

Uma Proposta de um Método para a Personalização do Ensino de Engenharia de Software

Andressa B. Ferreira

Instituto Federal de Educação,
Ciência e Tecnologia do Ceará
Crato, Brasil
andressa.ferreira@ifce.edu.br

Antônio R. M. de Oliveira

Instituto Federal de Educação,
Ciência e Tecnologia do Ceará
Crato, Brasil
renatomoura557@yahoo.com

Wiliana V. Lima

Instituto Federal de Educação,
Ciência e Tecnologia do Ceará
Crato, Brasil
wilianavl@hotmail.com

ABSTRACT

Developing a software is a task which involves different stages (e.g. analyse, project, implementation and test). Taking this in consideration, the Software Engineering disciplines must be committed with training students to many different roles. Based on this context, this paper shows a method proposal for roles attributions in academic projects of software development. This method makes use of Belbin Team Roles, proposed by the occupational psychology, with the purpose of mapping the students traits in the responsibilities associated with software development. For an evaluation process, one group of 48 students were submitted to this method. Partial results shows that this method can help the teaching process since the very first classes.

Keywords

Education, method, engineering, software

Classificação ACM

D.2.m. Software Engineering: Miscellaneous; K.3.m. Computers and Education: Miscellaneous.

INTRODUÇÃO

Atualmente, é possível encontrar diferentes cursos da área de tecnologia sendo ofertados nas mais diversas instituições do país. Ao analisar a grade curricular dos cursos dessa área, nos principais centros de ensino¹ (e.g., USP, UFPE, UFMG, UFCG), é possível observar um determinado comportamento comum: os estudantes são apresentados inicialmente, e durante a maior parte do curso, a disciplinas que fornecem conhecimento e apoio necessários para a implementação de software.

Apesar da importância dessas disciplinas para a formação do estudante, é possível encontrar alunos concluintes, sentindo-se inaptos a desenvolver outros trabalhos que não os de programação. No entanto, o desenvolvimento de software envolve não apenas atividades de implementação ou codificação, mas também planejamento, design, teste e evolução [1,2,6,7].

Usualmente, essas outras expertises (e.g. análise, gerência, design, teste) começam a ser trabalhadas em disciplinas de Engenharia de Software (ES), cujo objetivo é capacitar os estudantes a desenvolver software de modo profissional [2]. Geralmente, nas disciplinas de ES, por meio da participação em projetos de desenvolvimento de software, os estudantes são incentivados a conhecer e aprender sobre as diferentes atividades que representam as fases do ciclo de vida de um software.

Diante disso, este trabalho tem como objetivo propor um método para atribuição de papéis e responsabilidades aos alunos formadores de equipes de projetos em disciplinas de Engenharia de Software.

O método, por sua vez, tem como objetivo proporcionar ao professor, subsídios capazes de fazê-lo melhor entender o perfil de cada estudante e, trabalhar com cada um deles, competências complementares às suas habilidades. Com isso, espera-se que estudantes de cursos de tecnologia sintam-se e percebam-se aptos a desenvolver atividades que não apenas de desenvolvimento de código.

Para tanto, este trabalho faz uso da Teoria dos Papéis e do Team Role Self-Perception Inventory (TRSPI), ambos propostos por Raymond M. Belbin, na área de psicologia ocupacional [4,5]. O TRSPI é uma ferramenta de análise que, segundo [4,5], serve para classificar indivíduos identificando como eles podem se comportar e melhor contribuir para o progresso de um time.

No método proposto neste trabalho, o TRSPI é utilizado como ferramenta para identificar o perfil dos estudantes. Os resultados da identificação servem como entrada para um mapeamento entre as características dos indivíduos definidos por [4] e as principais atividades a serem desempenhadas ao longo da disciplina. O objetivo do mapeamento é dar ao professor a possibilidade de melhor entender os seus estudantes e capacitá-los de modo mais efetivo para diferentes funções associadas ao desenvolvimento de software.

