 2000, p. 148, tradução nossa)[7].

2.4.2.2.1 A ANÁLISE FATORIAL

A análise fatorial mede a dimensionalidade de um instrumento, o que significa definir quantos fatores de fato estão sendo medidos pelo instrumento proposto (LAROS, 2012; PASQUALI, 1999). Ao se construir um instrumento a intenção do investigador é medir um ou mais fatores, porém o instrumento não passa de uma hipótese, não sendo possível afirmar apenas com a análise teórica se os itens de maneira individual, ou como um todo, estão de fato medindo os fatores desejados.

[6] […] establishing a theoretically based causal relationship between indicators (items) and a latent variable (the factor or dimension).
[7] […] the goal of cluster analysis is to find an empirical classification or an a priori theoretically defined cluster structure.

Faz-se necessário, então, conduzir uma análise fatorial, cujo resultado irá definir a sua dimensionalidade (PASQUALI, 1999).

A análise fatorial produz resultados importantes com os quais se podem tomar decisões sobre a qualidade dos itens, bem como do instrumento no seu todo. Ela também identifica os conjuntos de variáveis altamente correlacionados entre si e que, portanto, formam um fator comum latente (KREBS; BERGER; FERLIGOJ, 2000), como o ilustrado na figura 02.

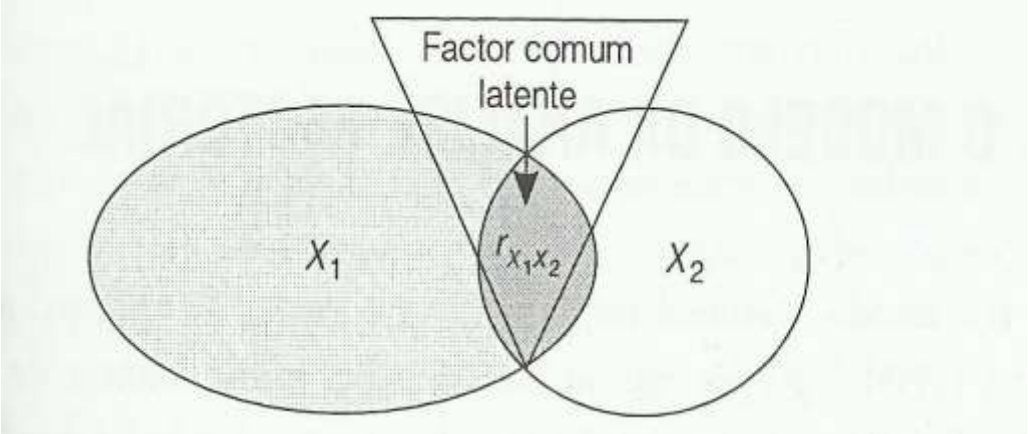

Figura 2 – Representação do fator comum latente Fonte: Maroco (2003)

Para cada item é produzida uma carga fatorial (saturação) deste no fator e esta carga fatorial indica a covariância entre o fator e o item. Isto quer dizer, a carga fatorial mostra quanto porcento existe de parentesco (covariância) entre o item e o fator, de sorte que quanto mais próximo de 100% de covariância item-fator, melhor será o item, pois ele assim se constitui num excelente representante comportamental do fator latente. O objetivo primordial da análise de fatorial é o de atribuir um *score* (quantificação) a fatores que não são diretamente observáveis (MAROCO, 2003).

Outro uso para os dados gerados pelo processo de análise fatorial é a respeito da redução ou não de dados. Seu objetivo é “[...] descobrir ponderações ótimas para as variáveis mensuradas [...]” (LAROS, 2012, p. 167). Isso ocorre através da exclusão ou substituição de itens que possuam uma baixa carga fatorial, permitindo assim um aumento na qualidade do instrumento atual ou insumo para sua reformulação.

Para a execução da Análise Fatorial faz-se necessário testar se a matriz de dados é suscetível à fatoração, o que significa analisar se os dados estão de fato aptos a serem sujeitados ao processo de análise (DAMASIO, 2012). Dois testes devem ser executados com essa finalidade: a “medida da adequação da amostragem de Kaiser-Meyer-Olkin (KMO)” e o “teste de esfericidade de Bartlett” (MAROCO, 2003).

O índice KMO possui características heurísticas e é uma medida da homogeneidade das variáveis, que compara as correlações simples com as correlações parciais observadas entre as variáveis. Como correlações parciais entende-se como a correlação existente entre duas variáveis depois de eliminadas as influências de outras variáveis que também apresentam correlação com as primeiras (MAROCO, 2003). Seu valor é calculado pelo quadrado das correlações totais dividido pelo quadrado das correlações parciais, podendo variar de 0 a 1 (DAMASIO, 2012). Sua classificação em relação aos valores é definida de acordo com o quadro 02.

Quadro 2 - Classificação pelo KMO

Valor do KMO	Classificação
0,9 – 1,0	Excelente
0,8 – 0,9	Boa
0,7 – 0,8	Média
0,6 – 0,7	Medíocre
0,5 – 0,6	Aceitável
< 0,5	Inaceitável

O “teste da esfericidade de Bartlett” tem como objetivo verificar a hipótese de que a matriz correlação é uma matriz-identidade, o que indicaria que não existe uma

correlação entre os dados, e que, portanto, o modelo fatorial não está apropriado (DAMASIO, 2012). Valores menores que 0,05 (5%) indicam que a matriz é fatorável, rejeitando a hipótese de que a matriz de dados é similar a matriz-identidade.

2.4.2.2.2 A ANÁLISE DE CLUSTERS

Ao se tentar identificar agrupamentos naturais uma medida de semelhança ou dissemelhança explícita, e com o mínimo de subjetividade, é necessária. A análise de grupos ou de *clusters* permite o agrupamento de sujeitos ou variáveis em grupos de maneira homogênea. Os agrupamentos de indivíduos tem como base seguir uma distância métrica, e os grupos de variáveis são fundamentados com medidas de correlação (MAROCO, 2003).

Tendo como objeto da análise os indivíduos, uma das métricas que podem ser utilizadas na análise de *clusters* é a Distância Euclidiana, que pode ser calculada com a raiz quadrada do somatório dos quadrados das diferenças entre as coordenados dos pontos em cada dimensão (IBRAHIM et al., 2014). Isso significa medir o comprimento da reta que une dois indivíduos em um espaço de n-dimensões. Quanto maior a distância euclidiana menor é a semelhança entre dois indivíduos, e quanto menor a distância euclidiana maior a semelhança entre eles (DANCEY, REIDY, 2006).

Quando as variáveis são o objeto de análise as medidas apropriadas são coeficientes de correlação amostrais. Para variáveis ordinais o coeficiente de Spearman deve ser o utilizado (MAROCO, 2003).

Dentre as técnicas de análise de *clusters* Ibrahim et al. (2014) consideram a técnica de Agrupamento Hierárquico como uma boa ferramenta para análise de dados em diferentes métodos. De acordo com o capítulo sobre análise de *clusters* presente no livro de Maroco (2003) as “[...] técnicas hierárquicas recorrem a passos sucessivos de agregação dos sujeitos considerados individualmente, i.e. cada sujeito é um Cluster, e depois estes vão sendo agrupados de acordo com suas proximidades [...]” (MAROCO, 2003, p. 302).

Como os *clusters* são formados por dois ou mais sujeitos, ou duas ou mais variáveis, faz-se necessário um modo que defina a distância entre os diferentes *clusters* (KREBS; BERGER; FERLIGOJ, 2000). Dentre os sete métodos possíveis, o

Método do Centróide cria um ponto de coordenadas compostas pelas médias dos sujeitos que fazem parte do *cluster*, e depois calcula a distância entre os *clusters* através do ponto de coordenadas criado. O Método da Menor Distância utiliza a menor distância de cada um dos elementos que constituem um *cluster* em relação aos demais (MAROCO, 2003).

2.4.2.2.3 ALFA DE CRONBACH

Medir a fidedignade de um instrumento é verificar sua precisão. Isso significa que “a fidedignidade de um instrumento diz respeito ao montante de variância verdadeira que ele produz vis-à-vis a variância erro, isto é, quanto maior a variância verdadeira e menor a variância erro, mais fidedigno o instrumento [...]” (PASQUALI, 1999, p. 67).

Uma das maneiras de se medir a fidedignidade é através do coeficiente denominado “Alfa de Cronbach” (α), uma técnica estatística que visa verificar a homogeneidade, necessitando que o teste seja aplicado em apenas uma única ocasião para que possa ser executado (PASQUALI, 1999). Foi apresentado pela primeira vez em 1951 por Lee J. Cronbach, e pode ser aplicado em itens que utilizam a mesma escala de medição. Seu cálculo é a partir da covariância entre os itens e da variância entre os itens individuais (FREITAS e RODRIGUES, 2005).

A literatura científica ainda não possui um consenso a respeito da relação entre os valores do alfa de cronbach e a confiabilidade de questionários, mas a classificação mais aceita é a apresentada no quadro 03.

Quadro 3 - Classificação pelo alfa de cronbach

Valor de α	Confiabilidade
α < 0,30	Muito baixa
0,30 ≤ α < 0,60	Baixa
0,60 ≤ α < 0,75	Moderada
0,75 ≤ α < 0,90	Alta
α ≥ 0,90	Muito alta

Diferentemente da análise teórica, o objeto de estudo da análise analítica não são as hipóteses (itens), mas sim as variáveis e as informações que elas podem oferecer (MAROCO, 2003).

3. METODOLOGIA

3.1 CARACTERIZAÇÃO DA PESQUISA

A presente pesquisa se caracteriza como um estudo de caso. De acordo com Nisbett e Watt (1978, *apud* André, 2005), que se basearam na definição estabelecida na Conferência de Cambridge, um estudo de caso é um termo amplo que designa uma família de métodos de pesquisa com um enfoque em uma instância específica, esta instância pode ser uma pessoa, um grupo, uma escola, uma instituição, um programa, etc.

As diferentes técnicas de coleta de dados utilizadas no estudo de caso o caracterizam como uma pesquisa de abordagem mista, já que se utiliza tanto de técnicas qualitativas quanto quantitativas.

Nisbett e Watt (1978, *apud* André, 2005) ainda indicam que o desenvolvimento de um estudo de caso segue, em geral, três fases:

1) A exploratória ou de definição do foco do estudo: momento em que se define a instância de estudo, e se fazem os primeiros contatos e caracterização do caso;

2) A coleta de dados: momento em que o pesquisador procede à coleta sistemática dos dados, escolhendo a abordagem metodológica (quantitativa, qualitativa, ou mista), utilizando-se de instrumentos e fontes variadas em momentos e situções diversificadas;

3) A análise sistemática dos dados: em um estudo de caso a análise está presente em diversos momentos, mas se torna sistemática após a coleta de dados. Assim, é o momento em que o pesquisador se debruça sobre os dados coletados, os lê e relê para daí extrair os sentidos e respostas às suas questões.

3.2 CARACTERIZAÇÃO DO CASO ESTUDADO

Em função da disponibilidade de acesso aos dados relevantes para pesquisa, a Faculdade do Gama (FGA) foi escolhida como instância para o presente estudo de caso. A instituição é um *campus* da Universidade de Brasília (UnB), inaugurado no ano de 2008.

De acordo com as informações constantes no site da instituição[8], especificamente no dia 28 de setembro de 2007, foi realizada reunião que abordou a pactuação dos cursos que seriam ofertados no Campus do Gama da UnB, durante a Fase I do Programa de Expansão das Instituições de Ensino Superior – REUNI, do Governo Federal.

Atualmente a Faculdade UnB Gama – FGA, oferta os seguintes cursos e vagas: 1) Engenharia Aeroespacial (60 vagas); 2) Engenharia Automotiva (60 vagas); 3) Engenharia Eletrônica (60 vagas); 4) Engenharia de Energia (60 vagas), e 5) Engenharia (60 vagas), totalizando 560 vagas anuais.

Uma curiosidade sobre a instituição é que atualmente o estudante ingressa no curso de Engenharias e posteriormente ele opta por uma das cinco engenharias citadas anteriormente. Assim, todos os alunos da FGA iniciam sua vida acadêmica como alunos do curso intitulado Engenharia Básica, e a partir do final do terceiro semestre devem fazer a opção por uma das especializações.

Os cursos de Engenharias tem uma duração média de 12 períodos, sendo que cada período corresponde a um semestre letivo na instituição. Sendo assim a matriz disciplinar é dividida em duas fases: a das disciplinas denominadas de tronco comum, que são obrigatórias a todos os alunos independentemente da especialização pretendida, e que de maneira geral são disciplinas presentes no fluxo dos três primeiros períodos; e após fazer a opção por um dos campos da engenharia, o aluno cursa as disciplinas próprias de cada especialização.

Atualmente, o primeiro contato do aluno do curso de Engenharia de Software com a aprendizagem de programação se dá na disciplina **Computação Básica** (CB), que possui carga horária de 6 horas de aula por semana e duração de um semestre letivo. Essa disciplina é ofertada no segundo semestre e é obrigatória a

[8] https://fga.unb.br/unb-gama/sobre. Consultado em 01/06/2014.

todos os alunos, independente da especialização em engenharia a ser escolhida no futuro.

Historicamente, a disciplina era ofertada como **Introdução a Ciência da Computação** (ICC), possuindo carga horária de 4 horas de aula por semana com duração de um semestre letivo, pertencia ao fluxo de matérias do primeiro período da graduação em Engenharia. No segundo semestre de 2013 (2013/2) ocorreu a mudança para a atual disciplina de Computação Básica (CB), que possui carga horária de 6 horas de aula por semana, e duração de um semestre letivo.

Todas as turmas são mistas, ou seja, são compostas por estudantes que pretendem cursar qualquer uma das especializações em engenharia, sem distinções relacionadas ao sexo, contato prévio com programação, ou provável opção de curso.

Na Universidade de Brasília (UnB) o valor das notas finais varia entre 0,0 e 10,0. Para obter aprovação o aluno precisa de nota final superior ou igual a 5,0 e possuir presença em ao menos 75% das aulas. As notas são classificadas em menções de acordo com esse resultado. São divididas então em oito categorias de acordo com o apresentado no quadro 04.

Quadro 4 - Tipos de menções aplicadas na Universidade de Brasília[9]

TIPO DA MENÇÃO	RESULTADO
Superior (SS)	Nota final entre 9,0 e 10,0
Média Superior (MS)	Nota final entre 7,0 e 8,9
Média (MM)	Nota final entre 5,0 e 6,9
Média Inferior (MI)	Nota final entre 3,0 e 4,9
Inferior (II)	Nota final entre 0,1 a 2,9
Sem Rendimento (SR)	Nota 0 ou possuir acima de 25% de faltas durante o período letivo
Trancamento (TR)	Trancamento da disciplina solicitado durante o período de trancamento especificado no calendário acadêmico
Trancamento Justificado (TJ)	Trancamento da disciplina solicitado em qualquer momento do semestre através de uma justificativa (motivo de saúde, etc)

Esse cenário cria uma interessante oportunidade de pesquisa, pois em decorrência da abordagem adotada pela instituição os estudantes que compõem as turmas de Computação Básica não são apenas alunos de Engenharia de Software, assim como nos casos reportados por Eckerdal et al., Guzdial (2010), e Mendes et

[9] Fonte: http://www.unb.br/unb/transparencia/downloads/regimento_estatuto_unb.pdf

al. (2012), onde as turmas eram compostas por graduandos em computação e de outros cursos.

Na verdade a grande maioria dos alunos frequentando as aulas de computação básica na FGA, exceto aqueles que por alguma razão não estejam na posição certa em relação ao fluxo de matérias, ainda não tiveram que fazer a opção por uma das especializações de engenharia.

Essa possibilidade implica que, mesmo que um aluno possua intenção inicial de optar por uma especialização, sua escolha pode mudar, já que antes de se fazer a opção em definitivo percorre-se três semestres de matérias. Por essa razão a disciplina Computação Básica pode estar exercendo um papel importante, pois as experiências ocorridas ao longo da disciplina impactarão na decisão de seguir ou não o caminho ligado a computação (BENNEDSEN e CASPERSEN, 2007; FORTE e GUZDIAL, 2005).

3.3 FASES DA PESQUISA

A metodologia utilizada neste trabalho constou de três fases: Na Fase 01, foi realizada uma revisão sistemática (RS). Na Fase 02, foi realizada uma pesquisa exploratória junto à universidade, para caracterizar de maneira mais apropriada a instância a ser estudada, constando de pesquisa documental e um levantamento junto aos alunos da disciplina de CB. Na Fase 03, foi desenvolvido um instrumento de pesquisa que foi aplicado a uma amostra de estudantes da instituição, sendo os seus resultados analisados. Estas fases são detalhadas nas seções a seguir.

3.3.1 FASE 01

Nessa fase foi conduzida a revisão sistemática de literatura, para se identificar o estado da arte e delimitar o escopo do tema, bem como, construir o referencial teórico da pesquisa.

Em consonância com a questão de pesquisa, foi definido como palavra-chave o termo *programming learning.* Iniciou-se, então, a execução da revisão sistemática, guiada pelo protocolo presente no **Apêndice (I)** deste documento.

No auxílio da execução desta revisão foi-se utilizada a ferramenta gratuita *State of The Art through Systematic Review* (StArt)[10], desenvolvida e mantida pelo Laboratório de Pesquisa em Engenharia de Software (LAPES) do Departamento de Computação da Universidade Federal de São Carlos (DC/UFSCAR). Esta ferramenta possui como função organizar e dar suporte a técnica da revisão sistemática, fornecendo uma maneira de organizar as diferentes fases da revisão sistemática, e gerando gráficos interativos que promovem uma melhor visualização dos dados coletados. Os resultados da revisão sistemática podem ser encontrados no **Apêndice (II)**.

A quantidade de artigos coletados em cada base de dados e que foram submetidos ao processo de seleção é apresentada no quadro 05.

Quadro 5 - Quantidade de artigos coletados em cada base de dados

Base de Dados	Quantidade de Artigos Coletados
ACM	9
IEEE	28

3.3.2 FASE 02

Para conhecer a realidade da Universidade de Brasília, campus do Gama, uma pesquisa exploratória foi realizada. A partir de documentos fornecidos pela secretaria da instituição, cujos dados eram relacionados ao controle acadêmico dos alunos que cursaram as disciplinas de Introdução à Ciência da Computação e Computação básica, foi feita uma análise para tentar descobrir algumas de suas características próprias que possam estar relacionadas com o processo de ensino-aprendizagem.

Também nessa fase foi realizado um levantamento, com a aplicação de um instrumento exploratório, baseado em um questionário contendo questões gerais,

[10] http://lapes.dc.ufscar.br/tools/start_tool

para a obtenção de informações demográficas e de perfil dos alunos da disciplina de computação básica.

3.3.3 FASE 03

Nessa fase um instrumento de pesquisa foi desenvolvido com objetivo de ser aplicado na coleta de dados da presente pesquisa junto aos indivíduos matriculados no curso de Computação Básica. Para tanto, foi construído um questionário de pesquisa seguindo as orientações propostas por Pasquali (1999), tendo como base teórica a literatura proveniente da revisão sistemática e nos levantamentos da pesquisa exploratória sobre a instituição.

As etapas envolvidas nesse processo são as apresentadas no diagrama apresentado na figura 03.

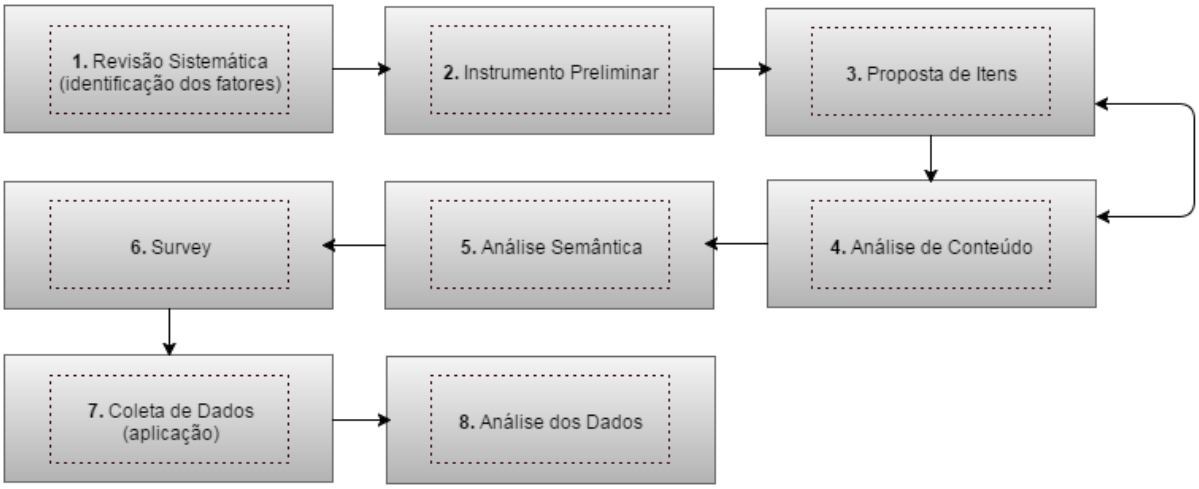

Figura 3 - Construção do survey

1. **Revisão Sistemática** – Após a leitura e análise de todos os artigos dois fatores que influenciam no processo de ensino aprendizagem, e seus aspectos foram identificados. O primeiro desses fatores é a motivação, e o segundo, a própria atividade de programar.

2. **Instrumento Preliminar** – Para a construção de um bom instrumento de pesquisa apenas a fundamentação teórica pode não ser suficiente, então recomenda-se sempre que possível a aplicação de um instrumento investigatório a uma amostra menor da população. Esse instrumento não possui a necessidade de ser limitado por escala ou formato, devendo permitir uma análise qualitativa

dos dados, já que seu principal objetivo é apontar caminhos e complementar os conhecimentos do pesquisador (LAROS, 2012).

Com esse objetivo, o instrumento apresentado no **Apêndice (III)**, e denominado como instrumento preliminar, foi proposto e aplicado em um grupo reduzido de alunos matriculados na disciplina Computação Básica. A finalidade foi de se identificar conceitos iniciais e criar familiaridade com o tópico, buscando determinar uma ênfase de quais conceitos devem ser medidos e como devem ser medidos de em relação a alguma das particularidades da instituição estudada. Os resultados obtidos e sua análise se encontram no **Apêndice (IV)**.

3. **Proposta de Itens –** Com os resultados da revisão sistemática e da análise exploratória em mãos, um conjunto de itens foi proposto. Esses itens possuem uma forma padronizada através de uma mesma escala de medida, e que visam tentar medir a percepção dos alunos matriculados na disciplina Computação Básica na FGA em relação aos fatores envolvidos na aprendizagem inicial de programação.

   Neste ponto o objetivo é de que os itens propostos conduzam uma análise explanatória ou explicativa. A finalidade dessa análise é tentar estabelecer as relações causais e de existência, além de questionar o porquê de essas relações serem como são. O conjunto de itens proposto nessa etapa é superior ao número de itens que vão compor o survey de fato, pois eles ainda serão confrontados com outras opiniões, o que pode levar a retirada ou modificação de itens que compõem a proposta.

4. **Análise de Conteúdo –** Conduzida por profissional com experiência e expertise na área para julgar a relevância dos itens propostos e sua construção. Foi analisado também o equilíbrio na distribuição dos itens e sua relação com o processo. A análise ocorreu em mais de uma iteração, até que todas as modificações necessárias foram feitas e aprovadas. O resultado da análise de conteúdo estabeleceu essa relação entre ambos de acordo com o apresentado no quadro 06.

Quadro 6 - Relação fatores/itens

Fator	Itens
Motivação	1, 3, 5, 6, 13, 14
Atividade de Programação	4, 7, 9, 10, 11, 15
Motivação e Atividade de Programação	2, 8, 12, 14

5. **Análise Semântica –** Conduzida por uma professora de português, foi responsável por julgar a qualidade semântica de cada item, como por exemplo, se as informações estavam claras e não apresentavam ambiguidades, além de uma revisão ortográfica.

6. **Survey –** Após o fim do processo de validação teórica chegou-se a proposta apresentada no **Apêndice (V)**. O instrumento é composto por um total de 16 questões relacionadas aos fatores da aprendizagem inicial de programação e 9 questões relacionadas a identificação da amostra.

7. **Coleta de Dados –** Com os dados levantados na instituição referentes à média de alunos matriculados na disciplina por período, que indicam a necessidade de uma alta aderência de respondentes em relação ao número de matriculados, para se atingir os critérios propostos por Laros (2012) e Pasquali (1999) a respeito do número mínimo de respondentes necessários, além da experiência adquirida com a aplicação do instrumento preliminar, o processo de coleta de dados do *survey* ocorreu de três maneiras distintas, visando alcançar o maior número de respondentes, de acordo com o apresentado no quadro 07.