As demais seções deste trabalho são organizadas de forma a apresentar: 1) a motivação para esta pesquisa, 2) o referencial teórico necessário, 3) a proposta do método, 4) detalhes da aplicação do método em duas turmas distintas

¹ De acordo com o Guia do Estudante 2017, as instituições mencionadas possuem avaliação 5 estrelas em seus cursos da área de tecnologia e, por isso, são consideradas neste trabalho como principais centros.

de engenharia de software e, por fim, 5) as considerações finais.

MOTIVAÇÃO

Como motivação para esta pesquisa, estão presentes: experiências de alunos, opiniões de professores e os conceitos e fundamentos identificados na literatura.

Por meio do uso de questionários disponibilizados via internet, alunos de cursos de graduação em Ciência da Computação, Sistemas de Informação, Engenharia de Software e Engenharia da Computação, foram estimulados a relatar suas experiências com as diferentes fases do desenvolvimento de software em projetos acadêmicos. Ao todo, 142 alunos de diferentes instituições de ensino, participaram do levantamento.

Segundo os estudantes, suas experiências acadêmicas com o desenvolvimento de software concentram-se, principalmente, a partir do quarto semestre do curso. A partir desse ponto (quarto semestre), a maioria dos estudantes (85%) acredita ter bons conhecimentos em programação e afirmam ingressar em disciplinas de engenharia de software com a intenção de entender melhor outras partes do processo de desenvolvimento, que não apenas atividades de codificação.

No entanto, a maior parte dos alunos (85%), afirma que a disciplina aborda as diferentes fases do processo de modo teórico e, quando estimulados a irem para a prática (e.g. por meio de projetos acadêmicos de desenvolvimento), os alunos não se sentem seguros ou motivados para se engajarem com atividades que já não o fizeram antes.

Também por meio do uso de questionários *web*, professores de Engenharia de Software nos cursos de graduação em Ciência da Computação, Sistemas de Informação, Engenharia de Software e Engenharia da Computação, foram convidados a relatar suas experiências com a realização de atividades de desenvolvimento de software em suas disciplinas. Ao todo, 18 professores contribuíram com esta pesquisa.

Segundo os professores, os projetos acadêmicos realizados em suas disciplinas contemplam desde os primeiros momentos a divisão de papéis entre os estudantes membros das equipes. Essa divisão, segundo os professores, parte dos próprios alunos formadores dos times e, segundo eles, são feitas com base em preferências pessoais ou experiências acadêmicas anteriores. Ainda segundo o grupo de professores entrevistados, não é considerada fácil nem para o professor nem para o aluno realizar a tarefa de fazer essas atribuições de papéis (e.g. Quem será o gerente do projeto? Quem serão os analistas do projeto?).

Após analisar as opiniões de professores e alunos da área, é possível perceber o interesse de ambos por ferramentas ou métodos capazes de auxiliar o processo de definição de

papéis em equipes acadêmicas de desenvolvimento de software.

Além do já exposto nesta seção, os conceitos e fundamentos presentes na literatura que servem como motivação para este trabalho, são apresentadas na Seção Referencial Teórico.

REFERENCIAL TEÓRICO

O desenvolvimento de software é uma atividade de crescente importância na sociedade contemporânea [1,2,6,7]. Visando melhorar a qualidade do software e aumentar a produtividade no processo de desenvolvimento, surgiu a Engenharia de Software (ES).

Segundo [2], Engenharia de Software é a disciplina de engenharia cujo foco está em todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até sua manutenção, quando o sistema já está sendo usado. Essa disciplina inclui ainda o gerenciamento do projeto de software e o desenvolvimento de ferramentas, métodos e teorias para auxiliar no desenvolvimento do software.

Para [1], a ES é uma disciplina que reúne metodologias, métodos e ferramentas a ser utilizados, desde a compreensão do problema até o momento em que o software desenvolvido deixa de ser operacional, visando resolver problemas do processo de desenvolvimento e do produto de software.

A IEEE, por sua vez, define a ES como a aplicação de uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, na operação e na manutenção de software [7].

Segundo [7], para que um projeto de software seja bem-sucedido, é necessário que alguns parâmetros sejam corretamente analisados, como o escopo do software, os riscos envolvidos, os recursos necessários, tarefas a serem realizadas, custos aplicados, cronograma e pessoal, envolvendo, este último, a definição da equipe de trabalho.

De acordo com [6], os papéis fundamentais em uma equipe de trabalho de desenvolvimento de software são:

1. Gerente, aquele responsável por supervisionar o trabalho dos demais membros da equipe e acompanhar quão bem o desenvolvimento do software está progredindo.
2. Analista, aquele responsável pela compreensão do problema relacionado ao sistema que se deve desenvolver, ou seja, pelo levantamento dos requisitos e sua efetiva modelagem. O analista deve, portanto, descobrir o que o cliente necessita.
3. Designer ou Projetista, aquele que deve levar em conta as especificações do analista e propor a melhor tecnologia para produzir um sistema executável para elas. Deve,

então, apresentar uma solução para as necessidades do cliente.

4. Desenvolvedor, aquele que têm a responsabilidade de compreender os requisitos especificados por analistas e projetistas e transformá-los em um produto executável.

5. Testador, aquele que é responsável por garantir a qualidade e eficiência do sistema que está sendo desenvolvido. Para isso, o testador verifica e valida o produto de software.

Conforme mencionado na seção de Introdução, a tarefa de atribuir papéis em um projeto acadêmico de desenvolvimento de software não é algo fácil para alunos ou professores. Mediante esse contexto, apresenta-se na seção a seguir um método para auxiliar essa atribuição.

O MÉTODO

De acordo com [3], o ensino é um ato socialmente localizado. Uma vez que o professor não ensina no vazio, em situações hipoteticamente semelhantes, mas sim com alunos reais e em situações definidas. O papel do professor vai, portanto, além de ensinar de modo mecânico ou descontextualizado.

Mediante esse contexto, o método apresentado neste trabalho é composto por três fases (Figura 1), que possibilitam ao professor entender o comportamento de seus alunos e direcioná-los melhor no processo de ensino e aprendizagem em disciplinas de Engenharia de Software.



Figura 1: As fases do método

Como é possível observar na Figura 1, a primeira fase do método consiste na aplicação do TRSPI. Como mencionado na Seção 1, o TRSPI é um teste capaz de identificar como pessoas tendem a se comportar e se relacionar com outros. Como desenvolvimento de software é uma tarefa essencialmente realizada por equipes, o TRSPI se torna adequado, pois o mesmo permite identificar como pessoas podem contribuir em atividades quando trabalhando em times. Maiores informações sobre o TRSPI são apresentadas na subseção referente a explanação da Fase 1.

Ainda de acordo com a Figura 1, a segunda fase do método consiste em realizar o mapeamento do resultado de cada aluno no TRSPI e as atividades inerentes a disciplina de Engenharia de Software. As atividades consideradas neste trabalho são apresentadas na subseção referente a Fase 1 e o mapeamento na subseção referente a Fase 2.

Por fim, a terceira e última fase consiste em definir e planejar estratégias de ensino com base no mapeamento

realizado na fase 2. Sugestões de como fazer isso são apresentadas na subseção referente a Fase 3 do método.

Fase 1: Aplicação do TRSPI

O TRSPI (*Team Role Self-Perception Inventory*) é um questionário de auto percepção, desenvolvido pelo pesquisador Raymond Meredith Belbin.

Belbin é um pesquisador britânico e teórico de gestão, conhecido por seu trabalho em equipes de gerenciamento.

O TRSPI é um teste que contém sete seções com oito itens cada, sendo que as seções expõem situações hipotéticas envolvendo trabalho em equipe e os itens descrevem como o indivíduo se comportaria.

O intuito do teste é captar e mostrar a tendência do respondente em se comportar de acordo com as características do perfil. Quanto mais alto a pontuação em um dos perfis, maior a tendência do respondente em assumir este perfil.

Desta forma, pode-se fazer uso de características comportamentais de um membro da equipe para maximizar seu potencial perante atividades recorrentes a sua função dentro da equipe.