   O *survey* existe em duas versões, concebidas para atenderem os diferentes procedimentos de coleta, e os princípios da maximização de respondentes. Uma das versões do survey é um formulário eletrônico criado através da plataforma *Google Forms*[11], e a outra é o *survey* em versão impressa em documento.

[11] https://www.google.com/forms/

Quadro 7 – Procedimentos de coleta

Procedimento de Coleta	Processo
Envio do questionário através do endereço de e-mail.	As turmas da disciplina foram visitadas no horário da aula pelos responsáveis pela pesquisa. Foi então feito um convite a todos para participarem, e uma folha foi passada entre os alunos para que os interessados anotassem seus endereços de e-mail para envio do *link* para o preenchimento do questionário.
Convocação dos alunos através das redes sociais.	Foram criados posts contendo o link para preenchimento do questionário nas comunidades da instituição e da disciplina nas redes sociais convocando os alunos matriculados na disciplina a participarem da pesquisa.
Instrumento impresso.	Versões impressas do instrumento foram entregues para preenchimento por alunos nas dependências da instituição. Antes de o instrumento ser entregue o aluno era questionado em relação a estar ou não matriculado na disciplina, e a já ter ou não preenchido o questionário previamente através de outros meios.

Outra maneira de se maximizar o número de respondentes foi através da aplicação dos princípios propostos por Pasquali (1999) de acordo com o apresentado no quadro 07.

Quadro 8 - Processo de aplicação dos princípios da maximização de respondentes

Princípio para maximização de respondentes	Maneira que o princípio foi aplicado
Recompensar o respondente	Esse princípio não foi aplicado, pois devido as limitações da pesquisa não era possível oferecer nenhum tipo de recompensa para os respondentes.
Reduzir o custo de responder	Survey disponível via formulário eletrônico, análise semântica dos itens, respostas utilizam uma escala padronizada de medida.
Estabelecer confiança	Em todos os procedimentos de coleta argumentou-se sobre a importância da pesquisa, reforçando a importância da participação dos alunos, e as credenciais dos envolvidos no trabalho.

Um total de 233 respondentes foram contabilizados ao se encerrarem as aplicações.

8. **Análise dos Resultados –** Os resultados de todas as aplicações do survey foram transportados para uma mesma planilha de dados para serem submetidos

as análises estatísticas. Nessa etapa foram conduzidas a análise fatorial, a análise de *clusters*, o cálculo do alfa de Cronbach. Além disso, os resultados das questões foram analisados com o uso de técnicas estatísticas descritivas: média e frequência.

4. ANÁLISE DOS RESULTADOS

4.1 RESULTADOS DO LEVANTAMENTO DE DADOS NA INSTITUIÇÃO

Os documentos fornecidos pela secretária da instituição eram referentes ao controle acadêmico das turmas de Introdução a Ciência da Computação e Computação Básica durante o período compreendido entre o primeiro semestre letivo de 2009 (2009/1) e o segundo semestre letivo de 2013 (2013/2).

Os documentos possuíam os dados do número de alunos que se matricularam em cada turma, e o número de alunos que receberam cada menção. As informações referentes a cada turma foram tratadas, e os principais resultados, de maneira sumarizada, são comentados abaixo.

Na tabela 01 são apresentados os dados referentes ao número de turmas e de alunos por turma no período compreendido pela pesquisa:

Tabela 1 - Distribuição de turmas e alunos por período

PERÍODO	NÚMERO DE TURMAS	TOTAL DE ALUNOS
2009/1	4	313
2009/2	4	298
2010/1	5	333
2010/2	8	395
2011/1	6	313
2011/2	5	347
2012/1	7	364
2012/2	7	391
2013/1	7	386
2013/2	7	146
Total 10	60	3286

Esses dados apontam que em média 328,6 alunos se matriculam na disciplina de introdução a programação por semestre. Saber desse número é importante, porque permite alcançar, através do planejamento do procedimento de coleta de dados, e da aplicação correta dos princípios de maximização de respondentes, que

o número médio de 200 de respondentes mínimos necessários para a correta aplicação de um survey que visa medir a percepção de dois fatores (LAROS, 2012; PASQUALI, 1999), seja atingido em uma aplicação que ocorra em um único período letivo, viabilizando que o estudo seja realizado na instituição.

No figura 04 é apresentado o número de alunos total a atingir determinada menção por período.

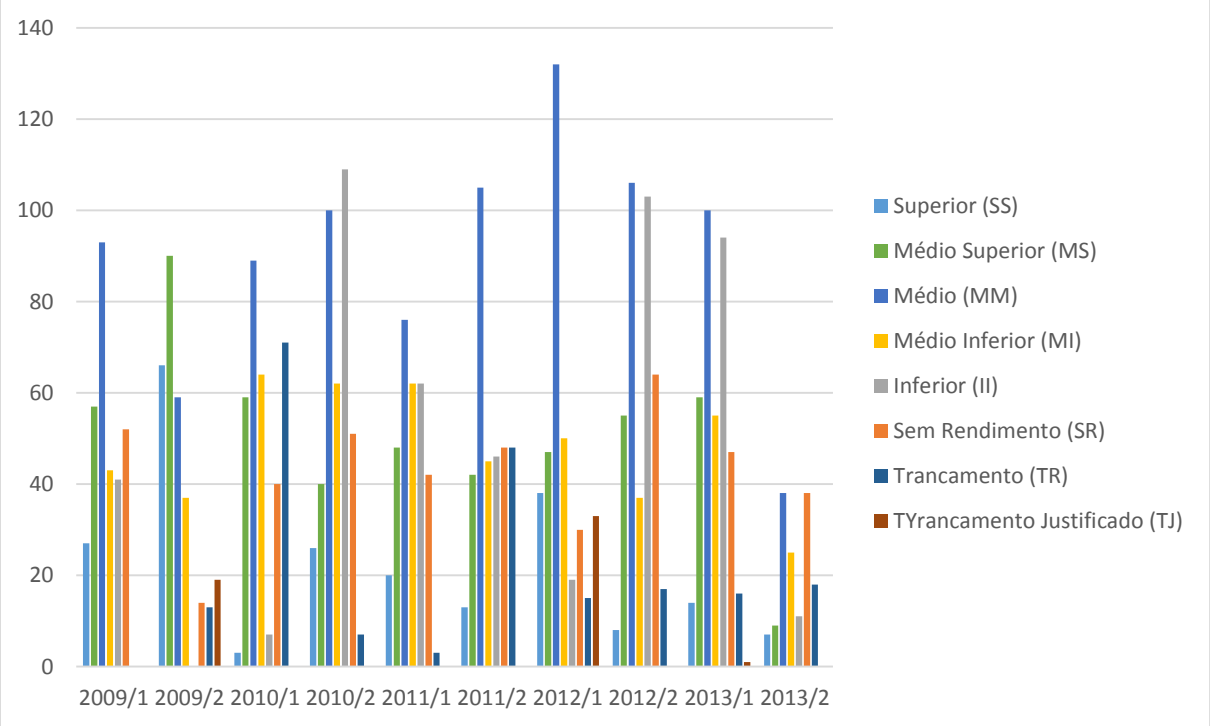

Figura 4 - Distribuição das menções por período

No figura 05 é apresentado o número de aprovações, reprovações, e trancamentos totais por período.

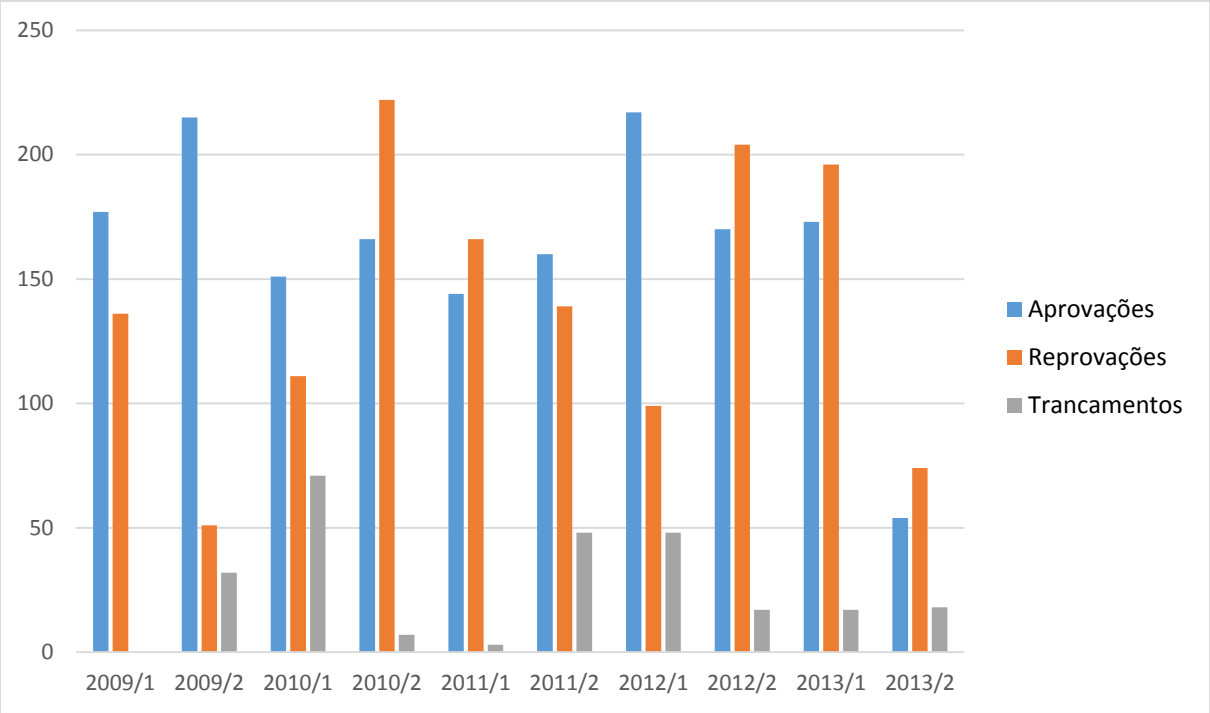

Figura 5 - Número de aprovações/reprovações/trancamentos por período

Na figura 06 é apresentada a porcentagem total do rendimento acadêmico no período que compreende a pesquisa.

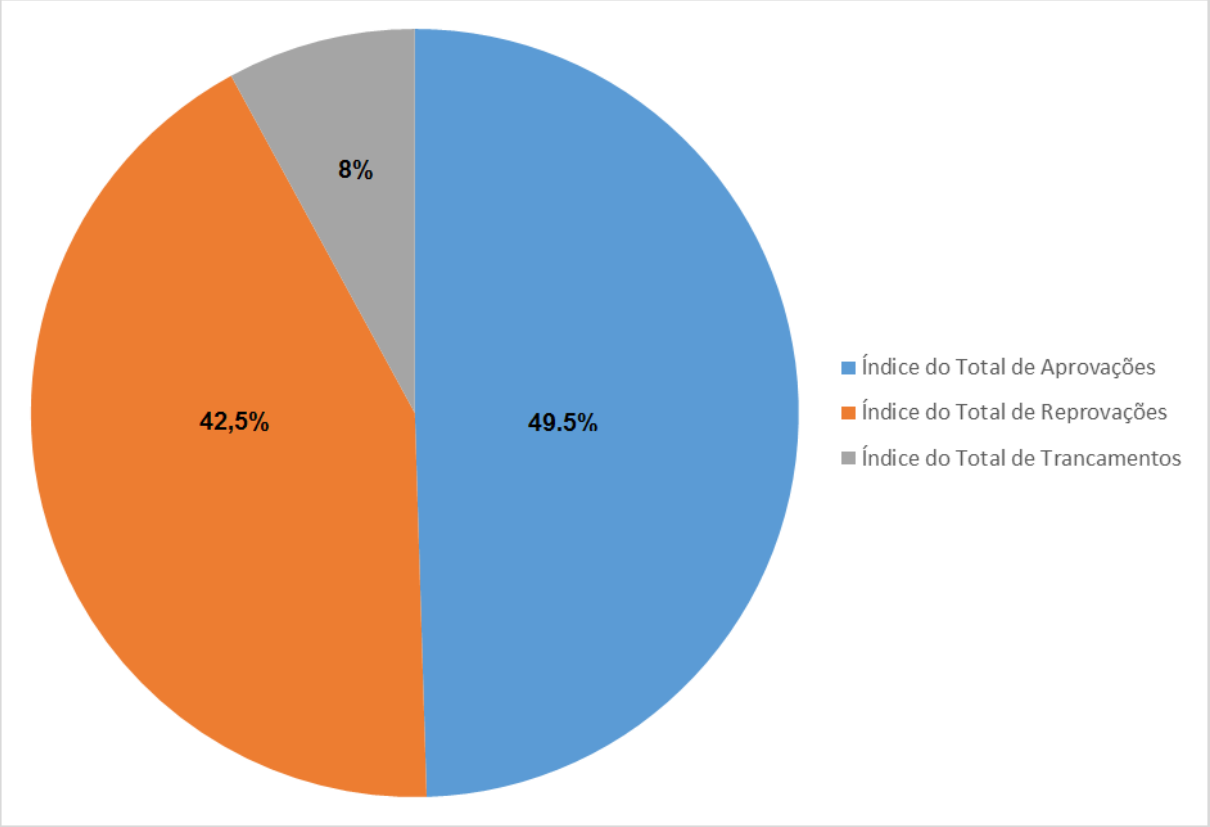

Figura 6 - Índice total de aprovações/reprovações/trancamentos entre 2009/1 e 2013/2

Esses resultados, quando comparados aos resultados apresentados por Bennedsen e Caspersen (2007) em seu estudo sobre as taxas de falha em cursos de programação, se mostram bem preocupantes. A taxa apresentada pelos autores, que incluiu 63 universidades, é de que em média 67% dos estudantes obtém sucesso no curso introdutório de programação, mas no contexto da FGA a taxa média é de apenas 49,5%; ou seja, a taxa de aprovação na FGA é mais de 30% menor do que a apresentada pelos autores.

Ramos Silveira e Reis Wakim (2010) escreveram um artigo no qual fazem um levantamento dos custos envolvidos no ensino de graduação superior na Universidade de Viçosa. Apesar desse levantamento não possuir uma aderência total a FGA, pois a mesma apresenta suas próprias peculiaridades, as variáveis para o cálculo do custo de cada disciplina por aluno são os mesmos nos dois casos, com diferença apenas no valor de alguns dados. Uma das conclusões chegadas pelos autores é que as disciplinas oferecidas pelo Departamento de Informática da Universidade de Viçosa possuem uma média de custo de 567,36 R$ por cada aluno matriculado.

Ao longo dos 10 semestres acompanhados pelo levantamento de dados feito junto à secretária da FGA, um total de 1.659 alunos matriculados falharam na aprovação na disciplina introdutória de programação da FGA. Pegando como referência o valor médio por matricula proposto por Ramos Silveira e Wakim (2010) para disciplinas de informática, teríamos um total de 941.250,24R$ gastos. Esse valor é muito representativo, sendo um valor de custo significativo para a universidade. Quando se pensa que esse cenário se repete em dezenas de universidades pelo país, esse valor representa grande impacto nos gastos do governo como um todo.

As características próprias da FGA obtidas através da pesquisa documental suportam a existência de especificidades e problemas da instituição relacionados ao seu processo de ensino-aprendizagem.

4.2 RESULTADOS SUMARIZADOS DO INSTRUMENTO PRELIMINAR

O instrumento preliminar foi aplicado a uma amostra de 83 sujeitos. Sua aplicação foi realizada através do instrumento impresso, que foi distribuído aos estudantes. É composto de 16 questões e pode ser encontrado no **Apêndice (III)**.

As respostas de todas as questões foram analisadas de maneira qualitativa, tendo como base o discutido na fundamentação teórica. Esses resultados são apresentados e discutidos no **Apêndice (IV)**. As principais informações extraídas foram utilizadas como parte dos insumos na concepção dos itens do survey e são apresentadas no quadro 09.

Quadro 9 - Resultados sumarizados do instrumento preliminar

Informação Extraída	Questão Relacionada
Aprender a programar aparenta influenciar na opção da especialização de engenharia.	Q2, Q3
O número de sujeitos que tiveram algum contato prévio com programação é significativo.	Q5
Os sujeitos aparentam possuir traços de desmotivação, baixa expectativa de aprovação, e falta de contextualização.	Q7, Q8, Q10, Q15, Q16
Computação em seu significado mais abrangente aparenta ser um tema de interesse dos sujeitos.	Q9
Os sujeitos aparentam não ter muita noção de como estudar o tema.	Q12, Q13

4.3 RESULTADOS OBTIDOS NO SURVEY

Todos os resultados relatados abaixo foram calculados através do pacote estatístico de ferramentas IBM SPSS[12]. Todos os dados coletados através da aplicação do *survey* foram transportados para uma planilha, que foi então importada para análise no programa.

[12] http://ibm.com/software/br/analytics/spss/

4.3.1 IDENTIFICAÇÃO DA AMOSTRA

Os 233 respondentes foram classificados e identificados, conforme mostram as seções a seguir:

4.3.1.1GÊNERO

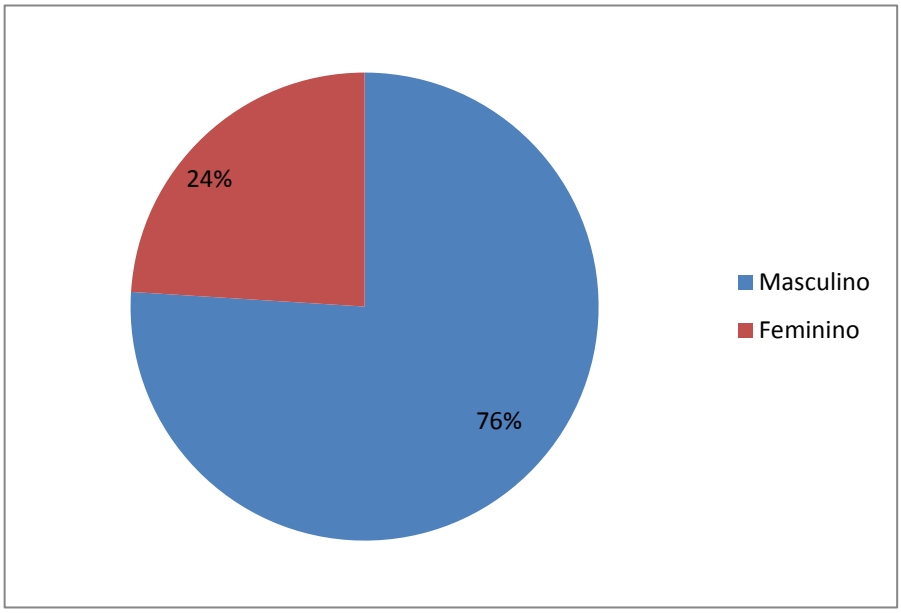

Figura 7 - Distribuição dos respondentes de acordo com o gênero

Participaram da pesquisa 177 (76%) representantes do sexo masculino e 56 (24%) representantes do sexo feminino.

4.3.1.2 IDADE

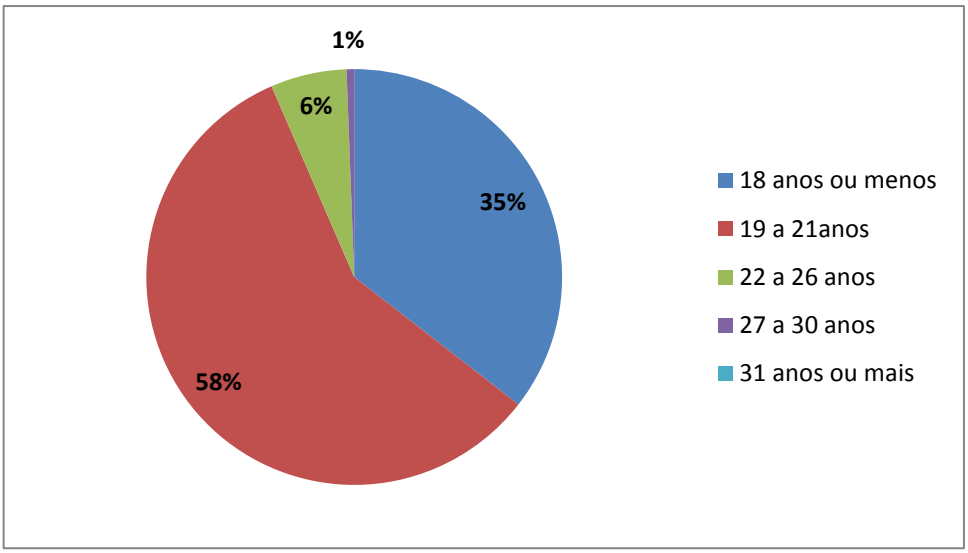

Figura 8 - Distribuição dos respondentes de acordo com a idade

A distribuição de idade entre os respondentes se deu da seguinte maneira: 82 (35%) dos respondentes possuiam 18 anos ou menos, 135 (58%) possuiam entre 19 e 21 anos, 14 (6%) possuiam entre 22 a 26 anos, 2 (1%) possuiam entre 27 a 30 anos; e nenhum (0%) possuiam 31 anos ou mais.

Esses dados mostram características de heterogeneidade das turmas, que são compostas por indivíduos de diferentes faixas etárias, sendo isso consequência de diferentes fatores como indivíduos que tiveram ingresso tardio na graduação, que estão cursando sua segunda graduação, que estão em sua primeira graduação, etc.

4.3.1.3 NÚMERO DE SEMESTRES NA INSTITUIÇÃO

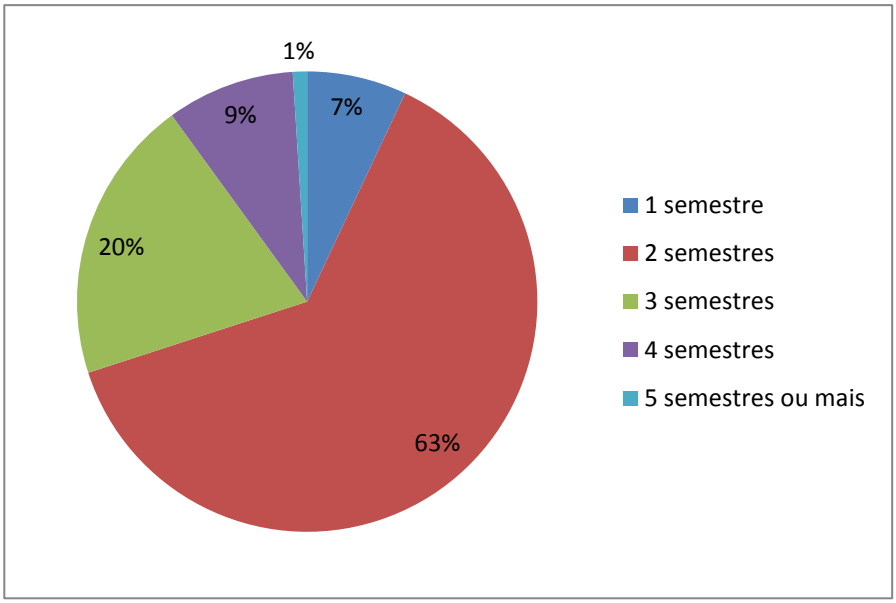

Figura 9 - Distribuição dos respondentes de acordo com o número de semestres matriculados na instituição

Dentre os respondentes 17 (7%) estão ainda em seu primeiro semestre na instituição, 146 (63%) se encontram no segundo semestre, 46 (20%) no terceiro, 21 (5%) no quarto, e 3 (2%) estão há 5 semestres ou mais.

Este dado é um outro indicador da heterogeneidade da amostra, mostrando que uma boa parte dos alunos não se encontra no fluxo de matérias recomendado, sendo isso consequência de diversos fatores como indivíduos que vieram transferidos de outros cursos, que já estão cursando a disciplina pela segunda ou terceira tentativa, etc.