Para a aplicação do TRSPI o professor necessita ter em mãos uma cópia do teste para cada aluno. O teste pode ser encontrado em: <https://www.belbin.com/>. É aconselhável traduzir o teste para o idioma utilizado em sala de aula, uma vez que o mesmo se encontra disponível apenas em inglês.

Uma vez aplicado o teste, as respostas dadas pelos alunos serão mapeadas dos papéis propostos por Belbin para papéis de projetos de software de acordo com as atividades e habilidades fundamentais em uma equipe de projeto de software.

Atividades relacionadas a Engenharia de Software

Para a definição de quais atividades seriam consideradas na proposta do método, e quais as habilidades necessárias para a realização de cada uma delas, foram avaliadas 31 ementas de disciplinas de engenharia de software em 31 diferentes cursos do país.

Após a análise de cada ementa, foram filtrados os conteúdos programáticos comuns a maioria delas. Além das ementas, para alguns cursos, foi possível encontrar e avaliar os planos de ensino utilizados pelos professores. Nesses casos, informações extraídas dos planos (e.g., objetivos definidos, atividades a serem realizadas, bibliografia recomendada, metodologia de avaliação) também foram consideradas.

Além do conteúdo programático e atividades relacionadas, estão presentes na Tabela 1 as habilidades necessárias ao

desenvolvimento de cada atividade. A definição das habilidades apresentadas na tabela foi feita com base nos planos de ensino, considerando disciplinas e assuntos que são pré-requisitos para outros, bem como os objetivos traçados pelos autores das bibliografias utilizadas em sala.

Tabela 1: Conteúdo, atividades e habilidades

Conteúdo Programático	Atividades Relacionadas	Habilidades e Competências Associadas
Modelos de Processo	1) Escolher processo a ser utilizado 2) Identificar atividades de um processo 3) Definir processo 4) Comparar processos	1) Análise de decisões
Gerência de Projetos	1) Gerenciar pessoas 2) Liderar pessoas 3) Definir papéis 4) Atribuir responsabilidades 5) Gerenciar tempo 6) Fazer cronograma 7) Gerenciar riscos 8) Calcular estimativas 9) Definir preço 10) Fazer plano de projeto	1) Análise de decisões 2) Habilidades de apresentação, comunicação e negociação 3) Liderança e desenvolvimento do espírito de equipe 4) Gerenciamento de tempo 5) Capacidade de decisão em situações de stress 6) Bom relacionamento interpessoal 7) Objetividade na definição e avaliação do trabalho
Engenharia de Requisitos	1) Diferenciar requisitos de usuários e requisitos de sistema 2) Diferenciar requisitos funcionais e não funcionais 3) Elicitar requisitos 4) Analisar requisitos 5) Validar requisitos 6) Organizar requisitos em um documento	1) Facilidade de expressão e comunicação 2) Facilidade de adaptação a mudanças 3) Iniciativa na solução de problemas e desenvolvimento de alternativas criativas 4) Presteza e Iniciativa 5) Organização 6) Proatividade 7) Objetividade
Projeto Arquitetural	1) Utilizar métodos gráficos para representar sistemas de software 2) Tomar decisões a respeito da arquitetura de um sistema 3) Organizar arquitetura de sistemas	1) Facilidade de expressão e 2) Facilidade de adaptação a mudanças 3) Iniciativa na solução de problemas e desenvolvimento de alternativas criativas 4) Organização 5) Objetividade
Desenvolvimento e Implementação	1) Desenvolvimento de código 2) Padrões de projeto	1) Atenção a detalhes e boa memória 2) Capacidade de concentração 3) Capacidade de resolver problemas práticos 4) Disciplina 5) Raciocínio lógico e matemático
Teste de Software	1) Testes individuais 2) Testes de sistema 3) Testes de componentes 4) Testes de release 5) Teste de aceitação 6) configuração e execução de testes 6) Criação de testes e casos de teste 7) Documento de testes	1) Conhecimento das abordagens e técnicas de testes 2) Capacidade para diagnosticar problemas 3) Habilidade de programação 4) Habilidade em depuração e diagnósticos

O conteúdo apresentado na Tabela 1 é utilizado para o mapeamento apresentado na Fase 2.