4.3.1.4 DISTRIBUIÇÃO DOS CURSOS

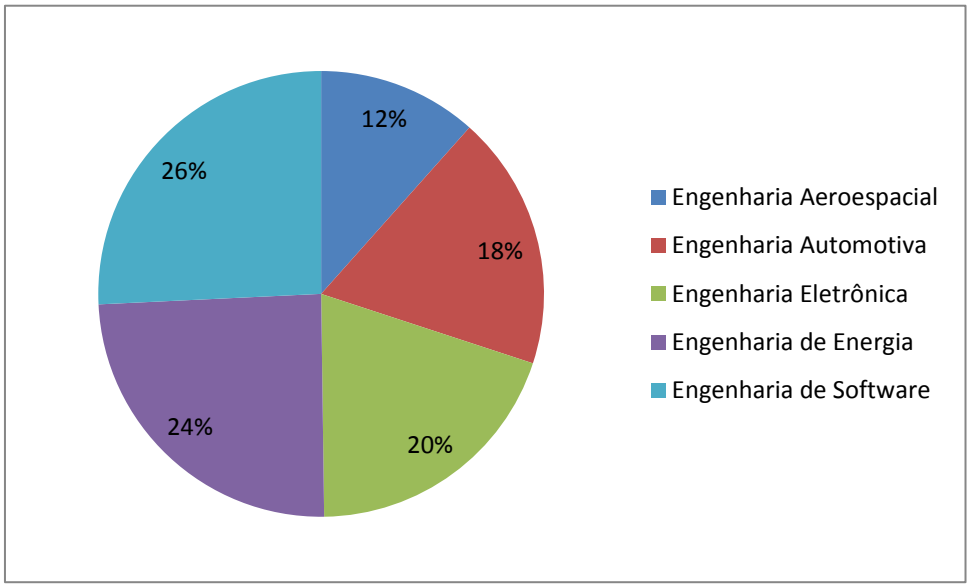

Figura 10 - Distribuição dos respondentes de acordo com o curso

Em relação às especializações das engenharias que os alunos pretendem cursar, ou já estão cursando, a distribuição ficou da seguinte forma: 27 (12%) respondentes de aeroespacial, 42 (18%) de automotiva, 46 (20%) de eletrônica, 57 (24%) de energia, e 60 (26%) de software.

Vale notar que apesar da obrigatoriedade de todos os alunos que vão cursar qualquer engenharia ofertada pela instituição terem de passar pela disciplina de introdução a programação, apenas 26% deles, que serão os prováveis a cursarem engenharia de software, irão obrigatoriamente continuar tendo contato com programação ao longo do curso e na vida profissional. Tanto a didática da disciplina, quanto a ementa, objetivos, tópicos estudados, etc, tem por padrão seguir diretrizes relacionadas aos cursos de computação. Ou seja, é uma abordagem pragmática heterogênea sendo aplicada para um público alvo totalmente heterogêneo quanto às aplicações futuras da matéria que estão estudando.

4.3.1.5 FORMA DE INGRESSO

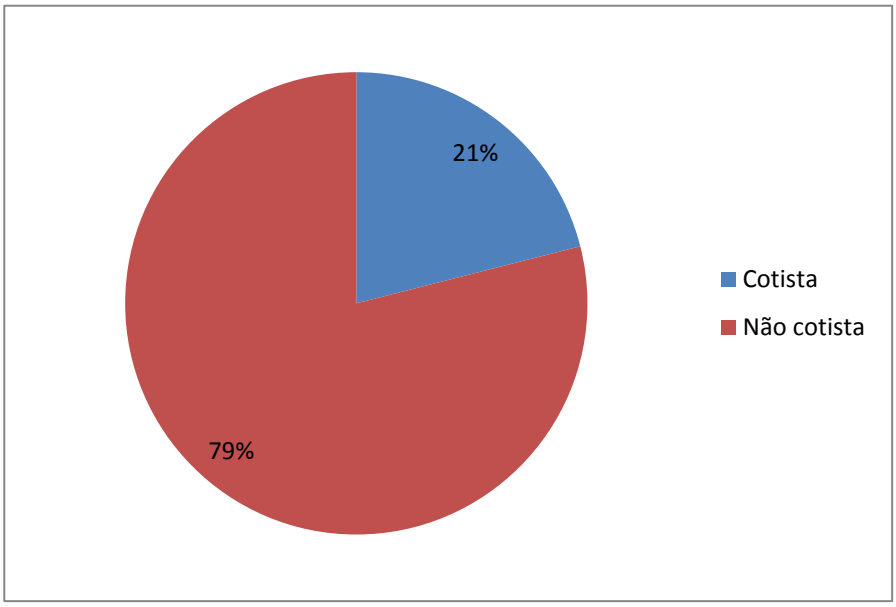

Figura 11 - Distribuição dos respondentes de acordo com a forma de ingresso na instituição

Dentre os respondentes 49 (21%) ingressaram na instituição através do sistema de cotas e 184 (79%) não utilizaram o sistema.

4.3.1.6 FLUÊNCIA EM LINGUA ESTRANGEIRA

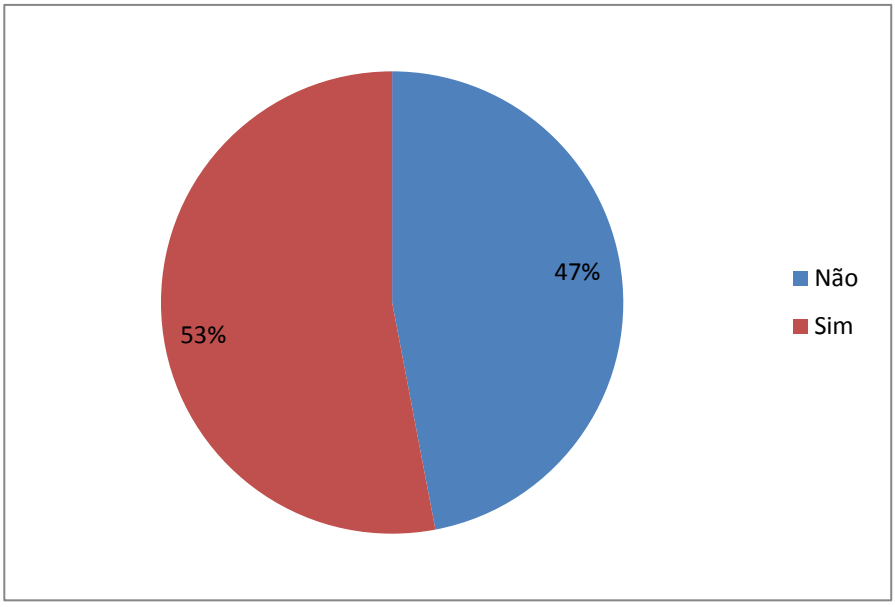

Figura 12 - Distribuição dos respondentes de acordo com a fluência em idioma estrangeiro

Dos respondentes 123 (53%) consideram dominar uma lingua estrangeira, enquanto 110 (47%) não dominam nenhuma. E dentre os que dizem dominar algum idioma estrangeiro foram identificados 3 idiomas: inglês, espanhol e italiano. Dentre

os que dominam uma lingua estrangeira a distribuição dos idiomas pode ser vista no gráfico a seguir:

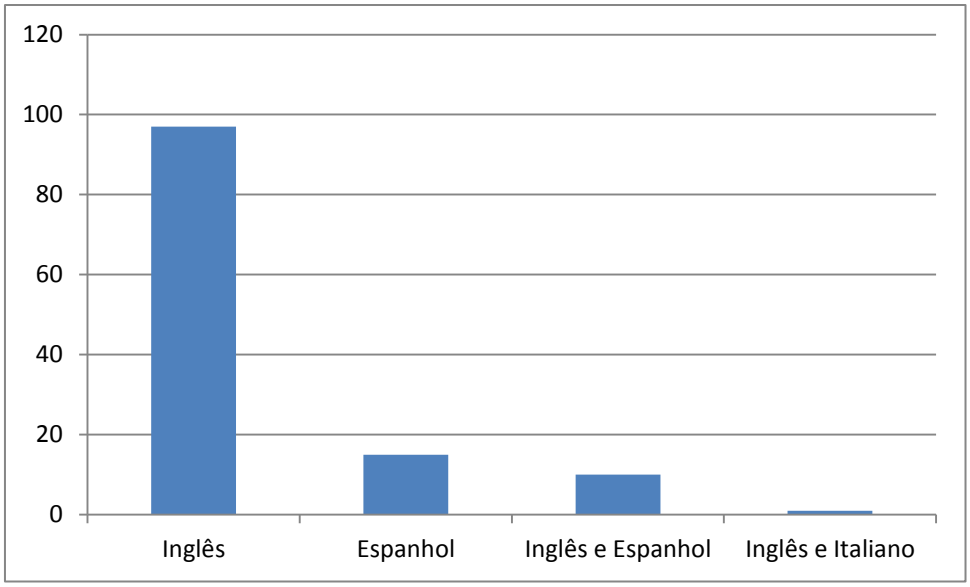

Figura 13 - Quantidade de indivíduos de acordo com o idioma de fluência

Esse dado é relevante pois a lingua original em que os raciocínios algoritmicos são construídos é o inglês, e isso pode fazer toda a diferença na construção da lógica por trás do aprendizado do indivíduo (CHUN; RYOO, 2010).

4.3.1.7 PRIMEIRO CONTATO COM PROGRAMAÇÃO

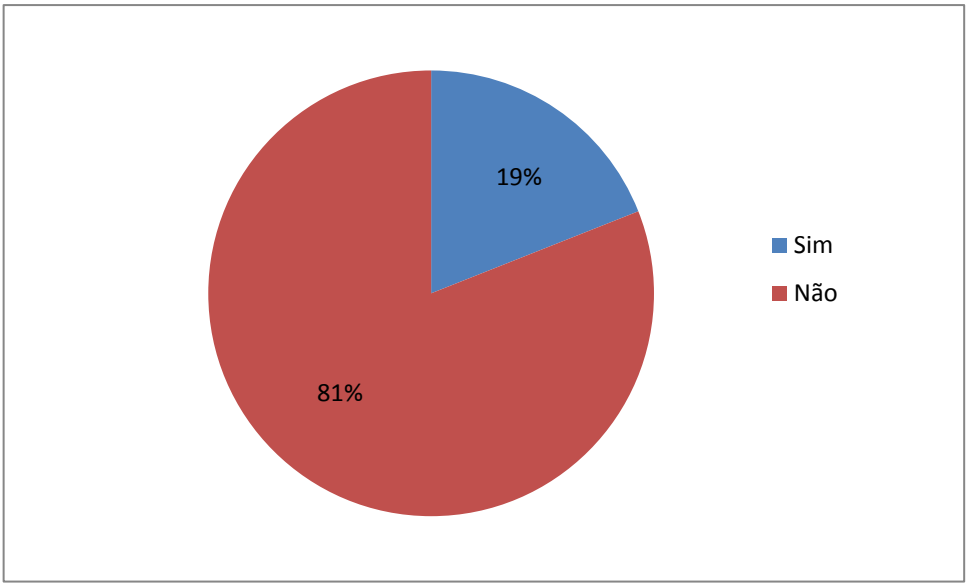

Figura 14 - Distribuição dos respondentess de acordo com o primeiro contato com programação

Quando perguntados se seu primeiro contato com programação foi na instituição cursando computação básica, 189 (81%) dos respondentes afirmaram que sim e 44 (19%) respondentes afirmaram que não.

Esses dados vão ao encontro das informações levantada nos intrumento preliminar, de que uma parcela significativa dos alunos matriculados na disciplina não estão tendo contato com o tema pela primeira vez.

4.3.1.8 REGIÃO EM QUE HABITA

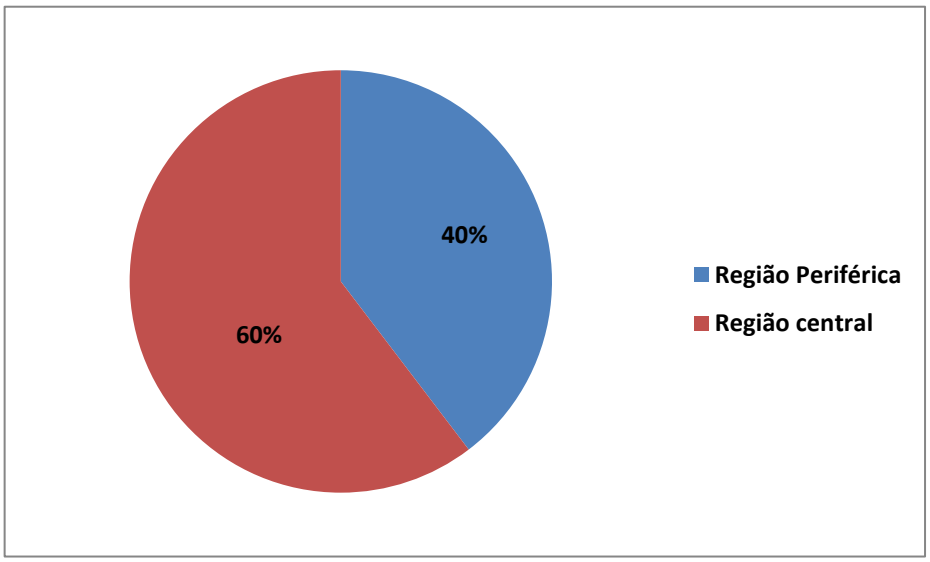

Figura 15 - Distribuição dos respondentes de acordo com a região em que habitam

Em relação a região em que habita os respondentes foram classificados em 2 categorias: região central, da qual 140 (60%) dos respondentes dizem habitar; e região periférica, da qual 93 (40%) dos respondentes dizem habitar.

4.3.2 VALIDAÇÃO DO INSTRUMENTO

A primeira análise realizada foi a da consistência dos dados registrados. A planilha de respostas foi inspecionada com o intuito de se identificarem respostas inválidas ou sequências de respostas idênticas, o que invalidariam essas respostas. Todas as respostas foram aprovadas nesse filtro.

Após este procedimento foram calculados os testes de KMO e de Bartlett, de maneira a verificar se a matriz é passível da análise de fatores. Os resultados destes procedimentos são os apresentados na tabela 02.

Tabela 2 - Resultado dos testes de KMO e Bartllet

Medida Kaiser-Meyer-Olkin de adequação de amostragem.		0,539
Teste de esfericidade de Bartlett	**Aprox. Qui-quadrado**	325,762
	df	105
	Sig.	,000

De acordo com os valores de referência apresentados no quadro 02 o resultado do teste KMO é aceitável, e o resultado do teste de esfericidade de Bartlett deu negativo, indicando que a matriz de correlações não é uma matriz-identidade, confirmando que existe alguma correlação entre os dados e que a análise fatorial é possível.

Ao final da execução do processo de análise fatorial uma série de dados foi gerada como saída: a matriz de correlações, a matriz de componentes, a variância total explicada e o *scree plot*. Cada um deles apresentando um aspecto da análise.

Na tabela 03 é apresentada a matriz de correlações. Essa matriz é utilizada para se investigar a dependência linear entre múltiplas variáveis ao mesmo tempo. É representada na forma de uma tabela contendo os coeficientes de correlação entre cada variável e as demais. Os coeficientes de correlação são uma medida de grau linear, cujo valor pode variar de -1,00 a 1,00. Quando o coeficiente entre duas variáveis possui carga positiva, à medida que uma variável aumenta a outra também aumenta. Quando o coeficiente entre as variáveis possui carga negativa, à medida que a variável aumenta a outra diminui.

De acordo com Laros (2012), o coeficiente de correlação apresenta resultados significativos quando seu valor é de no mínimo 0,2. Essa regra é aplicável tanto para cargas positivas quanto negativas, sendo que os valores abaixo dessa

marca representam correlações muito fracas entre as variáveis, podendo ser ignorados.

No caso dos resultados apresentados na tabela 02, apenas os itens grifados em negrito possuem um índice superior ou igual ao mínimo, o que significa que entre eles existe uma alguma correlação significativa, e indica que os dois exercem influência um no outro (TAYLOR, 1990). O fato de todos os outros coeficientes não possuírem valores significantes, indica uma baixa comunalidade, o que pode apontar para uma dimensionalidade do instrumento superior à hipótese inicial de dois fatores (PASQUALI, 1999).

Para uma análise mais profunda a respeitos dos fatores, quantos devem ser retidos, qual sua influência nos resultados, etc, os dados a serem analisados são a variância total explicada, cujo resultado se encontra na tabela 04; e a matriz de componentes, cujo resultado é apresentado na tabela 05.

A variância total explicada visa sumarizar a informação da matriz de correlação, e apresenta-la por outro ângulo. Seu objetivo é o de descrever a porcentagem da variância de cada uma das variáveis originais nas novas componentes.

Tabela 3 - Matriz de correlações

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15
Q1	1,00	**-,22**	,15	,08	-,15	,07	-,11	,08	,07	-,02	,04	-,08	-,11	-,13	-,08
Q2	**-,22**	1,00	**-,23**	-,10	,14	-,02	,04	,04	**-,24**	-,07	,04	,15	,02	,12	,10
Q3	,15	**-,23**	1,00	**,24**	,13	,06	,12	-,03	,15	,06	-,19	-,02	,04	-,07	-,04
Q4	,08	-,10	**,24**	1,00	,19	,08	**,20**	-,05	,06	,17	-,14	,03	-,18	,03	,09
Q5	-,14	,13	,13	,18	1,00	**,31**	,03	,01	-,16	-,08	,01	,15	-,17	,16	,13
Q6	,07	-,02	,06	,08	**,31**	1,00	-,07	,05	-,10	-,06	,12	,04	-,12	**,28**	-,07
Q7	-,11	,04	,12	**,20**	,03	-,07	1,00	,03	,11	-,06	-,18	-,03	,16	,03	,11
Q8	,08	,03	-,03	-,05	,01	,05	,03	1,00	-,16	-,08	,10	,14	,02	-,02	,14
Q9	,07	**-,23**	,15	,06	-,16	-,10	,11	-,16	1,00	**,25**	-,19	-,08	-,03	-,14	-,08
Q10	-,02	-,07	,05	,17	-,08	-,06	-,06	-,08	**,25**	1,00	,03	-,14	,09	,03	**-,22**
Q11	,03	,03	-,19	-,14	,01	,11	-,18	,10	-,19	0,3	1,00	-,03	,12	**,20**	,05
Q12	-,08	,14	-,02	,03	,15	,04	-,03	,14	-,08	-,14	-,03	1,00	,03	,06	,04
Q13	-,10	,01	,04	-,18	-,17	-,12	,16	,02	-,03	,09	,12	,03	1,00	,11	-,02
Q14	-,13	,12	-.07	,03	,16	**,28**	,03	-,02	-,14	,03	**,20**	,06	,11	1,00	-,11
Q15	-,07	,09	-,04	,09	,13	-,07	,11	,14	-,08	**-,22**	,05	0,4	-,02	-,11	1,00

Tabela 4 – Variância total explicada

Fator	Autovalores Iniciais		
	Total	% de Variância	% Cumulativa
1	2,017	13,448	13,448
2	1,706	11,371	24,819
3	1,471	9,810	34, 629
4	1,390	9,267	43,896
5	1,190	7,933	51,829

6	1,013	6,753	58,583
7	0,967	6,448	65,031
8	0,839	5,591	70,622
9	0,798	5,320	75,942
10	0,745	4,965	80,906
11	0,711	4,738	85,644
12	0,615	4,103	89,748
13	0,605	4,036	93,784
14	0,504	3,363	97,147
15	0,428	2,853	100,000

O número de componentes inicial é equivalente ao número de variáveis usadas na análise fatorial, pois apenas somando todos é possível obter os 100% da variância. Porém a ideia da análise fatorial é que se selecione um número otimizado de fatores, o que seria o número ideal de fatores a serem retidos, e é equivalente a menor quantidade de fatores com que é possível obter a explicação da maior parte da variância (MAROCO, 2003).

O autovalor inicial representa a variâncias dos fatores, e é um dos indicares de quais fatores devem ser retidos. No caso da análise fatorial recomenda-se que seja retido todo o fator cujo autovalor seja igual ou superior a 1,0.

A análise da tabela 04 aponta que para explicar mais de 50% da variação são necessários ao menos cinco fatores, uma variação que apesar de relevante ainda não é precisa o suficiente. “No caso de experiências de laboratório pode conseguir-se reter 95% ou mais [...] mas quando se trabalha com humanos, num grande número de situações, duas ou três componentes não conseguem explicar muito mais do que 50% [...]” (MAROCO, 2003, p.248). Quanto mais fatores são retidos, menos útil cada um se torna.

Apesar dos resultados da variância total explicada para os dados obtidos apontar que seis fatores devem ser retidos, esse resultado não nos interessa, já que o objetivo inicial é o de se construir um instrumento que meça dois fatores. Sendo assim decidiu-se por se reter dois fatores no momento de se gerar a matriz de componentes. Essa decisão vai permitir que seja feita uma análise das questões mais significtativas em relação aos dois fatores almejados, o que vai permitir inferências a respeitos de quais questões devem ser mantidas e quais devem ser rejeitadas na construção de um novo instrumento mais eficiente na medição de apenas dois fatores.

Os valores das cargas fatoriais são expressos de maneira similar aos coeficientes de correlação, com valores variando de -1,00 a 1,00. Um item com carga de 0,00 não possui nenhuma relação com o fator, e o valor de 0,30 (positivo ou negativo) é apontado como a carga mínima necessária para o item ser um representando útil do fator (PASQUALI, 1999).

O resultado das cargas fatoriais dos itens, para os dois fatores que decidiu-se por reter, são apresentados na tabela 05.

Tabela 5 - Matriz de componentes

	Fator 1	Fator 2
Q1	**-,308**	,001
Q2	**,548**	-,115
Q3	**-,375**	**,530**
Q4	-,190	**,668**
Q5	**,416**	**,626**
Q6	**,323**	**,449**
Q7	-,098	,288
Q8	,286	-,022
Q9	**-,642**	,068
Q10	**-,398**	-,026
Q11	**,383**	-,294
Q12	**,353**	,164
Q13	,023	**-,337**
Q14	**,393**	,175
Q15	,282	,160

De acordo com os resultados da matriz de componentes, 12 questões, cujas cargas fatoriais estão grifadas em negrito, possuem representação significativa junto aos dois fatores que são objetos de estudo.

Em seguida foi gerado o *Scree plot* referente aos itens. Esse gráfico permite uma análise subjetiva de que fatores são os principais responsáveis pela variância, ou seja, aquelas que exercem maior interferência direta no resultado. Ele é uma representação gráfica dos autovalores iniciais, e auxilia na decisão a respeito da retenção de fatores. A figura 16 apresenta o *scree plot* da amostra, suportando as conclusões feitas a respeito da variância total explicada.

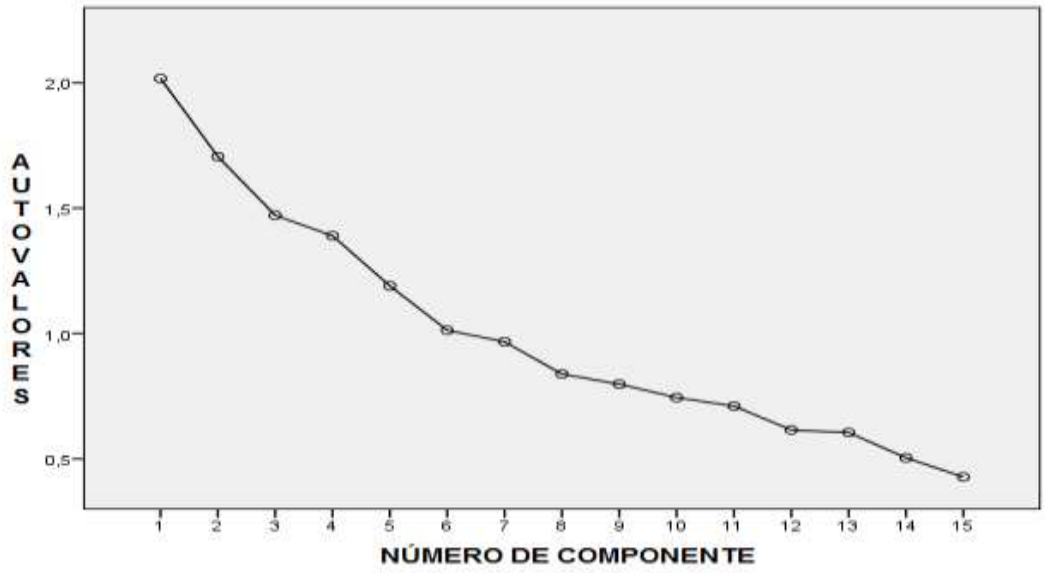

Figura 16 - Scree plot

Tanto a utilização dos autovalores quanto do *scree plot* são critérios subjetivos (MAROCO, 2003), recomendações, pois um componente com autovalor abaixo de 1,0 provalmente não é tão relevante quanto os demais que possuem autovalor superior, sendo que esta regra não é de aplicabilidade geral, e seu propósito principal é incentivar a reflexão sobre os itens presentes no instrumento.

O passo seguinte foi o de submeter os dados à análise de *clusters*. Os resultados dessa análise podem tanto confirmar as conclusões apontadas pela análise fatorial quanto apresentar novos pontos de vista levando a novas conclusões.

A análise de clusters foi executada de duas maneiras distintas. Uma tendo como objeto da análise as variáveis e a outra tendo como objeto da análise os indivíduos que participaram das pesquisas. Essa análise vai ser dirigida para a estrutura tipológica, com o objetivo de descobrir grupos de pessoas ou variáveis que reagem de forma semelhante às hipóteses descritas nos itens. Os grupos são formados por similaridades comparando as pontuações individuais sobre cada item com a média específica do *cluster* (de todos os objetos pertencentes a esse *cluster*).

A primeira análise cujo resultado está apresentado na tabela 06 teve como alvo os indivíduos, e seu resultado obtido através dos pontos centrais de cinco *clusters*, resultando em cinco tipos médios de invíduos.

Tabela 6 - Resultado da análise de clusters sobre os sujeitos

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
Q1	1	1	4	3	1
Q2	5	1	1	2	2
Q3	2	1	5	1	5
Q4	5	1	5	5	5
Q5	5	1	5	1	3
Q6	2	2	1	5	1
Q7	1	1	1	5	4
Q8	1	1	1	4	4
Q9	1	5	5	4	4
Q10	5	5	3	1	5
Q11	1	1	1	3	4
Q12	1	1	1	1	3
Q13	1	1	1	2	5
Q14	5	1	1	1	5
Q15	1	1	5	5	3

Efetuando uma inspeção visual nos dados é possível perceber que existem comunalidades entre os indivíduos médios. É possível se observar padrões que podem ser utilizados na construção de perfis. Uma análise pertinente pode ser efetuada ao se comparar a média e a mediana das respostas das questões dos resultados gerados a partir dos indivíduos médio, com a média e a mediana das respostas das questões geradas a partir das respostas de todos os indivíduos. A mediana nesse caso é a mais recomendada, pois a escala utilizada é do tipo *likert*, que por não ser contínua não é influenciada pelas casas decimais (PASQUALI, 1999; MAROCO, 2003; KREBS; BERGER; FERLIGOJ, 2000). Essa comparação pode ser vista na tabela 07.

Tabela 7 - Média e mediana do resultado geral e da análise de clusters

	Média Geral das Questões	Mediana Geral das Questões	Média da Análise de Clusters	Mediana da Análise de Clusters
Q1	2,416	2,00	2,00	1,00
Q2	1,622	1,00	2,20	2,00
Q3	2,854	3,00	2,80	2,00
Q4	3,660	4,00	4,20	5,00
Q5	1,849	1,00	3,00	3,00
Q6	2,163	2,00	2,20	2,00
Q7	1,364	1,00	2,40	1,00
Q8	1,454	1,00	2,20	1,00
Q9	4,111	5,00	3,80	4,00
Q10	3,510	4,00	3,80	5,00
Q11	2,051	2,00	2,00	1,00
Q12	1,211	1,00	1,40	1,00
Q13	1,961	2,00	2,00	1,00
Q14	1,502	1,00	2,60	1,00
Q15	3,042	3,00	3,00	3,00

Percebe-se que os resultados das medianas convergem para o mesmo espectro da escala, apoiando a hipótese de que existe um padrão em relação à percepção dos indivíduos aos questionamentos.

O outro procedimento de análise de *clusters* teve como alvo as relações das variáveis entre si, e o seu resultado foi o dendograma apresentado na figura 17, e que representa como as variáveis foram agrupadas em cada *cluster*.

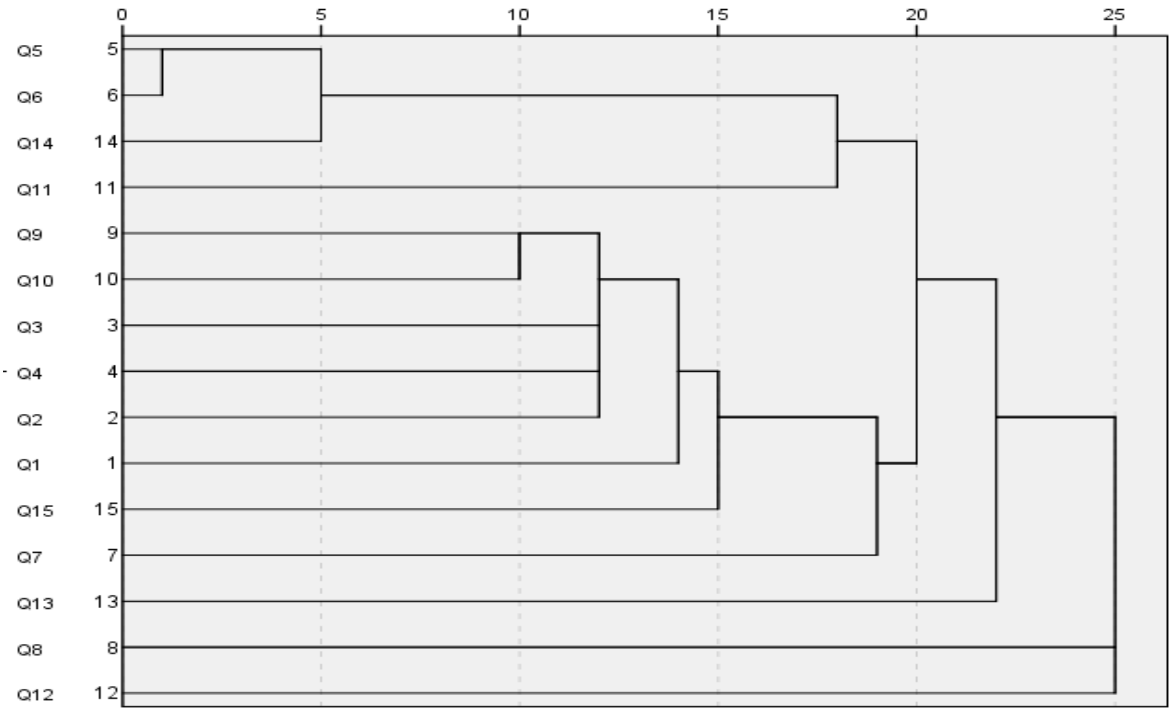

Figura 17 - Agrupamento de variáveis por cluster

O primeiro agrupamento feito envolveu as variáveis das questões 5 e 6, que na análise fatorial já haviam sido apontadas como as que possuíam a correlação mais alta. A grande vantagem do dendograma e da análise de clusters é que é possível identificar as semelhanças que ocorrem em estágios futuros, que são representadas pelo agrupamento de variáveis a outras variáveis já agrupadas. Isso permite inferir se as variáveis poderiam, por exemplo, ser combinadas na construção de uma nova, já que elas apresentam comunalidades quando agrupadas.

Por último foi efetuado o cálculo do alfa de cronbach, que pode ser calculado tanto para as componentes de maneira individual como para o instrumento como um todo. O resultado apresentado no quadro 10 foi calculado tendo como base todos os itens da amostra como um todo.

Quadro 10 - Resultado do alfa de cronbach geral

Alfa de Cronbach (α)	0,372

De acordo com os valores de referência apresentados no quadro 09 o valor de **α** encontrado é considerado de baixa confiabilidade. Esse valor já era de certa forma esperado pelas análises feitas da matriz de correlações e da variância total explicada, e sua interpretação é de que de fato o instrumento mede mais fatores do que o planejado, significando que se o teste for aplicado novamente para uma amostra com as mesmas características os resultados podem apresentar diferenças significativas em relação à aplicação anterior. Isso se deve ao fato de fatores não controlados ou que ainda não são compreendidos exercerem influência significativa nos valores dos resultados. Dentre as possíveis hipóteses pode-se citar a de heterogeneidade da amostra, e de que o objeto de estudo é mais multifacetado do que o inicialmente proposto.

De acordo com Laros (2012), a construção de um instrumento de pesquisa confiável o bastante para poder gerar generalização em sua aplicação, é um processo que, dependendo do objeto de estudo, pode levar anos, além de serem necessárias à participação e expertise de muitos colaboradores. Esse trabalho demanda que sejam executadas sucessivas análises fatoriais, um controle preciso da amostra para permitir a execução de teste e reteste, além de grupos segmentados para testes específicos e de exceção (LAROS, 2012).

4.3.3 ANÁLISE DAS QUESTÕES

Depois de efetuada a validação e a identificação dos respondentes que compõem a amostra, algumas conclusões foram desenvolvidas com base nas relações e resultados das respostas coletadas.

Esses resultados foram analisados com base em duas abordagens distintas: uma foi nomeada como abordagem “geral”, e a outra que foi denominada abordagem “comparatória”. Na abordagem geral utiliza-se na análise o resultado geral das respostas de todos os respondentes. Na abordagem comparatória os resultados foram divididos em dois grupos parem serem analisados de maneira comparativa: um grupo contendo apenas respondentes que possuem a pretensão de

cursar engenharia de software, e que foi nomeada como “grupo de engenharia de software”; e um grupo composto por respondentes que pretendem cursar as demais opções de engenharias, e que foi nomeado “grupo de outras engenharias.

O critério de escolha para qual abordagem deveria ser adotada na análise de cada questão, usou como base a relevância do questionamento para a discussão, diferenças estatísticas significativas entre as respostas, e a fundamentação teórica como um todo. O quadro 11 apresenta qual das abordagens foi adotada em cada questão.

Quadro 11 - Abordagem de análise adotada por questão

Abordagem Adotada	Questão
Geral	1,5,6,7,8,9,10,12
Comparatória	2, 3, 4, 11, 13, 15

Dentre as principais análises que podem ser inferidas a partir dos dados coletados destacam-se as seguintes:

4.3.3.1 QUESTÃO 1

Gostar de temas relacionados à computação (jogos, redes sociais, *tablets*, *smartphones*, etc) faz aumentar a motivação para aprender a programar.

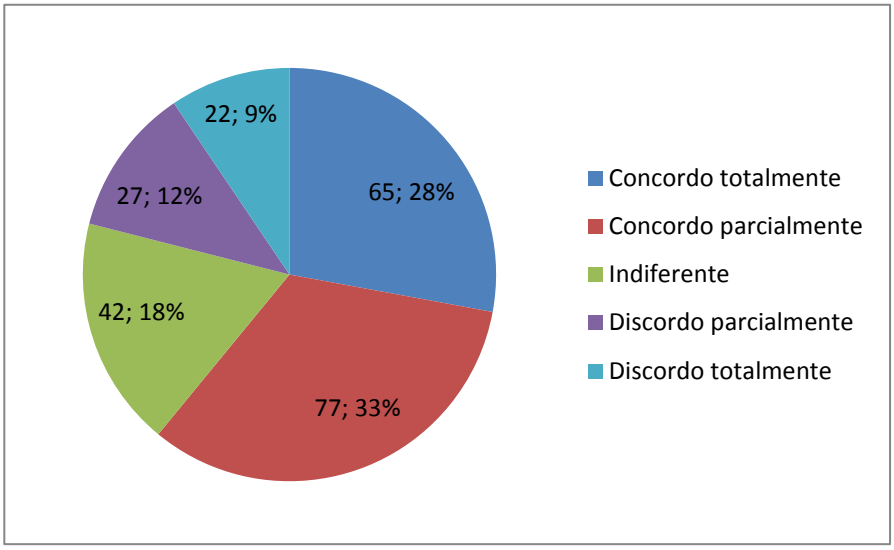

Figura 18 - Distribuição das respostas da questão 1 do survey

A questão 1 quando analisada sob ótica dos resultados apresentados no gráfico 14, e na média e mediana apresentadas na tabela 07, relaciona um ganho de motivação dos indivíduos quando existe gosto por algum tema relacionado a computação.

Essa ideia vai ao encontro do proposto por autores como Guzdial (2010), Mentes et al. (2012), Gianotti (1987), e outros. O aluno precisa se sentir contextualizado, e uma poderosa ferramenta para atingir essa meta no processo de aprendizagem é utilizar aquilo que é familiar fora da sala de aula (GODWIN-JONES, 2005).

4.3.3.2 QUESTÃO 2

Aprender a programar é uma tarefa complexa e difícil.

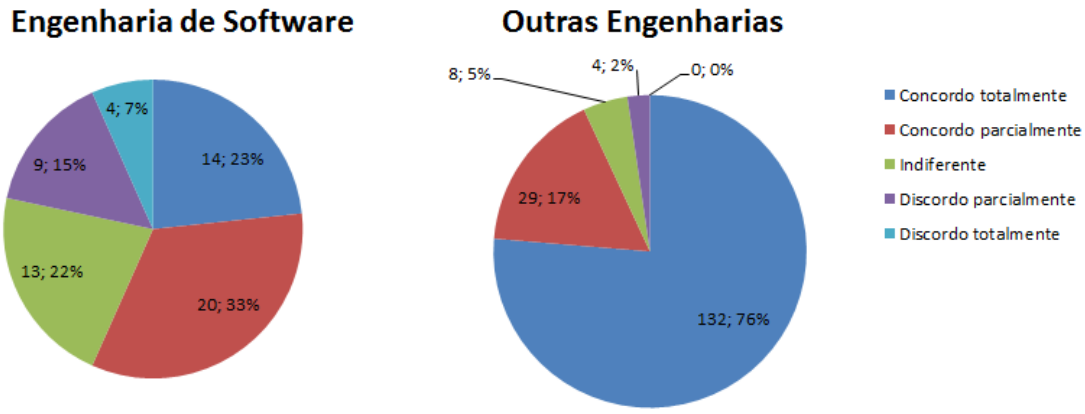

Figura 19 - Distribuição das respostas da questão 2 do survey

A percepção de que aprender a programar é um processo difícil é quase um consenso entre os indivíduos que não pretendem seguir o campo da engenharia de software, apresentando diferença estatística significativa quando as respostas são comparadas, ressaltando a ausência de respostas totalmente discordantes dentre os respondentes pertencentes ao grupo das outras engenharias.

Esses resultados são compatíveis com os apresentados nas entrevistas relatadas por Eckerdal et al. (2005), onde alguns dos entrevistados comentam sobre o consenso geral de que tentar aprender a programar é uma tarefa diferenciada, indicada para os que procuram desafios. Dentre as causas de tal pensamento, está

a dificuldade de se atingir os níveis de abstração necessários para se compreender o assunto de forma clara (BARROS et al., 2005; ESTEVES et al, 2008).

Este resultado pode ser considerado preocupante de acordo com os argumentos apresentados por Bennedsen e Caspersen (2007), de que se essa concepção recebe endosso, se enraizando dentre a comunidade de alunos, pode acarretar no declínio de alunos interessados em engenharia de software. O sentimento de que aprender a programar é uma tarefa intangível pode desanimar o aluno, contribuindo para o aumento do abandono de curso (BHATTACHARYA et al., 2011).

4.3.3.3 QUESTÃO 3

Aprender a programar é muito importante para a formação do engenheiro.

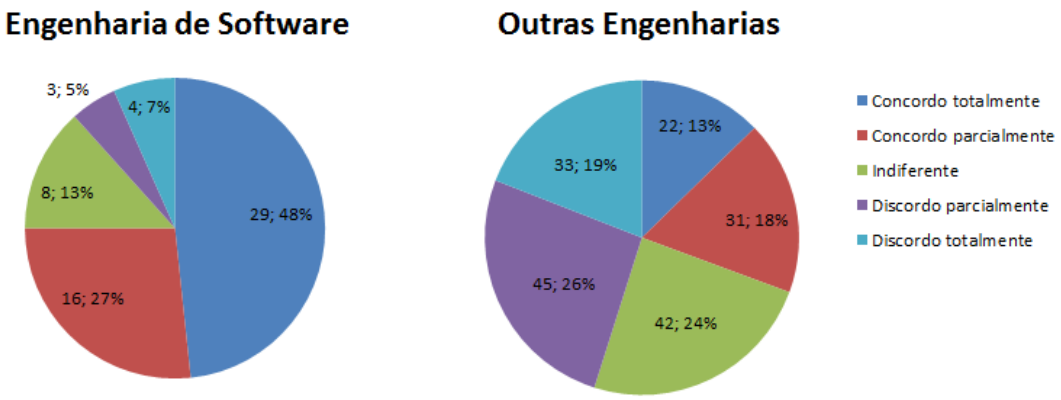

Figura 20 - Distribuição das respostas da questão 3 do survey

Na questão 3 a distribuição das respostas apresentou uma diferença significativa entre os dois grupos, mostrando que os respondentes de engenharia de software, ao contrário dos respondentes de outras engenharias, em sua maioria consideram que aprender a programar é importante para a formação do engenheiro. Essa pergunta apesar de parecer óbvia, se mostra relevante principalmente quando se pensa que o aluno ainda não escolheu em qual campo da engenharia vai se especializar, podendo influenciar na mudança de ideia (ECKERDAL et al., 2005; GUZDIAL, 2010). Mesmo os respondentes do grupo de engenharia de software

apresentam uma taxa significativa (25%) de sujeitos que ainda não compreenderam o papel da programação na engenharia.

Em termos de motivação isso é um erro, já que um indivíduo que não possui compreensão ou não entende o motivo do porquê de estar estudando certo assunto tende a um desempenho inferior aos que possuem essa compreensão (MENDES, 2012; GUZDIAL, 2010). Ter noção das inferências que os computadores e a programação de maneira geral exercem no dia-a-dia, onde já se mostram consolidados, demonstram-se como o primeiro e fundamental passo para se aprender a programar (GODWIN-JONES, 2005). Ressaltar a importância da programação para o futuro profissional do estudante é uma das maneiras de se extimular a sua motivação extrínseca, aquela que é considerada a mais decisiva, e mais impulsiona o indivíduo a se interessar por um determinado tema (JENKINS, 2002).

4.3.3.4 QUESTÃO 4

Cursar as disciplinas de Álgebra e Cálculo facilita aprender a programar.

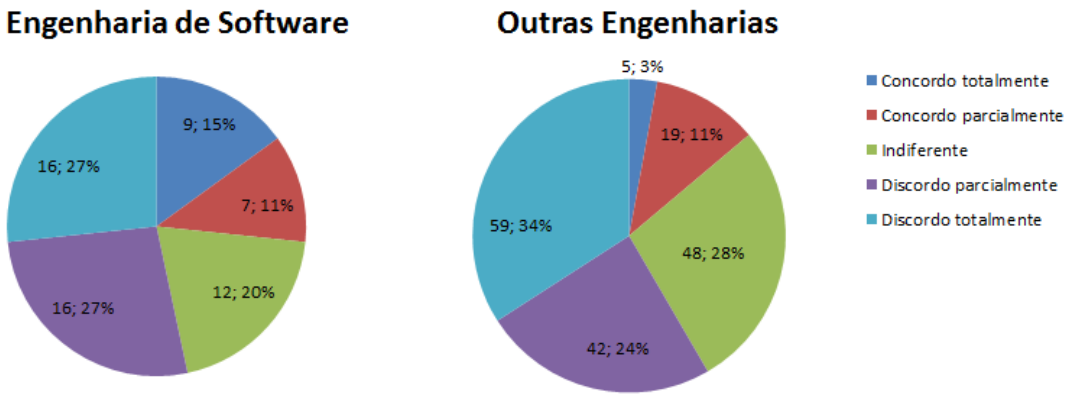

Figura 21 - Distribuição das respostas da questão 4 do survey

Os dados dos resultados da questão 4 indicam que os indivíduos dos dois grupos não relacionam o aprendizado algébrico de outras disciplinas que também fazem parte do currículo de engenharia como agregadoras ao ensino-aprendizado de programação. Essa hipótese difere do encontrado na literatura (CASTRO et al., 2008; SANTOS et al., 2013; ZHANG, 2009, 2010), onde conhecimentos matemáticos sólidos, e noções de vetores, são consideradas fundamentais na melhora do processo de aprendizagem de programação.

Apesar de os resultados entre o grupo de engenharia de software e o de outras engenharias não apresentarem tanta diferença, é importante uma comparação mostrando que até mesmo entre aqueles que na teoria são mais motivados e tem mais facilidade, que são os pertencentes ao grupo de engenharia de software, apresentam dificuldade em relacionar os temas. Isso pode ter diversas causas, dentre elas o descompasso entre os currículos acadêmicos (FORTE e GUZDIAL, 2005), e a didática usada pelo professor (FUNABIKI et al., 2013, JENKINS, 2002).

4.3.3.5 QUESTÃO 5

Estudar em grupo é uma prática que facilita a aprender a programar.

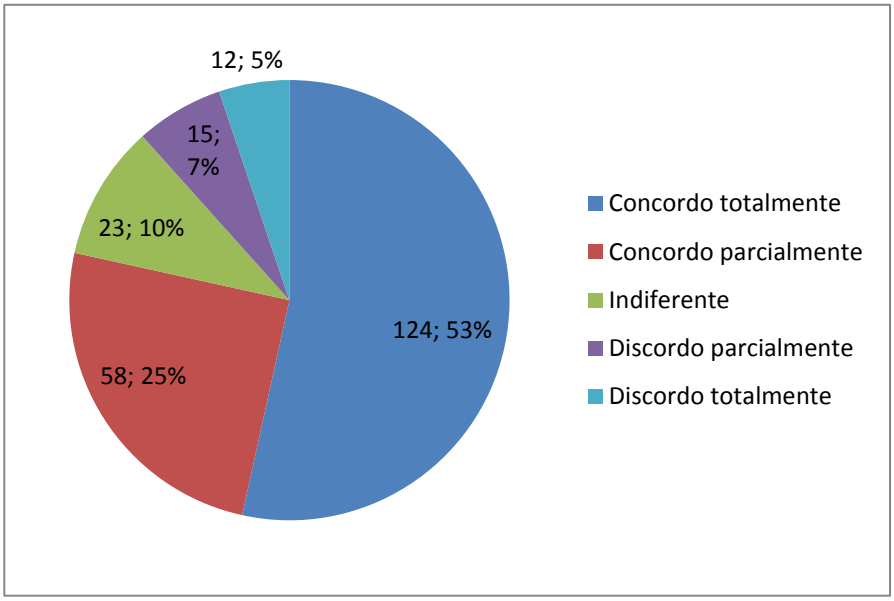

Figura 22 - Distribuição das respostas da questão 5 do survey

Os trabalhos de Alves et al. (2012), Phuong et al. (2009), dentre outros, propõem o estudo em grupo como um método de trabalhar o ensino de programação em diferentes aspectos, como a ajuda com a matemática, incentivos motivacionais, *feedback*, etc (FERNANDEZ-MEDINA et al., 2013; GUZDIAL, 2010).

As respostas da questão 5 mostram que a percepção geral dos indivíduos está de acordo com essas hipóteses, de que de fato estudar em grupo é um diferencial no processo de aprendizagem de programação. O estudo em grupo é importante principalmente para promover a troca de conhecimento entre os indivíduos, aumentando a confiança geral como um todo (PHUONG et al., 2009).

4.3.3.6 QUESTÃO 6

A forma de ensinar do professor é fundamental para aprender a programar.

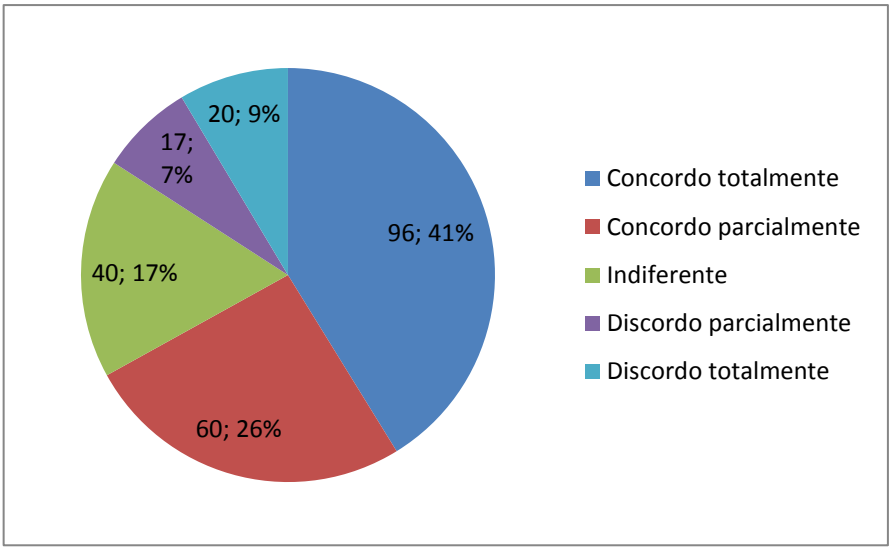

Figura 23 - Distribuição das respostas da questão 6 do survey

As respostas indicam concordância com a afirmação, endossando o apresentado por Chang et al. (2000), Jenkins (2002), Santos et al. (2011). O professor não resume sua importância a sua didática, mas possui outros papéis como o de supervisor, instrutor, mediador de conflitos, estimulador intelectual e motivacional (Funabiki et al., 2013).