Fase 2: O Mapeamento - Papéis de Belbin versus Habilidades e Competências necessárias ao Desenvolvimento de Software

Segundo [5], dentro de um contexto de trabalho em equipe todo indivíduo pode ser classificado de acordo com os seus conhecimentos e a sua função técnica. Para tanto, o autor construiu um conjunto de oito papéis de time, que descrevem padrões que caracterizam o comportamento de um indivíduo em relação aos outros na facilitação do progresso de um time.

Em [4], alguns nomes foram mudados e o papel “Especialista” foi acrescentado, totalizando nove papéis na versão atual do modelo, porém este papel não é muito utilizado na literatura e, por isso, não será tratado neste trabalho. Dito isto, os papéis apresentados por [5] e considerados neste trabalho são:

1. Implementador (*Implementer* - IMP): Gosta de fazer as coisas e se envolver em tarefas práticas. Têm o senso comum prático e muito autocontrole e disciplina. De modo geral, o implementador é tipicamente uma pessoa cuja lealdade e interesse estão na empresa. São eficientes e têm bom senso do que é praticável e relevante. São características do implementador: ser controlado, disciplinado, eficiente, metódico, estável e sistemático. Entre os seus pontos fortes estão: Transforma ideias em ações práticas, confiável, conservador e eficiente. Entre os seus pontos fracos encontra-se: tendência a se sobrecarregar com trabalho.

Com base nas características definidas por Belbin e as habilidades apresentadas na Tabela 1, o perfil implementador pode ser mapeado para os papéis de analista ou desenvolvedor.

2. Coordenador (*Co-ordinator* - CO): Tem a habilidade de fazer com que as pessoas trabalhem em direção a um mesmo objetivo. Maduros, confiantes e confiáveis, delegam prontamente. Em relações interpessoais, são rápidos para encontrar talentos individuais e usá-los para alcançar os objetivos do grupo. Tem uma visão abrangente das coisas e do mundo, o que geralmente incita respeito. São características do coordenador: Dominante, confia nos demais, positivo, tem autocontrole, possui disciplina, é estável. Entre os seus pontos fortes estão: confiante, bom diretor, esclarece objetivos, bom em tomada de decisão, delega bem. Entre os seus pontos fracos encontra-se: Pode ser visto como manipulador.

Com base nas características definidas por Belbin e as habilidades apresentadas na Tabela 1, o perfil coordenador pode ser mapeado para o papel de gerente de projetos.

3. Realizador ou formatador (*Shaper* - SH): são altamente motivados, com muita energia e necessidade de realização. São hábeis em vencer obstáculos. Muito sólidos e assertivos, mostram uma forte resposta emocional a qualquer forma de desapontamento ou frustração. São pensadores solitários, frequentemente argumentativos.

Com base nas características definidas por Belbin e as habilidades apresentadas na Tabela 1, o perfil realizador pode ser mapeado para o papel de designer ou projetista.

4. Criativo (*Planter* - PL): É a pessoa das ideias. São inovadores e criativos. Preferem certo distanciamento dos outros membros da equipe, usam a imaginação e frequentemente trabalham de modo pouco ortodoxo. Um de seus defeitos é que, geralmente, não sabem distinguir boas ideias de ideias ruins.

Com base nas características definidas por Belbin e as habilidades apresentadas na Tabela 1, o perfil criativo pode ser mapeado para o papel de analista ou designer ou projetista.

5. Investigador (*Resource Investigator* - RI): Gostam de novidades e estão prontos para assumir desafios. São entusiastas. Hábeis em encontrar e trazer recursos para a equipe, em saber o que está disponível e o que pode ser feito, é normalmente bem recebido por sua postura agradável. Entretanto, a menos que seja motivado, seu entusiasmo pode desvanecer rapidamente.

Com base nas características definidas por Belbin e as habilidades apresentadas na Tabela 1, o perfil investigador pode ser mapeado para o papel de analista ou designer ou projetista.

6. Avaliador (*Monitor Evaluator* - ME): São bons em analisar, avaliar ideias e sugestões. São lentos para tomar decisões, pois preferem pensar sobre as coisas. Geralmente têm habilidade crítica elevada e capacidade para examinar todos os aspectos de uma situação. Como ponto fraco, avaliadores não costumam demonstrar interesse em inspirar outras pessoas.

Com base nas características definidas por Belbin e as habilidades apresentadas na Tabela 1, o perfil avaliador pode ser mapeado para o papel de analista ou designer ou projetista.

7. Flexível (*Teamworker* - TW): É a pessoa que se oferece para trabalhar pelo grupo. São os membros que mais dão suporte a um grupo. Flexíveis, se adaptam bem a situações e pessoas diferentes. A diplomacia e a habilidade de percepção do perfil Flexível se torna um ativo real, especialmente em um regime em que os conflitos estão

sujeitos a acontecer ou ser artificialmente suprimidos. As pessoas cooperam mais quando eles estão à sua volta.

Com base nas características definidas por Belbin e as habilidades apresentadas na Tabela 1, o perfil flexível pode ser mapeado para o papel de analista ou gerente.

8. Finalizador (*Completer Finisher* - CF): São muito detalhistas e não gostam de começar algo que não possam concluir. Tipicamente introvertidos, requerem pouco estímulo ou incentivo externo. Podem ser intolerantes com aqueles que possuem uma disposição mais relaxada. Não gostam de delegar, preferindo fazer sozinhos o trabalho.

Com base nas características definidas por Belbin e as habilidades apresentadas na Tabela 1, o perfil finalizador pode ser mapeado para o papel de desenvolvedor ou testador.

Fase 3: A Definição de Estratégias de Ensino

Com o resultado do mapeamento de cada estudante em mãos, o professor será capaz de identificar para quais atividades da disciplina cada um deles tem maior ou menor aptidão.

Ao optar por uma metodologia de avaliação em grupo (e.g., projetos de desenvolvimento de software), o método poderá fornecer ao professor subsídios de como agrupar alunos. Como exemplo, não atribuir o papel de gerente a um aluno com o perfil de finalizador. Não atribuir na mesma equipe dois ou mais alunos com o perfil coordenador (e.g. seria o mesmo que ter em uma única equipe dois gerentes).

Ao optar por trabalhar com uma metodologia de avaliação individual (e.g., provas e exercícios), o professor poderá fazer uso do método para saber em quais atividades terá que dar mais suporte a cada aluno. Como exemplo, identificar a necessidade de trabalhar e aplicar técnicas para melhorar o entusiasmo de alunos com o perfil do tipo investigador.

Diante desse contexto, é possível afirmar que com o uso das informações obtidas por meio da aplicação do método, o professor pode previamente identificar dificuldades que os alunos poderão ter ao longo da disciplina. Espera-se que com isso, o docente trabalhe, de modo personalizado, as necessidades e habilidades de cada aluno, capacitando-o e estimulando-o.

AValiação DO MÉTODo

Como forma de avaliação, o método proposto neste trabalho foi aplicado em duas turmas de Engenharia de Software do curso de Sistemas de Informação do IFCE Campus Crato. A primeira turma, composta por 22 alunos, participou da aplicação do método no semestre 2017.2, a segunda turma, composta por 26 alunos, participou no semestre 2018.1.

O grupo total de alunos participantes da avaliação é composto exclusivamente por estudantes do sétimo semestre. Em sua maioria do sexo masculino (32 homens e 16 mulheres), os participantes possuem faixa etária entre 20 e 27 anos e já cursaram todas as disciplinas referentes ao intervalo do primeiro ao sexto semestre do curso.

A ementa da disciplina de Engenharia de Software, bem como as ementas das demais disciplinas já cursadas pelos estudantes participantes podem ser encontradas em: https://ifce.edu.br/crato/campus_crato/cursos/superiores/bac_harelados/sistemas/grade-curricular.

Com a aplicação do método foi possível identificar que:

1) 80% dos alunos acreditam ser muito boa a aplicação do método em disciplinas de engenharia de software (Figura 2).