4.3.3.7 QUESTÃO 7

A prática de exercícios facilita para aprender a programar.

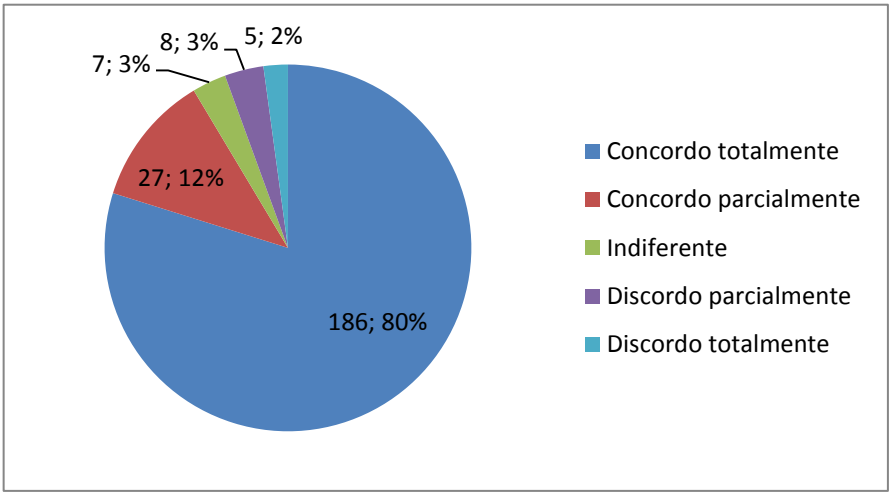

Figura 24 - Distribuição das respostas da questão 7 do survey

Outra das questões em que o conjunto das respostas foi bem mais homogêneo, concordando com o proposto por Esteves et al. (2008), Rountree et al. (2002), Santos et al. (2011, 2013). O indivíduo que está em processo de aprendizagem de programação de praticar diariamente, de maneira a cultivar e exercitar uma nova maneira de pensar necessária para se dominar uma nova inteligência (WU, 2011).

4.3.3.8 QUESTÃO 8

O feedback imediato nos exercícios e provas facilita pra aprender a programar.

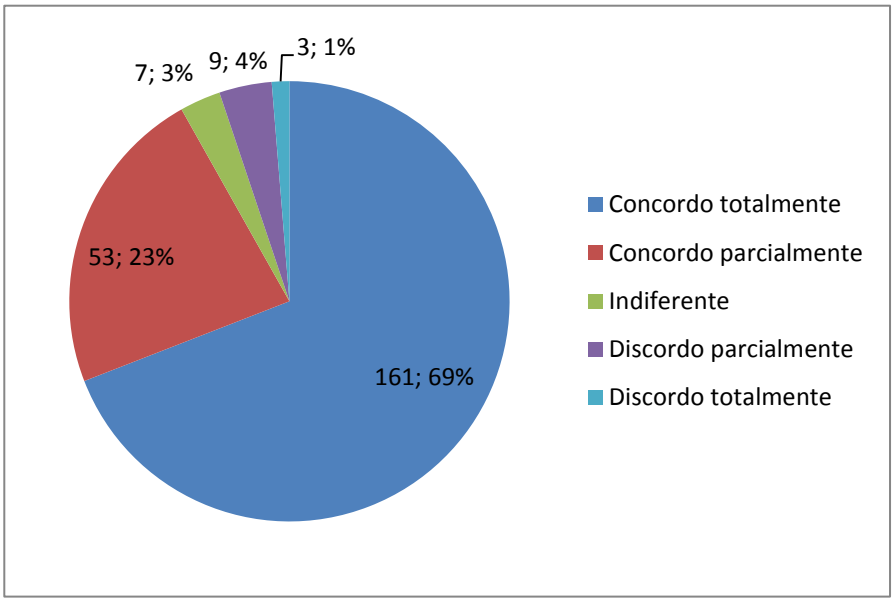

Figura 25 - Distribuição das respostas da questão 8 do survey

A importância do *feedback* constante é levantando pelos mesmos autores que promovem a necessidade da prática constante de exercícios, pois argumenta-se que um indivíduo que não confirma a solução e o seu desempenho em uma tarefa sente-se frustrado e com o pensamento de que seu esforço e trabalho estão sendo desperdiçados, gerando desmotivação (SANTOS et al., 2011, 2013).

4.3.3.9 QUESTÃO 9

É possível aprender a programar apenas assistindo as aulas.

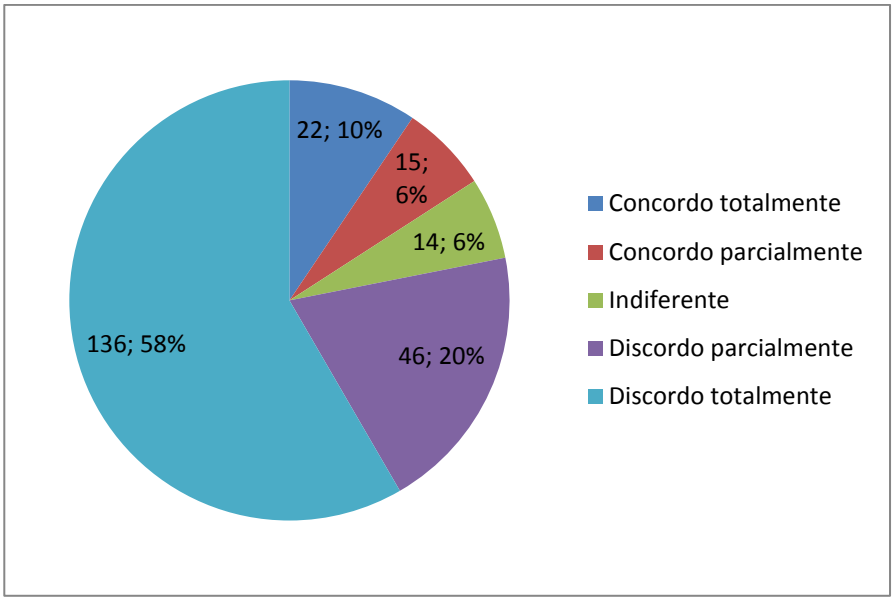

Figura 26 - Distribuição das respostas da questão 9 do survey

As respostas dessa questão indicam que a grande maioria dos indivíduos discorda em relação à afirmação. O que diz que as aulas de maneira geral são consideradas insuficientes, seja pela extensão do conteúdo, seja pela necessidade de alguma forma de complementação como vídeo aulas, frequentar monitorias, etc.

Forte e Guzdial (2005) propõem que justamente pela alta necessidade de se praticar e exercitar a atividade de programar e suas competências, apenas o tempo disponível em sala de aula não é suficiente, por mais que parte dessas aulas sejam práticas e realizadas em laboratório, pois a extensão do conteúdo e sua complexidade demandam tempo adicional de contato (ZHANG, 2010). Essa informação também possui relação com a dificuldade de se estudar o assunto apenas na sala de aula, já que várias técnicas estão envolvidas como a leitura de códigos de terceiros, prática em computador, cultivo das habilidades de lógica e resolução de problemas (JENKINS, 2002, MATHEWS et al., 2012).

4.3.3.10 QUESTÃO 10

Compreender e interpretar um problema é a etapa mais complexa e difícil no processo de desenvolvimento de um programa.

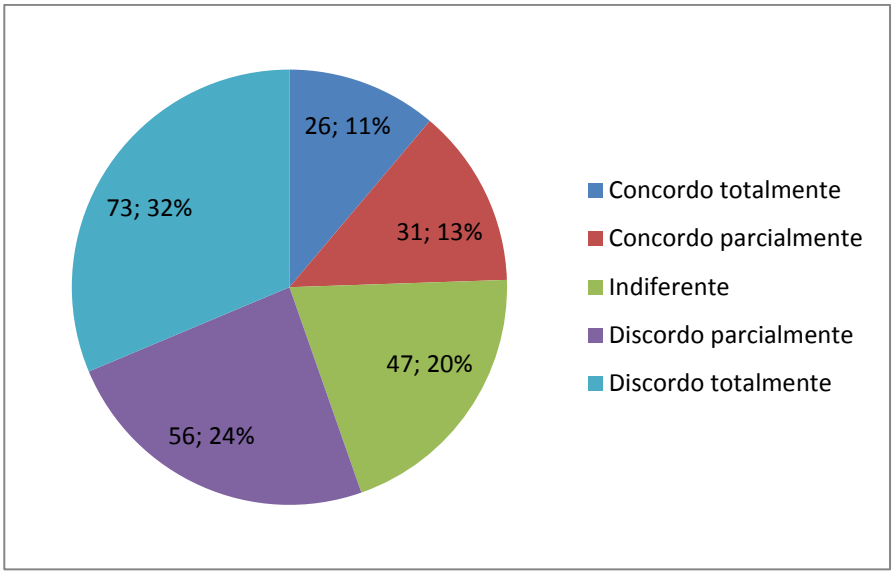

Figura 27 - Distribuição das respostas da questão 10 do survey

As respostas dessa questão foram mais homogêneas, apesar de uma pequena indicação para a discordância. Isso vai ao encontro do proposto por autores como Castro (2008), Gomes e Mendes (2010), que propõem que dependendo do indivíduo e sua percepção ele vai ter dificuldades em um dos diferentes estágios que compõem uma solução computacional: analisar e interpretar um problema, transcrever a solução para algoritmo, transformar o algoritmo em código (JENKINS, 2002; GUZDIAL, 2010; WU, 2011; ZHANG, 2010).

4.3.3.11 QUESTÃO 11

O tempo de estudo dedicado a aprender programação deve ser superior ao de outros conteúdos.

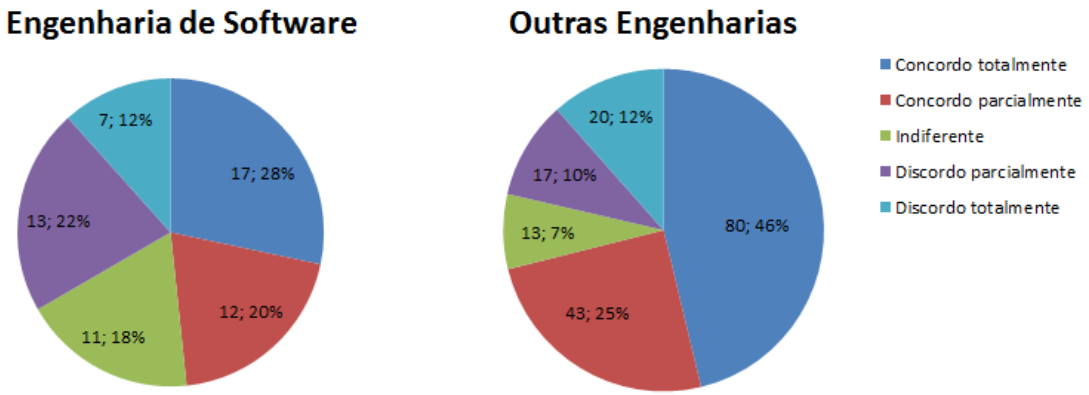

Figura 28 - Distribuição das respostas da questão 11 do survey

Nessa questão os dois grupos de respostas se mostraram bastante diferentes. Na percepção do grupo de engenharia de software, de maneira geral o tempo de estudo dedicado a aprendizagem de programação não deve ser muito superior ou inferior ao de outras disciplinas. Já no grupo que engloba os respondentes de outras engenharias, percebe-se claramente uma tendência concordante a questão.

Não é possível determinar uma causa direta para esses resultados, mas entre o proposto pela literatura pode-se ressaltar, que no caso do grupo dos respondentes de outras engenharias, isso pode ocorrer tanto por autocrítica, por a disciplina disciplina apresentar um conteúdo aceito como difícil, os indivíduos se forçam a estudar o conteúdo em que possuem maior deficiência em relação aos demais; ou pela dificuldade que possuem em conseguir estudar o tema em si, necessitando de mais tempo para que a matéria seja assimilada (ECKERDAL et al., 2005; FORTE e GUZDIAL, 2005; PHUONG et al. 2009; ZHANG, 2009). O oposto dessa lógica pode então ser aplicada para justificar a percepção do grupo de engenharia de software em relação a questão, pois esses sujeitos teoricamente apresentam mais facilidade na hora de estudar, e de maneira geral apresentam uma compreensão maior a respeito da atividade de programar em relação aos outros conteúdos.

4.3.3.12 QUESTÃO 12

Uma infraestrutura adequada (computadores, softwares, internet, biblioteca) facilita aprender a programar.

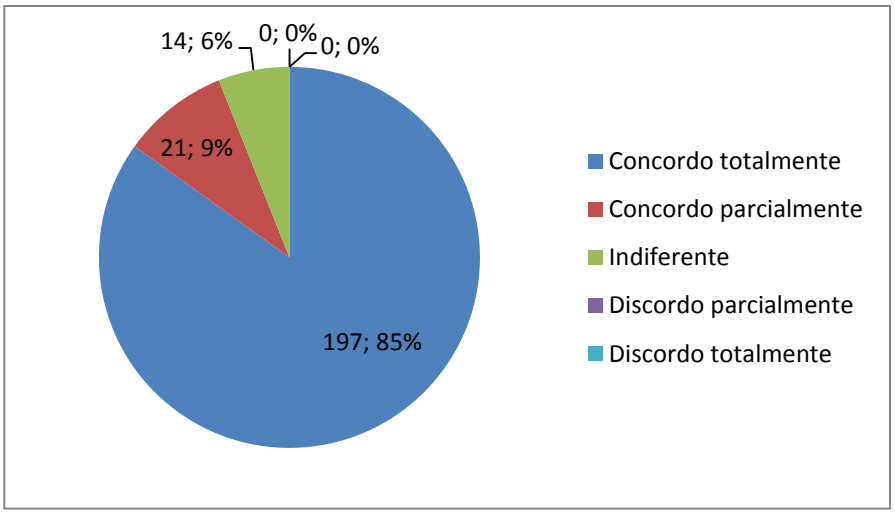

Figura 29 - Distribuição das respostas da questão 12 do survey

Este é o primeiro e único item do questionário em que não houve respostas discordantes. Apesar do tema não ter sido abordado na literatura pesquisada, a percepção dos indivíduos é que de fato uma infraestrutura adequada é essencial no processo de aprendizagem. Esse resultado é esperado principalmente quando comparado com os resultados de outras questões, como as relacionadas à prática de exercícios, horas de estudo, e estudo em grupo, já que todas se beneficiam de uma boa infraestrutura que comporte locais para a prática e estudo de programação.

4.3.3.13 QUESTÃO 13

Aprender a programar influencia na definição de qual engenharia seguir.

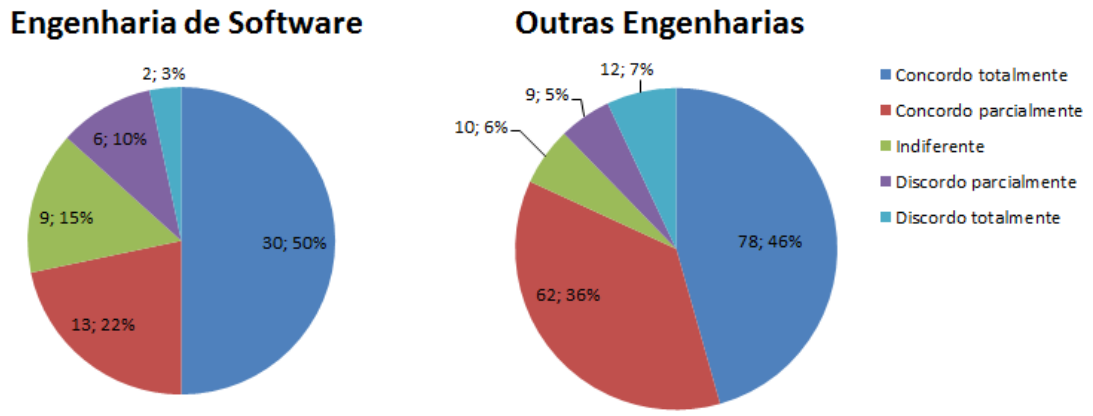

Figura 30 - Distribuição das respostas da questão 13 do survey

A origem dessa questão não possui raízes diretas na literatura estudada, mas sim no contexto específico da instituição FGA e nos resultados do instrumento preliminar. Por ser uma disciplina que consta na grade das obrigatórias do segundo semestre, um indivíduo padrão, que esteja cursando sua primeira graduação e esteja acompanhando o fluxo acadêmico, vai passar pela disciplina de introdução à computação antes de ter feito a opção de qual ramo da engenharia ele irá cursar.

O objetivo desse item era então o de testar na percepção dos indivíduos se as aulas de computação básica e aprender ou não a programar poderiam influenciar na sua escolha de engenharia a seguir, podendo, por exemplo, desistir de seguir a engenharia de software devido às dificuldades passadas durante o período de aprendizagem de programação.

E é por isso que a comparação dos dois grupos se torna importante nesse contexto, mesmo que os resultados possuam a mesma tendência em concordarem com a afirmação, pois as motivações por trás dessa concordância podem ser diferentes. No caso do grupo de engenharia de software isso pode ocorrer devido a percepção de que o indivíduo considera que aprendeu a programar, e por isso vai continuar seguindo por essa especialização. Já no caso do grupo referente as outras engenharias, isso pode significar que por esses indivíduos não terem aprendido a programar com excelência, se sentem inclinados a optarem por outras engenharias (BENNEDSEN e CASPERSEN, 2007; FORTE e GUZDIAL, 2005).

4.3.3.14 QUESTÃO 14

Os monitores são importantes para aprender a programar.

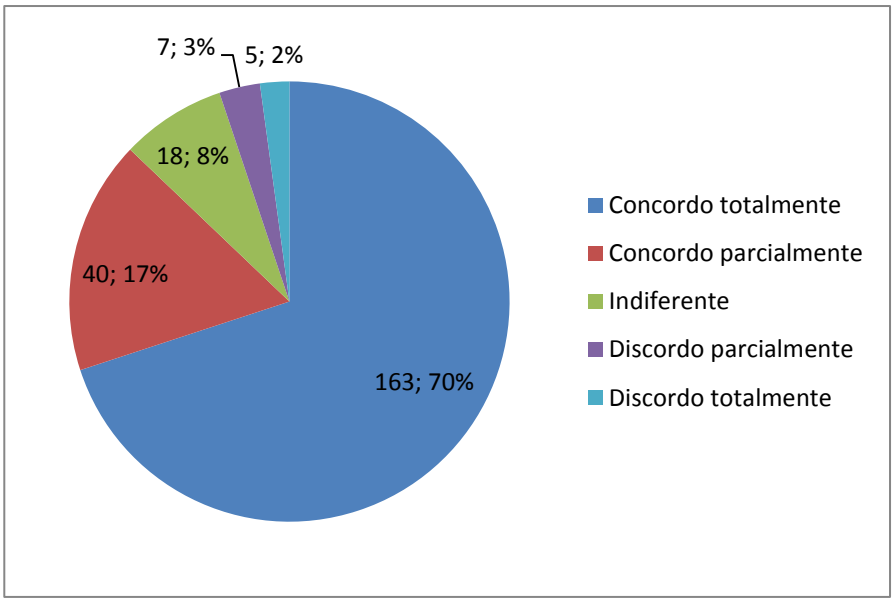

Figura 31 - Distribuição das respostas da questão 14 do survey

A forte tendência a concordância apresenta pelos resultados dessa questão possui embasamento não apenas na literatura como nas respostas das questões anteriores. Martins et al. (2010), e Phuong et al. (2009) falam em seus trabalhos sobre a importância da supervisão para o aprendizado de programação, e o monitor, na ausência do professor, é o principal agente a assumir esse papel. Sua presença, nos horários dentro e fora de classe, é fundamental para que o aluno nunca esteja sem feedback, principalmente ao realizar exercícios e sanar dúvidas, o que está de acordo com os resultados das questões 7, 8, 9 e 11.

4.3.3.15 QUESTÃO 15

O tipo de linguagem não tem nenhuma influência sobre aprender a programar.

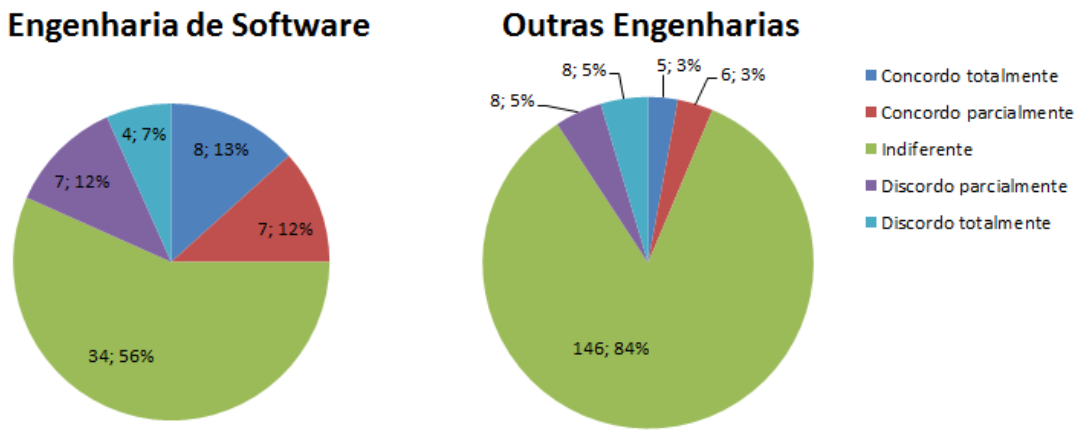

Figura 32 - Distribuição das respostas da questão 15 do survey

Essa foi a questão com o maior número de respostas indiferentes a afirmação, o que pode ser analisado de duas maneiras. A primeira é que como se tratam de indivíduos frequentando uma disciplina introdutória de programação, eles em sua maioria não se sentiram confortáveis em responder uma questão sobre algo que não dominam o suficiente para fazer juízo de valor. A segunda é que de fato na percepção dos indivíduos não existe linguagem de programação que facilite o aprendizado, raciocínio que vai ao encontro do proposto por Jenkins (2002), que explica que uma linguagem diferente nunca vai ser consenso entre todos os indivíduos, pois existem diferenças cognitivas na hora de se processar a informação que diferem de pessoa para pessoa.

Vale notar que no caso do grupo de engenharia de software apesar da tendência das respostas também ser a da indiferença, essa tendência já diminui consideravelmente. Isso pode ser explicado pelo fato de que de maneira geral esses indivíduos possuem mais interesso pelo assunto, aprender uma linguagem específica pode ser considerada melhor para um determinado mercadod e trabalho por exemplo (GOMES e MENDES, 2007).

4.3.3.16 QUESTÃO DISCURSIVA

Quais outros fatores você considera que facilitam ou dificultam a aprendizagem de programação? Registre a seguir: (OPCIONAL)

A questão que finaliza o *survey* é a única de preenchimento optativo. Essa decisão foi tomada na tentativa de que apenas os sujeitos que realmente tivessem o desejo de contribuir, e apresentassem algum interesse pelo tema respondessem, tornando as respostas mais propensas a terem sido produto de alguma opinião ou situação prévia.

Ao todo 35 (15%) sujeitos responderam a questão. Suas respostas foram então analisadas e agrupadas em categorias de acordo com o apresentado no quadro 12.

Quadro 12 - Resultados da questão discursiva

Fator Citado	Número de Incidência
Presença de monitores durante a aula.	6
Aumento da quantidade de monitores disponíveis no geral.	11
Uma melhor estrutura de laboratório para estudo e prática de exercícios.	12
Relacionar melhor programação com outros conteúdos.	3
Um período de aulas introdutórias mais básico antes do início efetivo da disciplina.	2
Melhor estrutura de internet.	8
Uma melhor didática do professor.	1
Aulas de introdução a programação que utilizem a linguagem Java como padrão.	1
Disponibilização de vídeo-aulas e materiais complementares.	2

Esses fatores propostos pelos estudantes possuem relação com o contemplado pela literatura e pelo instrumento de pesquisa. E demonstram a clara percepção de que aprender a programar demanda recursos e estrutura, além de uma necessidade de que sempre aja algum tipo de supervisão.

5. CONCLUSÔES E TRABALHOS FUTUROS

O presente trabalho foi iniciado com o objetivo de se investigar os fatores que exercem influência no aprendizado introdutório de programação. Tendo esse conceito em vista, foi feita uma revisão sistemática com o intuito de encontrar na literatura o estado da arte sobre o assunto, ou seja, de acordo com os estudiosos acerca do tema quais são esses fatores, seus aspectos e características. Chegou-se então a conclusão de que cinco fatores são os principais a influenciar o processo de ensino-aprendizagem de programação. Dentre eles dois foram escolhidos como objeto de estudo: a motivação e a atividade de programar.

Em seguida, foi proposta uma metodologia para a construção de um instrumento de pesquisa visando verificar a existência desses fatores, e a percepção sobre eles, em um contexto específico, o de estudantes de Engenharia de Software na Universidade de Brasília, *campus* do Gama. Para isso foi primeiramente efetuada uma pesquisa documental de campo, com o objetivo de coletar dados relevantes; e desenvolvido um instrumento exploratório, denominado instrumento preliminar, que possui fundamentações mais rudimentares e teóricas, sem metodologia de construção e análise pré-estabelecidas.

O objetivo desse instrumento preliminar e da pesquisa documental foram a de mera verificação de indícios desses fatores, utilizando como amostra populacional os alunos matriculados na disciplina Computação Básica.

Após a aplicação do instrumento preliminar, e verificada alguma incidência dos fatores na amostra da população, seus resultados serviram como insumo, junto da base teórica encontrada na literatura, e os resultados obtidos na pesquisa documental, na construção e desenvolvimento de um instrumento de pesquisa.

Esse instrumento, denominado como *survey*, utilizou como base metodológica em sua construção diretrizes já estabelecidas e comprovadas pela literatura a respeito do desenvolvimento de instrumentos de pesquisa. Esse instrumento não só deveria agregar as bases teóricas, mas permitir uma análise aprofundada dos dados coletados, através do uso de técnicas estatísticas de validação e análise.

Após a aplicação do *survey* e tratamento dos dados, foi possível se chegar a certas conclusões não só sobre as variáveis, mas sobre a qualidade do instrumento

como um todo. Descobriu-se através da realização da análise fatorial, e do cálculo do alpha de cronbach, que o instrumento proposto na verdade mede mais fatores do que os dois inicialmente propostos, o que significa que seus resultados podem apresentar variação quando aplicado em amostras diferentes da população, e por isso suas conclusões não possuem aplicabilidade geral, suficientes para a construção de uma teoria (LAROS, 2012).

Esse resultado tem origem em dois pontos: na própria amostra e no objeto de estudo. A amostra, como mostram os dados relacionados à sua identificação, possui uma constituição muito heterogênea, sendo composta de indivíduos de diferentes cursos, idades, formação, ou seja, apresentam características muito distintas e heterogêneas, o que não é o mais adequado para instrumentos que vão ser submetidos à análise fatorial (LAROS, 2012).

O objeto de estudo, como apontam os estudos apresentados na fundamentação teórica, e os próprios resultados do *survey*, é multifacetado e necessita de muitas variáveis envolvidas na sua análise. Esses aspectos levam a produção de ruído nos dados, pois um conjunto composto de sujeitos tão diferentes e um objeto de estudo tão amplo acabam possuindo muitas especificidades próprias, que não podem ser totalmente medidas (MAROCO, 2003; KREBS; BERGER; FERLIGOJ, 2000).

Tendo isso em mente, gostaria-se de propor algumas alterações a serem feitas no *survey*, de maneira a tentar aumentar sua confiabilidade, e aumentar sua capacidade de medir os fatores originalmente desejados. O critério utilizado na decisão de que questões devem ser excluídas, é o valor das cargas fatoriais presentes na matriz de componentes (LAROS, 2012; MAROCO, 2003), e as sugestões de questões a serem adicionadas tem origem na literatura estudada. Essas alterações são apresentadas no quadro 13.

Quadro 13 - Proposta de aperfeiçoamento do survey

Questões excluídas	Questões adicionadas
Q7. A prática de exercícios facilita para aprender a programar.	Atingir uma boa menção na disciplina de introdução a programação é importante para mim (ROUNTREE et al. 2002).

Q8. O *feedback* imediato nos exercícios e provas facilita para aprender a programar	Quando uma dúvida sobre programação surge não é difícil obter uma orientação a respeito de uma solução (PHUONG et al. 2009).
Q15. O tipo de linguagem não tem nenhuma influência sobre aprender a programar.	Possuir um computador pessoal facilita a aprender a programar (BYRNE e GERRY, 2001).

Apesar dessa implicação, conseguiu-se observar, através do processo de análise de clusters e das estatísticas descritivas, como a média e a mediana, que os resultados possuíam um nível de consistência e relacionamento que permite a inferência de conclusões amparadas pela estatística. E é a partir desse último processo, o de inferência e análise dos resultados que se chega ao objetivo final do trabalho, que é o de indicar os principais aspectos que aparentam estar exercendo influência no processo de ensino-aprendizagem dos indivíduos.

Dentre as conclusões que se pode chegar analisando as questões, está a de que os indivíduos que pretendem seguir o campo da engenharia de software apresentam diferenças significativas em relação aos demais indivíduos quando suas respostas são comparadas.

De maneira geral os respondentes pertencentes ao grupo de software, quando comparados aos demais, apresentam maior facilidade com o tema da programação, sendo de maneira geral mais contextualizados, motivados e com uma facilidade maior na hora de estudar esse tema.

A análise dos resultados também permite apontar falhas no processo de aprendizagem de programação do contexto estudado, pois mostra que os estudantes não conseguem relacionar muito bem programação com outros campos da engenharia, nem com disciplinas consideradas básicas e correlatas pela literatura. A análise pode ir além, apontando que de maneira geral os indivíduos tem pouca noção de como a programação possui um envolvimento intrínseco com o cotidiano na atualidade (GODWIN-JONES, 2005), e da importância da programação no mercado de trabalho geral como um todo (JENKINS, 2002).

Outra conclusão que é possível chegar-se através da análise dos dados é a da necessidade dos indivíduos por uma estrutura adequada, e supervisão constante,

como fatores decisivos no processo de aprendizagem de programação. Esse resultado pode ser observado tanto nas respostas da questão discursiva quanto nas que tratam do tema, como as questões 8, 9, 12 e 14. O indivíduo que está na fase inicial de aprendizagem de programação precisa da confiança de que suas dúvidas serão sanadas o mais rápido possível (BHATTACHARYA et al., 2011).

Pelo resultado das questões 7 e 11, percebe-se que a percepção dos respondentes antentam para a necessidade de se estar sempre treinando e exercitando as competências envolvidas em programação. Por se tratar de uma nova linguagem com características próprias, a programação deve ser exercitada ao limite, assim como a tabuada quando se é mais novo, por exemplo (GOMES e MENDES, 2007).

Todos esses resultados se tornam mais relevantes a luz das respostas das questões 3 e 13. Esses resultados apontam que a percepção dos sujeitos é de que aprender ou não a programar tem sérias implicações em relação ao caminho de engenharia a ser seguido, porém ao mesmo tempo a percepção é de que aprender a programar não é algo essencial para a formação do engenheiro de maneira geral.

Esses dados são relevantes e podem explicar o índice de pessoas que deixam ou não de optar pela engenharia de software, por exemplo. Se o sujeito não relaciona a importância da programação às outras engenharias, após se sentir frustrado com sua primeira experiência com programação, a tendência é que esse indivíduo se afaste ao máximo de qualquer coisa que em sua visão possa ser considerada correlata (BENNEDSEN e CASPERSEN, 2007). Mas essas consequências não se limitam a software, já que outras engenharias apresentam mais familiaridade com programação em seus temas relacionados, como a engenharia de eletrônica. Isso pode inclusive influênciar os interesses de trabalho em outras engenharias, pois sujeitos podem se afastar de temas que relacionam engenharia de energia com computação, como a simulação de fluídos entre outros.

Essas análises mostram como o fato de aprender ou não a programar pode ser decisivo em vários níveis, sendo esta uma habilidade que vai ser cada vez mais exigida, principalmente nos que estão fazendo ou pretendem fazer um curso de tecnologia. Porém a abordagem atual não tem acompanhado a evolução constante pela qual a computação e consequentemente a programação vem sofrendo.

Programar ainda é tratado como algo que só possui relação com graduações de computação e com trabalho, sendo que suas implicações no cotidiano e em outros cursos ainda são em parte ignoradas.

Conclui-se então a importância da necessidade de estudar-se mais a fundo esse tema, cuja complexidade demanda estudos ainda mais aprofundados, e preferencialmente com um diálogo maior com a psicologia e pedagogia. Apenas assim será possível de fato uma melhora no processo de aprendizagem de programação.

Dentre os trabalhos que podem ser originados a partir deste, o primeiro seria o de promover o aprimoramento do *survey* aqui proposto, utilizando como ponto de partidas as informações contidas no documento e a proposta de aprimoramento apresentada previamente.

Outro trabalho interessante seria baseado na pesquisa de Eckerdal et al. (2005), através da aplicação de entrevistas com grupos focais. Assim seria possível uma maior compreensão das especificidades inerentes à aprendizagem de programação, e como elas afetam os diferentes grupos de alunos. Uma ideia interessante seria por exemplo focar exclusivamente no público feminino, já que de acordo com dados apresentados na literatura (FORTE e GUZDIAL, 2005; BENNEDSEN e CASPERSEN, 2007) esse é um grupo que tende a evitar o contato com programação.

BIBLIOGRAFIA

ANDRÉ, M. *Estudo de caso em pesquisa e avaliação educacional*. Brasília: Liber Livro Editora, 2005.

ALVES, L. M. et al. A multinational case study on using diverse feedback types applied to introductory programming learning. In: Proceedings of the 2012 IEEE Frontiers in Education Conference (FIE). IEEE Computer Society, 2012. p. 1-6.

BENNEDSEN, J.; CASPERSEN, M. E**.** *Failure rates in introductory programming.* ACM SIGCSE Bulletin, Nova York, v. 39, n. 2, p. 32-36, 2007.

BHATTACHARYA, P. et al. A Collaborative Interactive Cyber-learning Platform for Anywhere Anytime Java Programming Learning. In: Advanced Learning Technologies (ICALT), 2011 11th IEEE International Conference on. IEEE, 2011. p. 14-16.

BIOLCHINI, J. et al**.** *Systematic review in software engineering*. System Engineering and Computer Science Department COPPE/UFRJ, Technical Report ES, v. 679, n. 05, 2005.

BRASIL. Ministério da Educação. Conselho Nacional de Educação. *Parecer CNE/CES nº136/2012*. Brasília**:** MEC/CNE, 2012. 22 p.

BYRNE, P.; GERRY, L. *The effect of student atributes on success in programming.* ACM SIGCSE Bulletin, v. 33, n. 3, p. 49-52, 2001.

CHANG, K. E. et al. A programming learning system for beginners – A completion strategy approach. Education, IEEE Transactions on, v.43, n.2, p.211-220, 2000.

CHUN, S.; RYOO, J. *Development and application of a web-based programming learning system with LED display kits.* In: Proceedings of the 41st ACM technical symposium on Computer science education. ACM, 2010. p. 310-314.

CUMMINS, R. A.; GULLONE, E. Why we should use 5-point Likert scales: The case for subjective quality of life measurement. In:Proceedings, second international conference on quality of life in cities. 2000. p. 74-93.

DAMÁSIO, B. F. Uso da análise fatorial exploratória em psicologia. Avaliação psicológica, v. 11, n. 2, p. 213-228, 2012.

DANCEY, P. C., REIDY, J. *Estatística sem matemática para Psicologia: usando o SPSS para Windows*. [S.l.]: Artmed, 2006.

DE BARROS, L. N. et al. *A tool for programming learning with pedagogical patterns.* In: Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange. ACM, 2005. p. 125-129.

DE CASTRO, T. H. C. Sistematização da aprendizagem de programação em Grupo. Tese de doutorado - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2011.

_____. *The analysis of a case study for group programming learning.* In: Advanced Learning Technologies. ICALT. 2008. p. 850-854.

DE OLIVEIRA, K. L. et al. O psicólogo comportamental e a utilização de técnicas e instrumentos psicológicos. Psicologia em Estudo, v. 10, n. 1, p. 127-135, 2005.

DRUCKER, P. F. Desafios gerenciais para o século XXI. São Paulo: Pioneira, 1999.

ECKERDAL, A.; THUNÉ, M.; BERGLUND, A. What does it take to learn'programming thinking'?. In: Proceedings of the first international workshop on Computing education research. ACM, 2005. p. 135-142.

ESTEVES, M. et al. *Contextualization of programming learning: A virtual environment study.* In: Frontiers in Education Conference. IEEE, 2008. p. F2A-17-F2A-22.

FERNANDEZ-MEDINA, C. et al. *Assistance in computer programming learning using educational data mining and learning analytics.* In: Proceedings of the 18th ACM conference on Innovation and technology in computer science education. ACM, 2013. p. 237-242.

FIGUEIREDO, R. M. C.; RIBEIRO Jr, L. C. M.; RAMOS, C. S.; CANEDO, E. D. Graduação em Engenharia de Software *versus* Graduação em Engenharia de Computação: uma reflexão. III Fórum de Educação em Engenharia de Software (FEES). Salvador, Bahia. 2010.

FORTE, A.; GUZDIAL, M. Motivation and nonmajors in computer science: identifying discrete audiences for introductory courses. Education, IEEE Transactions on, v. 48, n. 2, p. 248-253, 2005.

FREITAS, A. L. P.; RODRIGUES, S. G. A avaliação da confiabilidade de questionários: uma análise utilizando o coeficiente alfa de Cronbach. Anais do Simpósio de Engenharia de Produção–SIMPEP, Bauru, São Paulo, v. 12, 2005.

FUNABIKI, N. et al. *An extension of fill-in-the-blank problem function in Java programming learning assistant system*. In: Humanitarian Technology Conference (R10-HTC), 2013 IEEE Region 10. IEEE, 2013. p. 85-90.

GIANNOTTI, E. *Algorithm animator: a tool for programming learning.* In:ACM SIGCSE Bulletin. ACM, 1987. p. 308-314.

GODWIN-JONES, Robert. Emerging technologies: Messaging, gaming, peer-to-peer sharing: Language learning strategies & tools for the millennial generation. Language learning & technology, v. 9, n. 1, p. 17-22, 2005.

GOMES, A.; MENDES, A. J. *Studies and proposals about initial programming learning.* In: Frontiers in Education Conference (FIE), 2010 IEEE. IEEE, 2010. p. S3F-1-S3F-6.

GOMES, A.; MENDES, A.J. *Learning to program – difficulties and solutions.* In: International Conference on Engineering Education – ICEE. 2007.

GUZDIAL, M. *Why is it so hard to learn to program?* In: ORAM, A; WILSON, G. (Org). Making software: what really works, and why we believe it. O'Reilly Media Inc. 2010. p. 111-123.

IBRAHIM, Ali A.; AL RIKABI, Zainib HA; AHMED, Najwa Sh. Using Hierarchical Cluster and Factor Analysis to Classify and Built a phylogenetic Tree Species of ND1

Mitochondria. *Journal of Al-Nahrain University*, Bagdá, vol. 17, n. 1, p. 114 – 122, 2014.

JADZGEVIČIENĖ, V.; URBONIENĖ, J. 2012. *Cooperation in programming learning.* In: Proceedings of the 12th Koli Calling International Conference on Computing Education Research. ACM, 2012. p. 143-144.

JENKINS, T. *On the difficulty of learning to program.* In: Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences. 2002. p. 53-58.

KLINE, P. *The handbook of psychological testing*. London: Routledge, 1995.

KREBS, D.; BERGER, M; FERLIGOJ, A. Approaching achievement motivation-comparing factor analysis and cluster analysis. New approaches in applied statistics, Metodoloski zvezki, v. 16, 2000.

LAROS, J. A. O uso da análise fatorial: algumas diretrizes para pesquisadores. In: Análise fatorial para pesquisadores. 2012. p. 141-160

MCGETTRICK, A, BOYLE, R, IBBETT, R, LLOYD, J, LOVEGROVE, L, and MANDER, K. (2005) Grand Challenges in Computing: Education – A Summary. The Computer Journal. The British Computer Society. Vol. 48, No. 1. pp 42-48.

MAROCO, J. *Análise estatística com utilização do SPSS*. Edições Sílabo Ltda, 2003.

MARTINS, S. W. et al. *A context for programming learning based on research communities.* In: Edicaton Engineering (EDUCON), 2010. IEEE, 2010. p. 1317-1326.

MATTHEWS, R. et al. *Merits and pitfalls of programming learning objects: a pilot study.* In: Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia. ACM, 2012. p. 293-296.

MENDES, A. J. et al. *Increasing student commitment in introductory programming learning*. In: Proceedings of the 2012 IEEE Frontiers in Education Conference (FIE). IEEE Computer Society, 2012. p. 1-6.

MILNE, I & ROWE, G. Difficulties in Learning and Teaching Programming—Views of Students and Tutors. Education and Information Technologies. Netherlands, Kluwer Academic Publishers., 55–66, 2002.

PASQUALI, L. *Instrumentos psicológicos: manual prático de elaboração.* Brasília: LabPAM/IBAPP, 1999.

PHUONG, D. D. et al. *Encouraging programming learning for novices with grouping and convincing opinions.* In: Computer Sciences and Convergence Information Technology, 2009. ICCIT'09. IEEE, 2009. p. 283-288.

RAMOS SILVEIRA, S. de F.; REIS WAKIM, V. *Custo do ensino de graduação em instituições federais de ensino superior: o caso da Universidade Federal de Viçosa*. rap—rio de Janeiro, v. 44, n. 3, p. 637-66, 2010.

RAYMUNDO, V. P. *Construção e validação de instrumentos: um desafio para a psicolinguística*. Letras de Hoje, v. 44, n. 3, 2009.

ROBINS, A., ROUNTREE, J., ROUNTREE, N. Learning and teaching programming: A review and discussion - Computer Science Education, 2003.

ROUNTREE, N. et al. *Predictors of success and failure in a CS1 course.* ACM SIGCSE Bulletin, Nova York, v. 34, n. 4, p. 121-124, 2002.

SAMPAIO, R.F.; MANCINI, M. C. *Estudos de revisão sistemática: um guia para síntese criteriosa da evidência científica.* Braz. J. Phys. Ther.(Impr.), v. 11, n. 1, p. 83-89. 2007.

SANTOS, A. et al. *A class record and reviewing system designed to promote programming learning.* In: Frontiers in Education Conference (FIE), 2011. IEEE, 2011. p. F4G-1-F4G-6.

SANTOS, A. et al. *A taxonomy of exercises to support individual learning paths in initial programming learning*. In: Frontiers in Education Conference. IEEE 2013. P. 87-93.

SBC. *Currículo de Referência da SBC para Cursos de Graduação em Computação e Informática.* Campinas: SBC, 1999. Disponível em: <http://www.sbc.org.br/index.php?option=com_jdownloads&Itemid=195&task=view.download&catid=36&cid=52>. Acesso em: 10 abr. 2014.

SERRANO CÁMARA, L. M. et al. *Evaluation of a collaborative instructional framework for programming learning.* In: Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education. ACM, 2012. p. 162-167.

SHADIEV, R. et al. *Applying Unidirectional versus Reciprocal Teaching Strategies in Web-Based Environment and Their Effects on Computer Programming Learning.* In: Advanced Learning Technologies (ICALT), 2013 IEEE 13th International Conference on. IEEE, 2013. p. 5-9.

SOLOWAY, E. and EHRLICH, K, BONAR, J., and GREENSPAN, J. (1983) What do novices know about programming? In: Shneiderman, B. and Badre, A. (Eds), Directions in Human-Computer Interactions, 27–53. Ablex, Inc., Norwood, NJ.

TAYLOR, Richard. Interpretation of the correlation coefficient: a basic review.Journal of diagnostic medical sonography, v. 6, n. 1, p. 35-39, 1990.

VIANNA, H. M. Avaliações em debate: saeb, enem, Provão. Editora Plano, 2003.

WU, B. et al. *Live Programming Learning Objects on Cloud.* In: Advanced Learning Technologies (ICALT), 2011 11th IEEE International Conference on. IEEE, 2011. p. 362-363.

ZHANG, J. *A model on adaptive example recommendation for programming learning.* In: Computer Science & Education, 2009. ICCSE'09. 4th International Conference on. IEEE, 2009. p. 1689-1692.

_____. An adaptive model customized for programming learning in e-learning. Computer Science and Information Technology (ICCSIT), 2010. IEEE, 2010. p. 443-447.

APÊNDICES

APÊNDICE I – PROTOCOLO DA REVISÃO SISTEMÁTICA

1. FORMULAÇÃO DA QUESTÃO

1.1. FOCO DA QUESTÃO

Levantar os principais registros relacionados as características da dificuldade de aprendizagem em programação, buscando, a partir delas, identificar os principais fundamentos e fatores a serem trabalhados na tentativa de facilitar e aumentar as chances de sucesso nessa aprendizagem.

1.2. QUALIDADE E AMPLITUDE DA QUESTÃO

1.2.1. PROBLEMA:

A habilidade de programar é muito requisitada no mercado de trabalho. Parte integrante dos cursos de graduação em computação, o aprendizado de programação é indispensável dentre as habilidades dos egressos. Estudos mostram que aprender a programar é uma tarefa difícil e complicada, possuindo elevado índice de reprovação e alta curva de aprendizado. Ainda existe porém uma grande lacuna, principalmente no Brasil, na pesquisa e validação de dados dessa área da *Computer Science Education (CSE).*

1.2.2. QUESTÕES:

- Quais os principais fundamentos que devem ser trabalhados na aprendizagem de programação?
- Quais as soluções apontadas pela literatura?

1.2.3. PALAVRAS-CHAVE E SINÔNIMOS

- *learn to program, learning to program;*
- *computer science education, cse;*
- *programming learning;*

2. SELEÇÃO DAS FONTES

2.1. CRITÉRIOS DE SELEÇÃO DAS FONTES

- Disponibilidade para consulta na Internet;
- Disponibilidade para consulta nos portais de periódicos da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES);
- Presente nas pesquisas utilizando as palavras-chave;
- Presente nas referências bibliográficas ou de mesmo autor de fontes aproveitadas;
- Recomendada pelo orientador;

2.2. IDIOMAS DAS FONTES

- Inglês;
- Português;

2.3. IDENTIFICAÇÃO DAS FONTES

2.3.1. BASE DE DADOS IEEE E ACM

2.3.1.1. STRING DE BUSCA, UTILIZADA NA OPÇÃO “ADVANCED SEARCH” (IEEE)

- ("Document Title":”programming learning”)

2.3.1.2. STRING DE BUSCA, UTILIZADA NA OPÇÃO “ADVANCED SEARCH” (ACM)

- acmdlTitle:("programming learning")

3. SELEÇÃO DOS DOCUMENTOS

3.1. DEFINIÇÃO DOS DOCUMENTOS

3.1.1. CRITÉRIOS DE INCLUSÃO

- O documento possui menção a palavra-chave *programming learning* no título;
- O conteúdo do documento possui correlação com os fatores da aprendizagem de programação;
- O *abstract* do documento menciona a aprendizagem de programação;
- Documento disponível na internet;

3.1.2. CRITÉRIOS DE EXCLUSÃO

- O documento está incompleto;
- O público-alvo do conteúdo não são os estudantes de graduação;
- O contexto abordado não é o da fase inicial da aprendizagem e programação;

3.2. PROCEDIMENTO PARA A SELEÇÃO DOS DOCUMENTOS

As *strings* de busca devem ser utilizadas nas fontes de pesquisa escolhidas. Com o resultado da pesquisa em mãos, devem ser lidos os títulos, *abstracts* e palavras-chave, a fim de aplicar os critérios de inclusão e exclusão definidos na Seção 3.1 deste Protocolo. Os artigos que não se encaixarem em nenhum critério de exclusão e possuir os critérios de inclusão deverão ser lidos.

4. EXTRAÇÃO DE INFORMAÇÃO E SINTETIZAÇÃO DOS DADOS

Os documentos serão compreendidos, avaliados de acordo com seu grau de relevância e alinhamento junto às perguntas de pesquisa e sintetizados e, por fim, terem suas idéias condensadas.

APÊNDICE II – RESULTADOS DA REVISÃO SISTEMÁTICA

3.1.1.1 IDENTIFICAÇÃO DOS ESTUDOS

Com a execução das *strings* presentes no protocolo de pesquisa, foram retornados 37 trabalhos, sendo 28 da base de dados *IEEEXplore* e 9 da *ACM Digital Library.* Além desses foram selecionados trabalhos manualmente, provenientes de recomendações do orientador ou citados nas referências bibliográficas de algum dos artigos analisados.