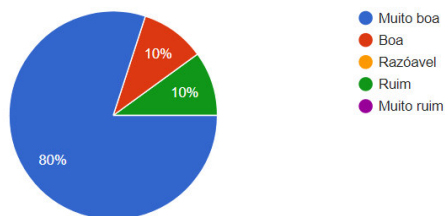


Figura 2: Respostas para a questão "Como você classificaria a aplicação e uso do método em disciplinas de ES?"

2) 90% dos alunos consideram que seu aprendizado na disciplina onde se fez uso do método foi Bom ou Muito Bom (Figura 3).

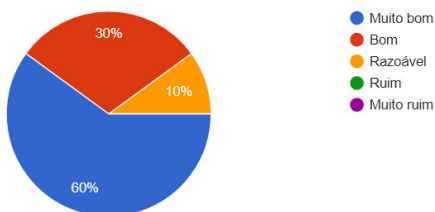


Figura 3: Respostas para a questão "Como você classificaria o seu aprendizado na disciplina de ES com o uso do método?"

3) 80% dos alunos concordam que ter os subsídios do método para definir qual papel deve assumir no projeto é melhor do que definir baseado apenas em experiências anteriores ou gosto pessoal (Figura 4).

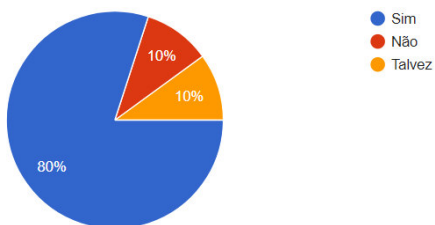


Figura 4: Respostas para a questão "Você considera que ter o auxílio do método na definição do seu papel em uma equipe de desenvolvimento de software é algo positivo?"

Com base nos resultados apresentado acima, considera-se positiva a aplicação do método em sala de aula. Principalmente sob o ponto de vista de um dos principais interessados no processo de aprendizagem: o aluno.

CONSIDERAÇÕES FINAIS

Este trabalho apresentou uma proposta de um método para a personalização do ensino em disciplinas de Engenharia de Software. O método proposto busca incentivar professores de ES a captar e utilizar desde os momentos iniciais do curso características dos seus estudantes que podem ser aperfeiçoadas, desenvolvidas e estimuladas por meio da disciplina.

Como resultados da aplicação do método em duas turmas de ES observou-se que sua utilização trás subsídios para os professores melhor trabalharem conteúdos em suas disciplinas, de modo personalizado. Foi possível constatar também a aceitação e entusiasmo por parte dos alunos com a abordagem proposta.

No entanto, mesmo diante do positivo progresso, constatou-se as seguintes oportunidades de trabalhos futuros: 1) Automatizar a aplicação do método em salas de aula por meio da construção de uma ferramenta de software capaz de servir como meio para os alunos responderem o TRSPI. Além disso, a ferramenta irá prover para o professor o resultado da aplicação para cada aluno e sugerir estratégias de ensino direcionadas para cada estudante. 2) Acrescentar, após a aplicação do TRSPI e mapeamento, um segundo teste sobre conhecimentos técnicos na área do perfil identificado para o estudante. Com isso, o professor terá em suas mãos informações sobre o nível de conhecimento de cada estudante na área onde tem mais aptidão de acordo com o TRSPI e o mapeamento.

REFERÊNCIAS

1. Ariadne R. Carvalho; Thelma S. Chiossi. 2001. Introdução à engenharia de software. Campinas: Ed. da Unicamp.
2. Ian Sommerville. 2011. Engenharia de Software. 9. ed. São Paulo: Pearson Education.
3. Maria I. da Cunha. 1992. O bom professor e sua prática. 2 ed. São Paulo: Papirus., 182p.
4. Raymond M. Belbin. 1981. Management Teams: Why they succeed or Fail. Oxford: Butterworth Heinemann.
5. Raymond M. Belbin. 1993. Team Roles at Work. Oxford: Butterworth Heinemann.
6. Raul S. Wazlawick. 2013. Engenharia de Software: conceitos e práticas. São Paulo: Elsevier.
7. Roger S. Pressman. 2011. Engenharia de Software: uma abordagem profissional. 7. ed. Porto Alegre: AHGH.