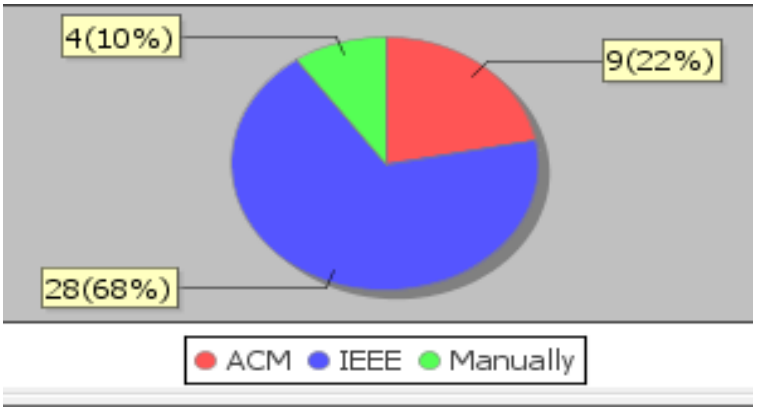

Figura 33 - Trabalhos coletados por fonte

3.1.2 SELEÇÃO DOS ESTUDOS

A fase de Seleção teve como objetivo excluir os documentos que não possuem relação com aprendizagem de programação a partir do título e dos *abstracts*, e identificar duplicação nas publicações encontradas. Nessa fase 11 artigos foram rejeitados com base nos critérios de exclusão e 30 foram aceitos.

3.1.3 EXTRAÇÃO DOS DADOS

Já na parte de Extração, os trabalhos que foram aceitos na Seleção são lidos na íntegra, para que sejam selecionados apenas aqueles que possuem todos os critérios de inclusão definidos no Protocolo de Revisão Sistemática, presente no Apêndice (I) deste documento. Dos trinta artigos que foram selecionados, apenas um foi rejeitado durante a fase de extração.

Quadro 14 - Fatores da aprendizagem de programação por artigo

Artigo	Fator Abordado
A class record and reviewing system	Motivação

designed to promote programming learning	A Atividade de Programar
A Collaborative Interactive Cyber-learning Platform for Anywhere Anytime Java Programming Learning	Motivação
A context for programming learning based on research communities	Motivação A Atividade de Programar
Algorithm animator: a tool for programming learning	Motivação
A model on adaptive example recommendation for programming learning	Motivação
A multinational case study on using diverse feedback types applied to introductory programming learning	Motivação
An adaptive model customized for programming learning in e-learning	Motivação A Atividade de Programar
An extension of fill-in-the-blank problem function in Java programming learning assistant system	Motivação
Applying Unidirectional versus Reciprocal Teaching Strategies in Web-Based Environment and Their Effects on Computer Programming Learning	A Atividade de Programar
A programming learning system for beginners-a completion strategy approach	Motivação A Atividade de Programar
Assistance in computer programming learning using educational data mining and learning analytics	Motivação
A taxonomy of exercises to support individual learning paths in initial programming learning	Motivação A Atividade de Programar
A tool for programming learning with pedagogical patterns	A Atividade de Programar
Contextualization of programming learning: A virtual environment study	Motivação A Atividade de Programar
Cooperation in programming learning	Motivação
Development and application of a web-based programming learning system with LED display kits	A Atividade de Programar
Encouraging Programming Learning for Novices with Grouping and Convincing Opinions	Motivação
Evaluation of a collaborative instructional framework for programming learning	Motivação
Increasing student commitment in introductory programming learning	Motivação
Learning to program – difficulties and solutions	A Atividade de Programar
Live Programming Learning Objects on Cloud	A Atividade de Programar
Making teaching of programming learning-	Motivação

oriented and learner-directed	
Merits and pitfalls of programming learning objects: a pilot study	A Atividade de Programar
On the Difficulty of Learning to Program	Motivação A Atividade de Programar
Studies and proposals about initial programming learning	Motivação A Atividade de Programar
The Analysis of a Case Study for Group Programming Learning	Motivação
The effect of student attributes on success in programming	A Atividade de Programar
The Study of Self-Assessment with Prompts, Learning Journal and Referencing through Sharing for Regulation of Cognition and Their Effect on Web-Based Programming Learning	Motivação A Atividade de Programar
Why is it so hard to learn to program?	Motivação A Atividade de Programar

APÊNDICE III – INSTRUMENTO PRELIMINAR

Fundamentos da Aprendizagem de Programação

1) **Quando foi seu ingresso na FGA (semestre/ano)?**
2) **Qual engenharia você pretende cursar?**
    a. Engenharia de Software;
    b. Engenharia Automotiva;
    c. Engenharia Aeroespacial;
    d. Engenharia de Energia;
    e. Engenharia Eletrônica;
    f. Ainda não decidi, mas não pretendo cursar software;
    g. Ainda não decidi, mas software é uma opção;
3) **As aulas de Computação Básica (CB) te influenciaram na escolha de qual engenharia seguir? De que maneira?**

4) **Em qual tentativa você se encontra em relação a disciplina Computação Básica (CB), incluindo as vezes que você se matriculou mas trancou a disciplina?**
    a. Primeira tentativa;
    b. Segunda tentativa;
    c. Terceira tentativa;
    d. Outra. Qual?
5) **Antes da matéria Computação Básica (CB), você já havia tido algum contato com programação?**
    a. Computação Básica foi meu primeiro contato com programação;
    b. Eu já possuía noções básicas de programação e algoritmos;
    c. Eu já havia feito um curso de programação;

6) **Em relação ao seu desempenho na disciplina Cálculo 1 (C1):**

   a. Fui aprovado na primeira tentativa e com menção final MS ou SS;

   b. Fui aprovado na primeira tentativa e com menção final MM;

   c. Fui aprovado na segunda ou terceira tentativas;

   d. Ainda não fui aprovado em Cálculo 1, mas já me matriculei na disciplina ao menos uma vez;

   e. Ainda não fui aprovado em Cálculo 1, e não me matriculei na disciplina nenhuma vez;

7) **Qual era sua expectativa em relação a disciplina Computação Básica antes do início do semestre?**

   a. Ser aprovado com desempenho acima da média (MS ou SS) sem precisar dedicar muita atenção e tempo a disciplina;

   b. Ser aprovado com desempenho acima da média (MS ou SS) após dedicar muita atenção e tempo a disciplina;

   c. Ser aprovado (MM) sem precisar dedicar muita atenção e tempo a disciplina;

   d. Ser aprovado (MM) após dedicar muita atenção e tempo a disciplina;

   e. Não obter aprovação independentemente do esforço dedicado a disciplina;

8) **E com o semestre já em andamento, qual a sua expectativa em relação a disciplina?**

   a. Ser aprovado com desempenho acima da média (MS ou SS), apesar de não estar dedicando muita atenção e tempo a disciplina.

   b. Ser aprovado com desempenho acima da média (MS ou SS), pois estou dedicando muita atenção e tempo a disciplina;

   c. Ser aprovado (MM), apesar de não estar dedicando muita atenção e tempo a disciplina;

   d. Não obter aprovação na disciplina;

9) **Você tem interesse por outros temas de computação (jogos, redes sociais, computadores, smartphones, etc) que não sejam relacionados a programação?**

   a. Sim, computação de maneira geral desperta meu interesse;

   b. Não, computação de maneira geral não desperta meu interesse;

10) **Você acha que saber programar é algo essencial na sua formação como engenheiro?**

a. Acho que saber programar é fundamental, e vai fazer toda a diferença na minha formação;

b. Acho que saber programar pode fazer diferença na minha formação, dependendo da área de atuação escolhida por mim;

c. Acho que saber programar não é essencial na minha formação;

11) Qual desses momentos você considera o mais difícil no processo de se desenvolver uma solução computacional para um problema:

a. Compreender e interpretar o problema;

b. Transformar o problema em um algoritmo;

c. Transformar o algoritmo em código;

d. Todos os anteriores;

12) Qual sua principal ferramenta de estudo além das aulas ministradas pelo professor?

a. Sites de aprendizado de programação (codecademy.com, koding.com, etc);

b. Slides ou anotações;

c. Livros;

d. Apenas assisto as aulas;

e. Outra. Qual?

13) Qual dessas técnicas você mais utiliza na hora de estudar:

a. Crio programas baseados no que foi ensinado em aula;

b. Estudo e leio códigos de programas vistos em sala ou similares;

c. Estudo a sintaxe da linguagem;

d. Todas as anteriores;

e. Outra. Qual?

14) Você acha que o professor e o monitor da disciplina são suficientes para sanar todas as dúvidas que surgem em decorrência da matéria?

a. Sim. Tanto no horário da aula quanto fora de sala;

b. Apenas no horário da aula;

c. Apenas fora do horário da aula;

d. O professor e o monitor não conseguem atender a todos os alunos;

e. Tiro minhas dúvidas com colegas de classe;

f. Tiro minhas dúvidas usando a internet;

15) Você se sente motivado a cursar essa disciplina?

a. Sim;

b. Não;

16) Justifique sua resposta na questão anterior.

APÊNDICE IV – RESULTADOS DO INSTRUMENTO PRELIMINAR

QUESTÃO 1

Apesar de a disciplina Computação Básica pertencer ao segundo semestre do fluxo, ela não possui pré-requisitos, permitindo que o aluno solicite a matéria a partir do seu primeiro semestre na faculdade. Na Universidade de Brasília os alunos possuem a opção de montar sua própria grade, independente do fluxo recomendando, desde que o aluno preencha os pré-requisitos das matérias que deseja cursar, permitindo que alunos em qualquer momento do curso se matriculem em Computação Básica. Alunos que estão cursando a matéria mas ingressaram na faculdade a mais de dois semestres possuem grandes chances de já estarem na segunda ou terceira tentativas em relação a matéria.

Quadro 15 - Respostas da questão 1 do instrumento preliminar

RESPOSTA	QUANTIDADE	PORCENTAGEM
2014/1	2	2,4%
2013/2	38	45,8%
2013/1	9	10,8%
2012/2	12	14,4%
2012/1	5	6,0%
2011/2	6	7,2%
2011/1	3	3,6%
2010/2	4	4,8%
2010/1	0	0%
2009/2	2	2,4%

2009/1	1	1,2%
2008/2	1	1,2%

QUESTÃO 2

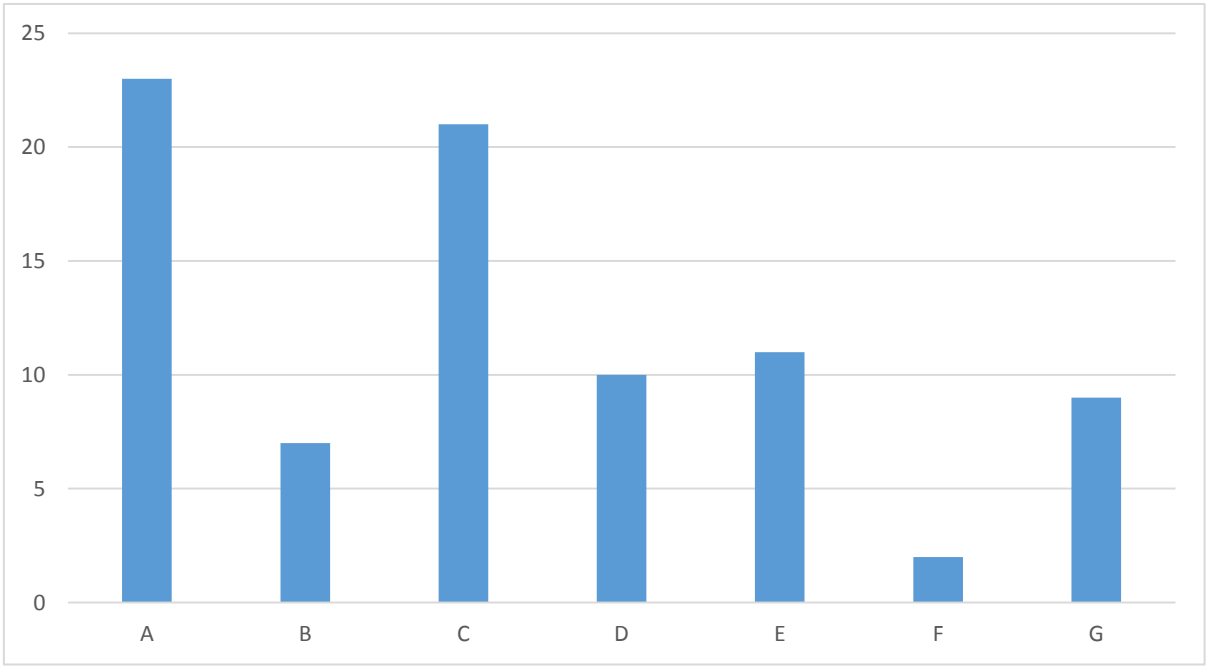

Figura 34 - Distribuição das respostas da questão 2 do instrumento preliminar

Como dito anteriormente, os alunos optam pela especialização que desejam seguir a partir do terceiro semestre. É interessante notar que a maioria dos resultados apontam que na verdade os alunos que pretendem se especializar em Engenharia de Software representam uma minoria quando comparados com os alunos que pretendem seguir outras especializações. A tabela abaixo compara os valores daqueles que vão ou podem optar por Engenharia de Software com os resultados de quem exluiu Engenharia de Software totalmente:

Quadro 16 - Porcentagem das respostas da engenharia de software x outras engenharias

ENGENHARIA DE SOFTWARE	OUTRAS RESPOSTAS
32 (38,5%)	51 (61,5%)

Esses dados levantam uma pergunta: ao planejar as aulas e conteúdos acadêmicos será que o professor leva em consideração que diferentemente da

Engenharia de Software habilidades em programar não é uma dos principais requisitos dos outros cursos?

QUESTÃO 3

O objetivo dessa questão era tentar compreender se e de que forma as aulas de programação influenciam na decisão do aluno em qual especialização seguir. Os resultados contabilizados são os apresentados na tabela:

Quadro 17 - Respostas da questão 3 do instrumento preliminar

RESPOSTA	QUANTIDADE
Influenciou	51 (61%)
Não influenciou	32 (39%)

Dos que afirmaram terem sido influenciados por Computação Básica, houve alta incidência de argumentações a respeito da influência negativa que a disciplina teve sobre estes indivíduos, fazendo-os não optar por software. Já dos que afirmam não terem sido influenciados, a maior incidência de argumentos é no sentido destes indivíduos já terem feito sua opção de especialização antes de cursar a disciplina.

QUESTÃO 4

O objetivo da quarta questão era fazer um levantamento da média das tentativas em que os alunos se encontram em relação a computação básica.

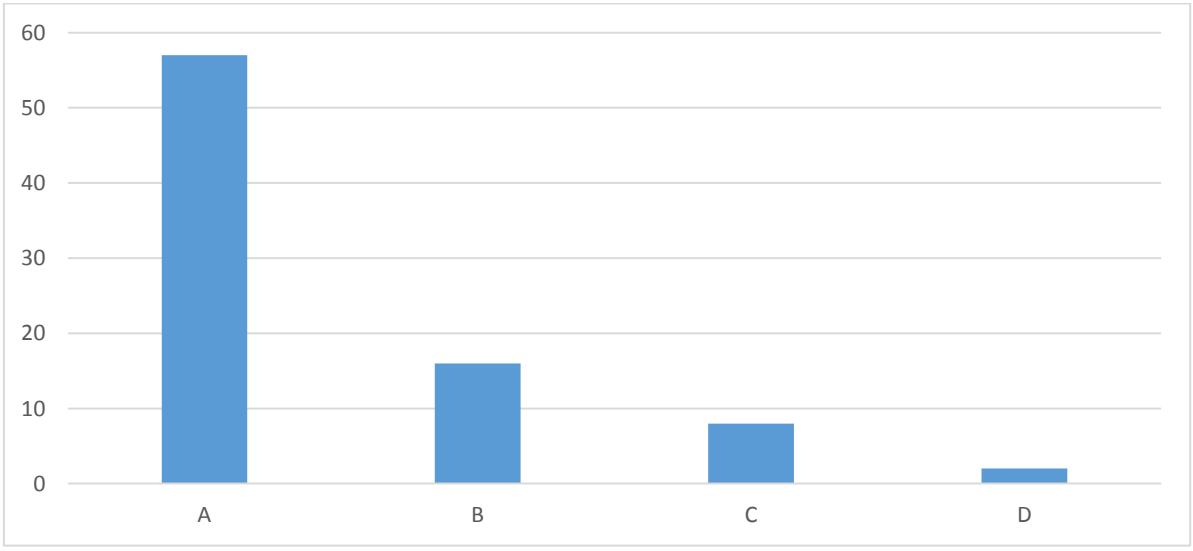

Figura 35 – Distribuição das respostas da questão 4 do instrumento preliminar

O resultado mostra que apesar da diversidade de semestres em que se encontram os alunos a grande maioria está cursando a disciplina Computação Básica pela primeira vez. Esse resultado porém não é confiável já que antes a disciplina atendia por outro nome, não impedindo que alunos que já tenham cursado Introdução a Ciência da Computação estejam fazendo Computação Básica pela primeira vez.

QUESTÃO 5

O objetivo dessa questão era tentar contabilizar se uma parcela significativa dos alunos matriculados em Computação Básica possui alguma experiência prévia com programação. Os resultados estão apresentados no gráfico:

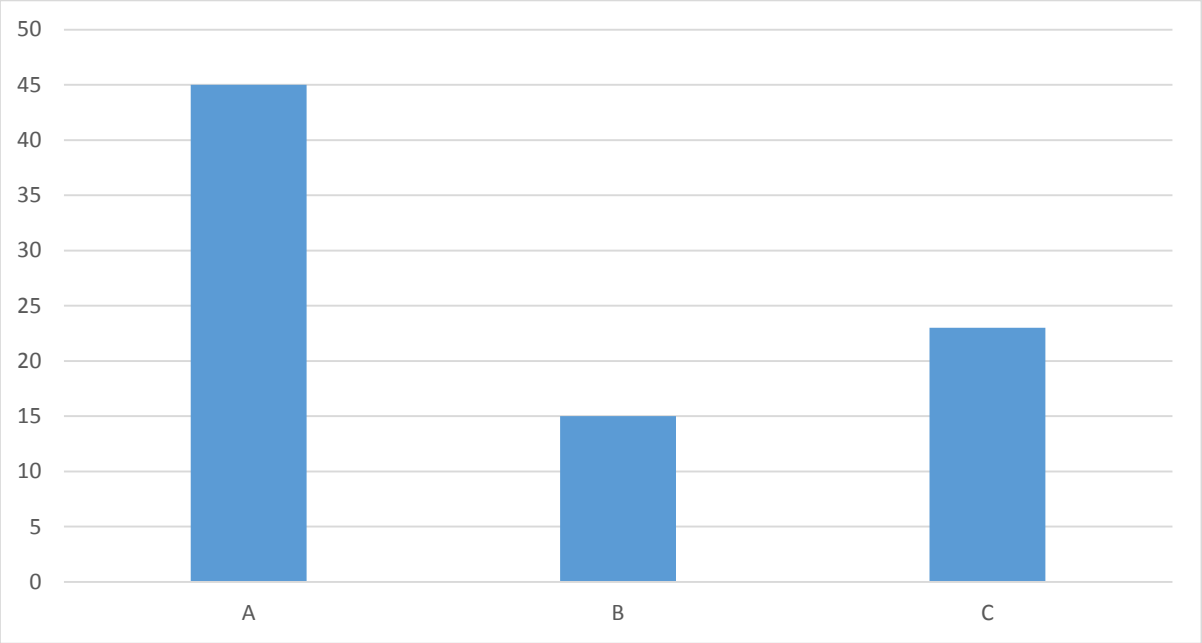

Figura 36 - Distribuição das respostas da questão 5 do instrumento preliminar

O resultado indica que menos da metade dos alunos possuíam algum tipo de conhecimento ou contato prévio com programação, porém essa parcela se mostra sim significativa e contribui para heterogeneidade das turmas.

QUESTÃO 6

O objetivo dessa questão era tentar relacionar as habilidades matemáticas do aluno com ausência ou presença de dificuldade em programação. Porém da maneira como o questionário foi planejado não possibilitou essa rastreabilidade. O gráfico demonstra a distribuição das respostas:

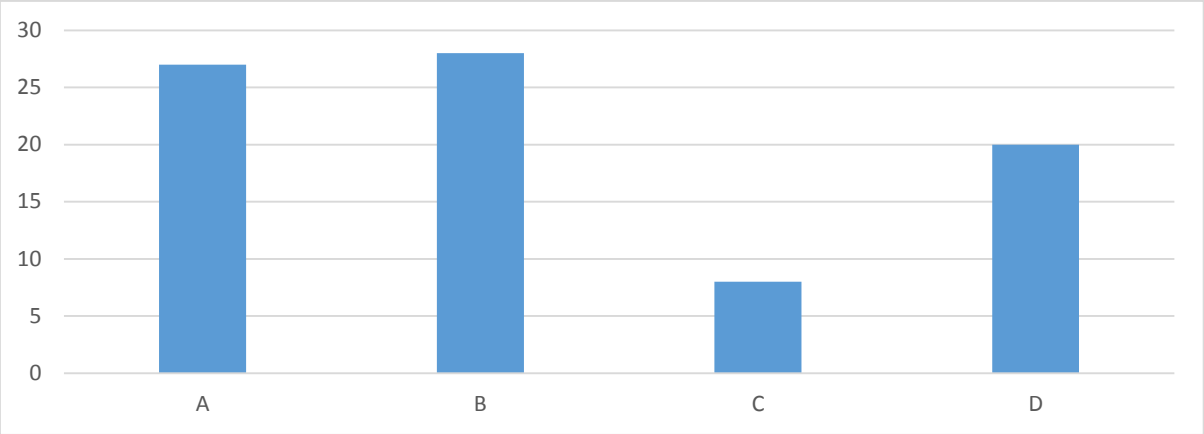

Figura 37 - Distribuição das respostas da questão 6 do instrumento preliminar

QUESTÃO 7

O objetivo da sétima questão era a de compreender como são as expectativas dos alunos antes no início da disciplina Computação Básica, fazendo também uma comparação com os dados levantados que se referem as turmas anteriores. A distribuição dos resultados se encontra na tabela a baixo:

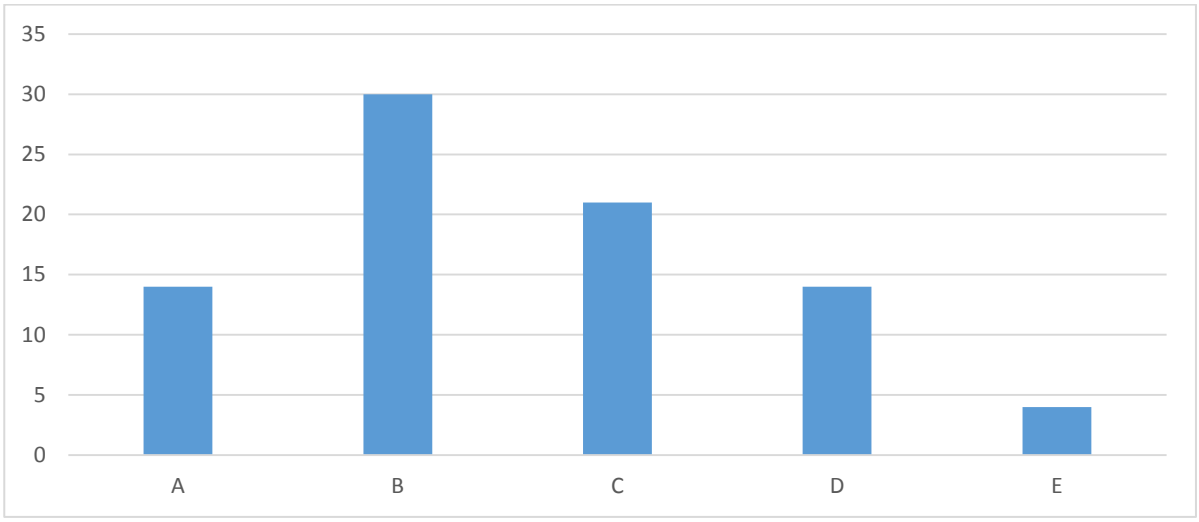

Figura 38 - Distribuição das respostas da questão 7 do instrumento preliminar

È interessante notar que a expectativa da grande maioria dos alunos é a de obter a aprovação, sendo que a maioria desses possuía uma expectativa de se sair acima da média.

QUESTÃO 8

A oitava questão foi elaborada com o intuito de tentar compreender se existe mudança significativa nas expectativas dos alunos, e comparar essa nova expectativa com os resultados coletados referentes aos semestres anteriores. A distribuição dos resultados está representada no gráfico:

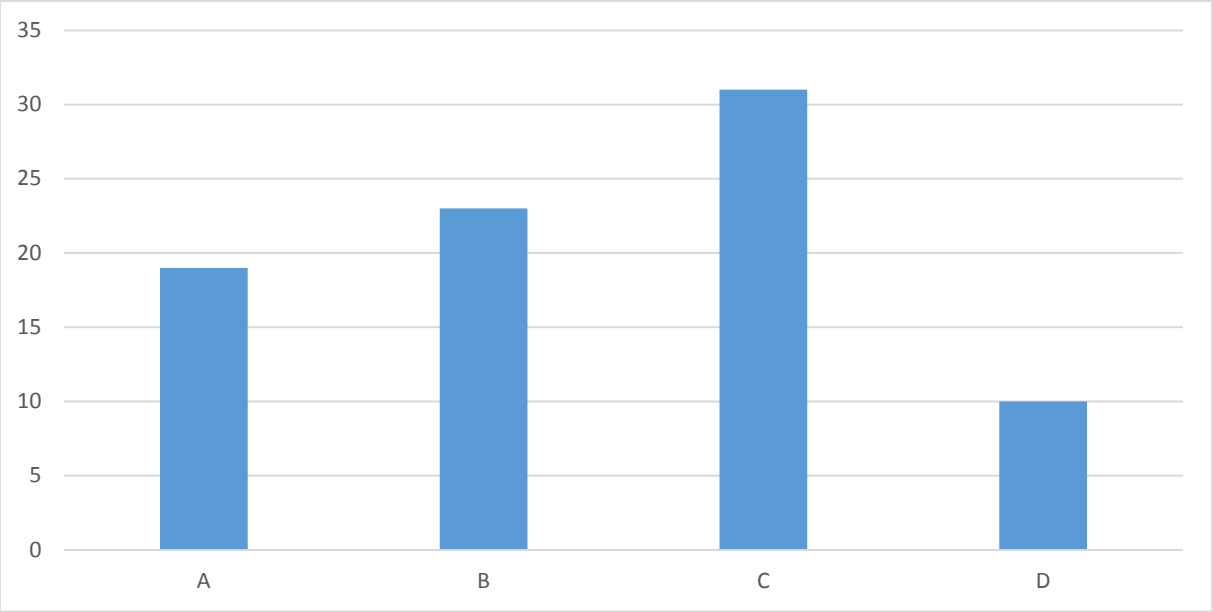

Figura 39 - Distribuição das respostas da questão 8 do instrumento preliminar

Os resultados mostram que a expectativa dos alunos sofre sim uma baixa, mas ainda está longe das médias dos resultados dos semestres anteriores.

QUESTÃO 9

A intenção dessa pergunta é compreender se computação de forma geral é um assunto de interesse dos estudantes. Segue o gráfico com os resultados:

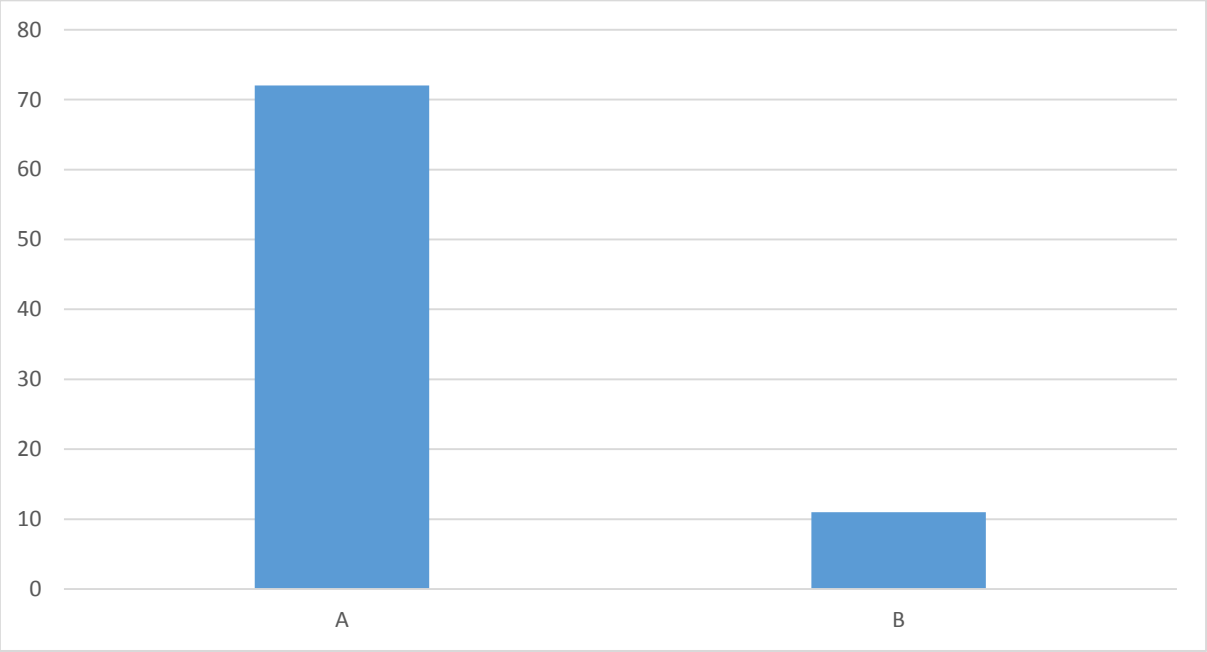

Figura 40 - Distribuição das respostas da questão 9 do instrumento preliminar

Esse resultado está em acordo com o apresentado pela literatura, que descreve que a maioria dos alunos tem sim algum tipo de interesse em Computação

de maneira geral, mas que normalmente não é estabelecida uma conexão entre programação e esses assuntos.

QUESTÃO 10

Como um grande número dos alunos matriculados em Computação Básica devem seguir diferentes especializações, a questão dez tem como objetivo verificar se independente da especialização seguida os alunos possuem ou não a percepção da importância dessa disciplina no curso e na carreira. Os resultados obtidos são os apresentados no gráfico:

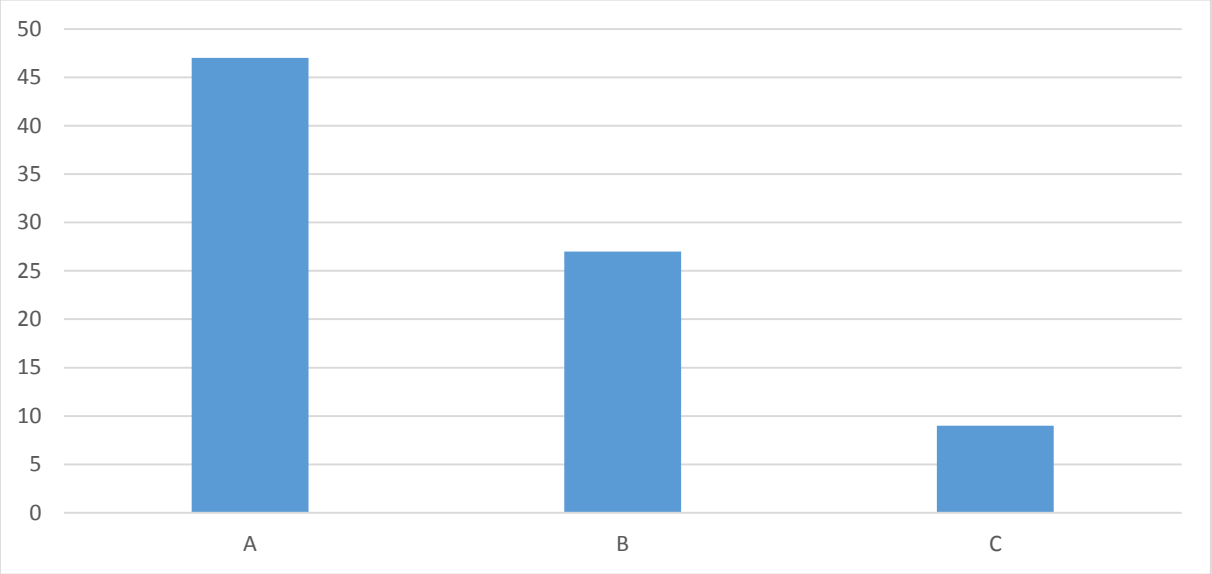

Figura 41 - Distribuição das respostas da questão 10 do instrumento preliminar

Esses resultados demonstram que a grande maioria dos alunos compreende a importância que o conhecimento em programação pode desempenhar no seu futuro.

QUESTÃO 11

A ideia por trás dessa questão é tentar compreender que momentos do processo de desenvolvimento de uma solução computacional apresenta mais dificuldade aos alunos. Os resultados foram:

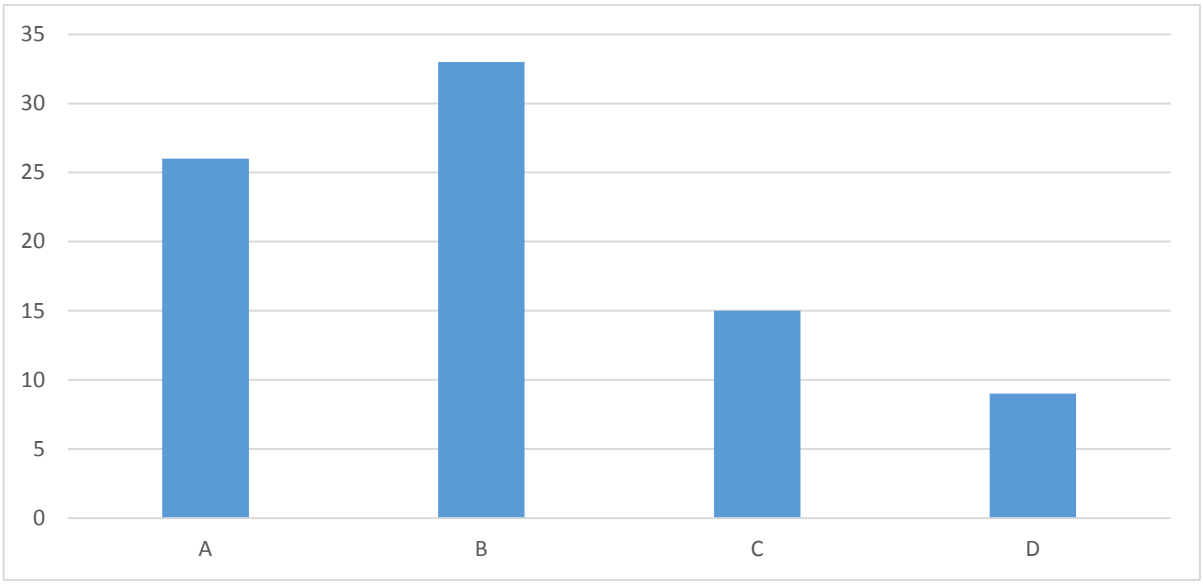

Figura 42 - Distribuição das respostas da questão 11 do instrumento preliminar

Esse resultado é inconclusivo pois não ouve disparidade suficiente entre as respostas.

QUESTÃO 12

O objetivo dessa questão era tentar compreender qual a principal ferramenta utilizada pelos alunos na hora de estudar programação. Os resultados foram os seguintes:

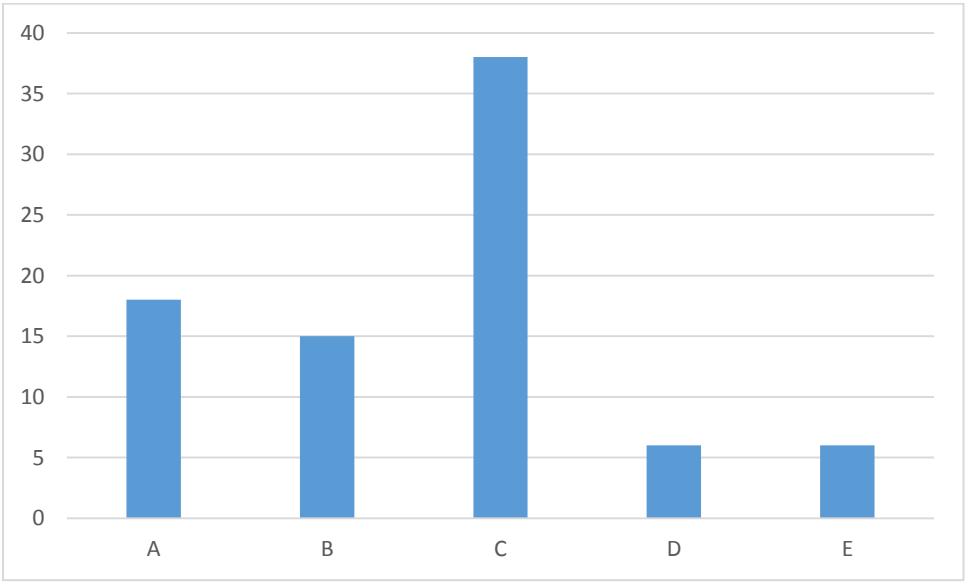

Figura 43 - Distribuição das respostas da questão 12 do instrumento preliminar

QUESTÃO 13

O objetivo dessa questão era tentar compreender qual a principal ferramenta utilizada pelos alunos na hora de estudar programação. Os resultados foram os seguintes:

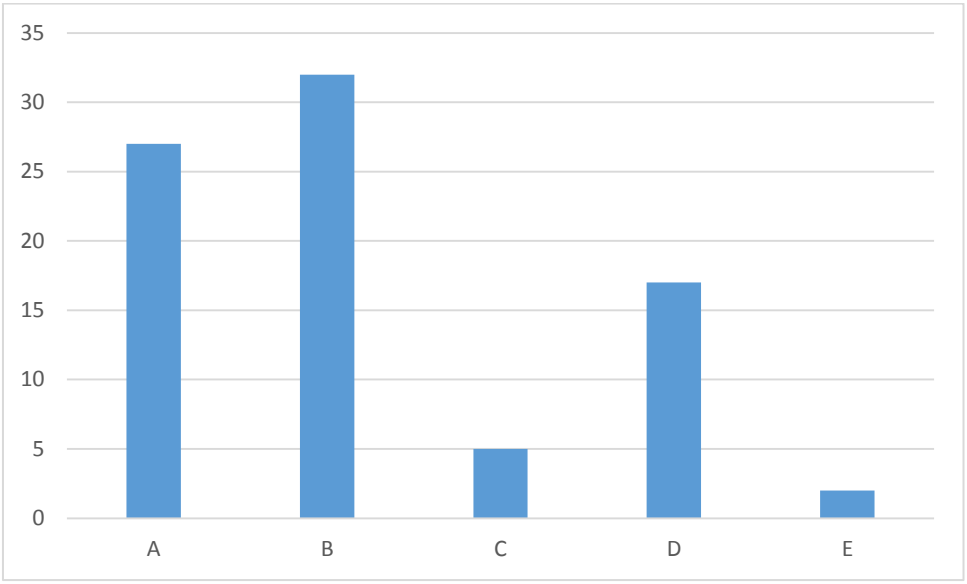

Figura 44 - Distribuição das respostas da questão 13 do instrumento preliminar

QUESTÃO 14

A necessidade de supervisão é um dos aspectos mais mencionados na literatura. O objetivo dessa pergunta era depreender se os alunos estão tendo a supervisão adequada. O gráfico da distribuição das respostas é o seguinte:

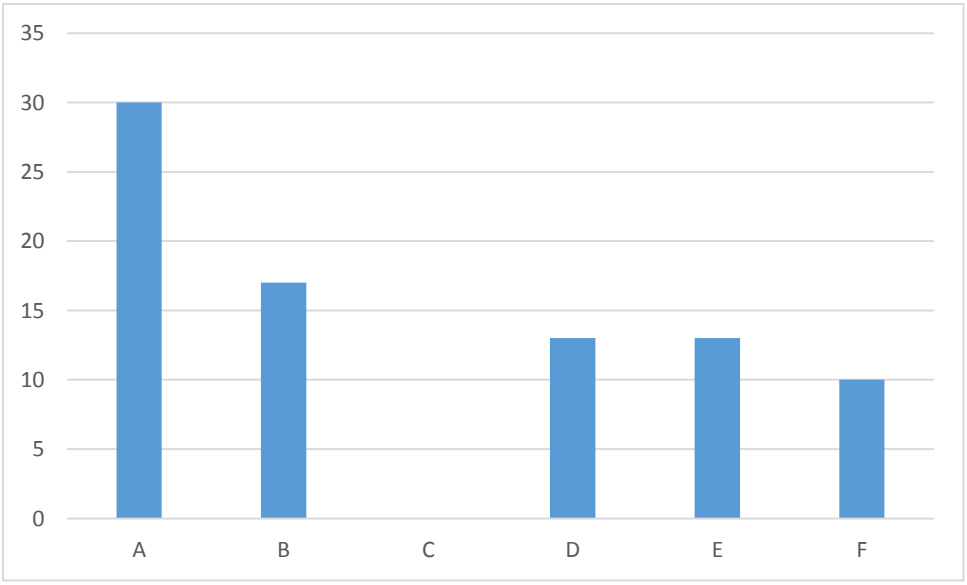

Figura 45 - Distribuição das respostas da questão 14 do instrumento preliminar

Esse resultado demonstra que nem todas as turmas possuem pessoas suficientes para supervisionar de maneira adequada os alunos, isso pode impactar

negativamente nos resultados finais dos alunos das turmas sem monitores ou com professores com pouca disponibilidade.

QUESTÃO 15

De acordo com as respostas nessa questão seria possível determinar se de maneira geral os alunos se sentem motivados. O resultado é o apresentado abaixo:

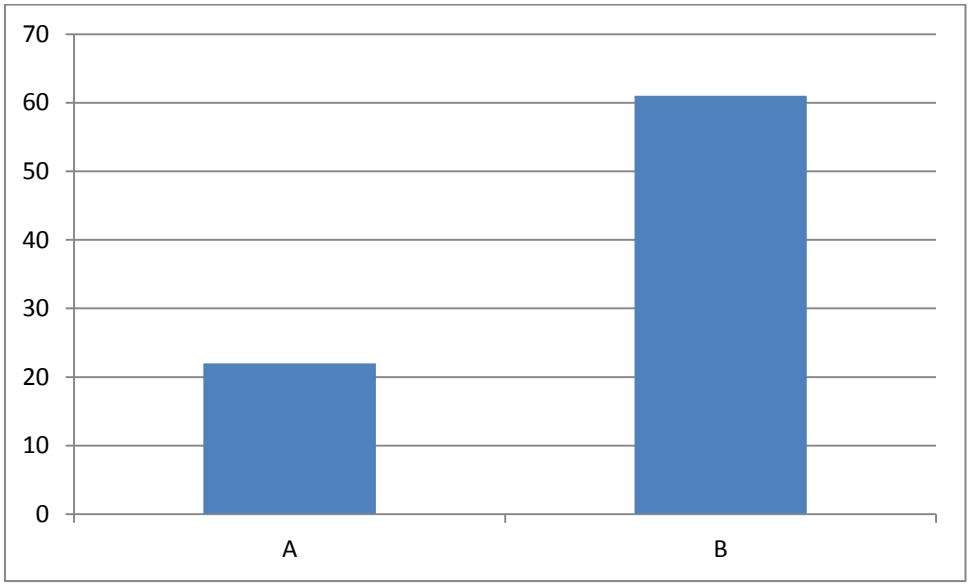

Figura 46 - Distribuição das respostas da questão 15 do instrumento preliminar

Esse resultado mostra que os índices de falta de motivação são altos, apresentando alta disparidade quando comparado com o número de alunos que se sentem motivados. O próximo passo seria tentar verificar se são justamente os alunos desmotivados que acabam falhando na matéria.

QUESTÃO 16

As respostas além de muito diversificadas, normalmente apresentavam linhas de raciocínio não fundamentado, fazendo que não fosse possível estabelecer afirmações a respeito da origem da desmotivação dos alunos. O único fator em que foi observado algum sinal de repetição é o de alunos que citaram como desmotivador o fato de suas turmas não possuírem monitor da matéria.

APÊNDICE V – SURVEY

Questionário de percepção sobre o ensino-aprendizagem de programação

Prezado Aluno

Você está sendo convidado a responder um questionário sobre o ensino-aprendizagem de programação, que faz parte do TCC do aluno Guilherme Calixto, sob orientação do professor Dr. Wander Cleber M. Pereira da Silva.

O questionário apresentado está dividido em duas partes, uma parte que contempla itens relacionados à ... parte relacionada .... Você deverá atribuir uma nota de um a cinco para indicar o grau de ocorrência das afirmativas apresentadas. Como mostrado na escala abaixo:

1.Concordo totalmente 2. Concordo parcialmente 3. Indiferente 4. Discordo parcialmente 5. Discordo totalmente

←----------→

Quanto mais próximo de **um (1)**, maior o grau de concordância.

Quanto mais próximo de **cinco (5)**, maior o grau de discordância.

Lembre-se que:

- O questionário deverá ser respondido individualmente;
- Não existem respostas “certas” ou “erradas”. O importante é mostrar de forma sincera como você percebe, durante seu dia-a-dia no trabalho, cada uma das afirmações apresentadas;
- Sua participação é muito importante para o sucesso deste trabalho;
- Não há identificação no questionário, portanto, será preservado o anonimato do respondente.

Desde já agradecemos à atenção dispensada e nos colocamos à disposição para os esclarecimentos necessários.

Atenciosamente,

Wander C. M. Pereira da Silva

81128473 (wander.pereira@unb.br)

Guilherme Calixto

84079907

(guilhermelcalixto@gmail.com)

Questionário sobre aprendizagem de programação

Classifique as questões abaixo conforme a escala indicada: 1 – Concordo totalmente 2 – Concordo parcialmente 3 - Indiferente 4 – Discordo parcialmente 5 – Discordo totalmente	
Sobre a aprendizagem de programação é possível afirmar que:	**Avaliação** **1 2 3 4 5**
1. Gostar de temas relacionados à computação (jogos, redes sociais,*tablets*, *smartphones*, etc) faz aumentar a motivação para aprender a programar.	☐ ☐ ☐ ☐ ☐
2. Aprender a programar é uma tarefa complexa e difícil.	☐ ☐ ☐ ☐ ☐
3. Aprender a programar é muito importante para a formação do engenheiro.	☐ ☐ ☐ ☐ ☐
4. Cursar as disciplinas de Álgebra e Cálculo facilita a aprender a programar.	☐ ☐ ☐ ☐ ☐
5. Estudar em grupo é uma prática que facilita aprender a programar.	☐ ☐ ☐ ☐ ☐
6. A forma de ensinar do professor é fundamental para aprender a programar.	☐ ☐ ☐ ☐ ☐
7. A prática de exercícios facilita para aprender a programar.	☐ ☐ ☐ ☐ ☐
8. O *feedback* imediato nos exercícios e provas facilita para aprender a programar.	☐ ☐ ☐ ☐ ☐
9. É possível aprender a programar apenas assistindo as aulas.	☐ ☐ ☐ ☐ ☐
10. Compreender e interpretar um problema é a etapa mais complexa e difícil no processo de desenvolvimento de um programa.	☐ ☐ ☐ ☐ ☐
11. O tempo de estudo dedicado a aprender programação deve ser superior a de outros conteúdos.	☐ ☐ ☐ ☐ ☐

12. Uma infra-estrutura adequada (computadores, softwares, internet, biblioteca) facilita aprender a programar.	☐ ☐ ☐ ☐ ☐
13. Aprender a programar influencia na definição de qual engenharia seguir.	☐ ☐ ☐ ☐ ☐
14. Os monitores são importantes para aprender a programar.	☐ ☐ ☐ ☐ ☐
15. O tipo de linguagem não tem nenhuma influência sobre aprender a programar.	☐ ☐ ☐ ☐ ☐

Identificação
Gênero: 1☐ Masculino 2☐ Feminino 3☐ Outro.
Idade: 1☐ 20 anos ou menos 2☐ 21 a 25 anos 3☐ 26 a 30 anos 4☐ 31 a 35 anos 5☐ Acima de 36 anos
Você está na instituição há quanto tempo: 1 ☐ 01 semestre 2☐ 02 semestres 3☐ 03 semestres 4☐ 04 semestres 5☐ 05 semestres ou mais.
Qual engenharia você pretende cursar? 1☐ Aeroespacial 2☐ Automotiva 3 ☐ Energia 4☐ Eletrônica 5☐ Software
Qual a forma de ingresso: 1☐ Cotista 2☐ Não cotista
Em qual escola/instituição você concluiu o ensino médio? _____
Em qual região você mora? _____
Domina (lê, fala, escreve) alguma língua estrangeira?_____ **Qual?**_____
Computação Básica foi seu primeiro contato com programação? ☐ Sim ☐ Não

Quais outros fatores você considera que facilitam ou dificultam a aprendizagem de programação? Registre a seguir: