Ensino de Engenharia de Software utilizando Sala de Aula Invertida

Plínio R. S .Vilela[1]

[1]Faculdade de Tecnologia – Universidade Estadual de Campinas (Unicamp)
R. Paschoal Marmo, 1888 – 13.484-332 - Limeira – SP – Brasil

prvilela@unicamp.br

Abstract. *Teaching Software Engineering in undergraduate courses has never been an easy task. In general, the concepts presented are too abstract for the student who hasn't had practical experience in the job market, specially with the development of professional software. Often such abstract content is presented in a excessively expository manner, with little or no practical application, which distances the student from the real world and ends up demotivating them throughout the semester. This work presents the experience of applying an active methodology in teaching software engineering to an undergraduate course at a brazilian university.*

Resumo. *Ensinar Engenharia de Software em disciplinas de graduação nunca foi tarefa fácil. No geral, os conceitos apresentados são muito abstratos para o aluno que ainda não tem uma experiência prática no mercado de trabalho, mais especificamente com o desenvolvimento de software profissional. Muitas vezes esse conteúdo muito abstrato é apresentado de uma maneira demasiadamente expositiva, com pouca ou nenhuma aplicação prática, o que distancia o estudante do mundo real e termina por desmotivá-lo ao longo do semestre. Este trabalho apresenta uma experiência prática de aplicação de uma metodologia ativa no ensino de engenharia de software em um curso de graduação de uma universidade brasileira.*

1. Introdução

Talvez um dos livros mais tradicionais sobre engenharia de software seja o do professor, pesquisador e consultor americano Roger Pressman. O livro se chama "Software Engineering: A practitioner's Approach" com a primeira edição publicada em 1982 [Pressman 1982]. Uma tradução livre do título seria: "Engenharia de Software: Uma Abordagem de Praticante"[1], no sentido de que não era um livro para acadêmicos e sim um livro que iria passar conceitos, ferramentas e técnicas que pudessem ser colocadas imediatamente em prática.

Grady Booch, co-autor da UML (*Unified Modeling Language*) [Booch et al. 2005], publicou um artigo sobre a história da engenharia de software em 2019 [Booch 2019] onde descreve a origem do termo "Engenharia de Software", basicamente relacionando com o crescimento acelerado da demanda por software no período pós segunda guerra. Ou seja, o termo foi cunhado e toda a disciplina

[1]Título da versão em Português: "Engenharia de Software: Uma Abordagem Profissional".

desenvolvida em resposta a uma demanda do mundo real, uma demanda prática. Portanto a natureza prática dessa disciplina é inegável. Pode-se dizer que Engenharia de Software diz respeito ao desenvolvimento de software no mundo real. Assim, não basta escrever o código corretamente, deve-se escrever o código que atenda às necessidades do cliente, implementando corretamente os requisitos elicitados, dentro das restrições de prazo e custo estabelecidas, gerando um código e documentação associada que possa ser mantido e modificado ao longo do tempo, à medida que o sistema evolui. A percepção de que a engenharia de software deve ser ensinada de uma maneira prática é reforçada pelo trabalho [Marques Samary et al. 2015] que avaliou 173 estudos publicados em congressos e revistas da área, o estudo é de 2015 mas já mostra claramente essa tendência.

Algumas perguntas são colocadas como desafios aos docentes que ministram disciplinas de engenharia de software em nível de graduação:

- Como ensinar essa disciplina para alunos que não têm vivência prática na área?
- Como falar de padrões de qualidade de código para um aluno que acabou de aprender a escrever os primeiros programas?
- Como abordar processos de desenvolvimento de software, metodologias ágeis, controle de versão, métricas, com alunos que basicamente desenvolvem código individualmente e depois de receber a nota esquecem daquele código?

Esse é o desafio que o autor vem tentando enfrentar desde 2018 através da aplicação da metodologia ativa de ensino conhecida como Sala de Aula Invertida [Bergmann and Sams 2012]. A metodologia vem sendo aplicada para o ensino de engenharia de software já há algum tempo. Veras, Rocha e Viana [Veras et al. 2020] apresentam um mapeamento sistemático sobre o assunto incluindo artigos publicados entre 2008 e 2020 num total de 26 estudos primários analisados. Outro estudo publicado por Marques, Quispe e Ochoa [Marques Samary et al. 2015] a partir da análise de 173 artigos destaca a importância dada para a experiência prática simulando o trabalho na indústria em disciplinas de engenharia de software, listando a sala de aula invertida como uma das metodologias indicadas para se incorporar a experiência prática no ensino dessas disciplinas.

Este artigo não tem a pretensão de apresentar o planejamento detalhado de uma disciplina de engenharia de software para cursos de graduação, até porque a ementa e o programa dessas disciplinas variam muito de curso para curso em diferentes Instituições de Ensino Superior, mas sim relatar a experiência do autor com a aplicação da metologia em disciplinas de graduação. Apresenta-se também um guia de boas práticas para que outros professores universitários possam adaptar e aplicar a metodologia.

2. Sala de Aula Invertida

Em uma sala de aula convencional dois momentos distintos são tradicionalmente percebidos, o primeiro momento de aula, um encontro síncrono normalmente presencial, onde o professor faz uma exposição de conteúdo para os alunos e um segundo momento pós-aula, assíncrono e tradicionalmente remoto, onde o aluno recebe a incumbência de aplicar o conteúdo ao qual foi exposto em atividades extra classe.

No modelo de sala de aula invertida essa ordem é trocada, ou seja, em um momento pré-aula o aluno é exposto ao conteúdo e durante a aula síncrona esse conteúdo

é aplicado em atividades práticas. Essa inversão já vem sendo experimentada por professores a muito tempo, mas a popularização da metodologia e o uso do termo ”sala de aula invertida” (*flipped classroom*) ocorreu com o trabalho de dois professores de química do Colorado, Estados Unidos, Jon Bergmann e Aaron Sams que começaram a gravar suas aulas e disponibilizar para os alunos em 2007 [Bergmann and Sams 2012]. Inicialmente o propósito era ter um apoio para os alunos que por ventura tinham faltado à aula ou para aqueles que quisessem um material extra de revisão. Eles perceberam que os vídeos permitiam que os alunos aprendessem o conteúdo básico em casa e usassem o tempo da sala de aula para aprofundar o entendimento e realizar atividades práticas. Essa descoberta levou-os a desenvolver e promover a ideia da sala de aula invertida [Bergmann and Sams 2012].

Algumas vantagens são percebidas com o uso dessa metodologia, como por exemplo a possibilidade de se acomodar os diferentes ritmos de aprendizagem ou mesmo estilos de aprendizagem dos diferentes alunos. Como o conteúdo é apresentado de maneira assíncrona cada aluno pode levar um tempo diferente para trabalhar sobre o conteúdo disponibilizado. Em uma sala de aula tradicional o professor deve seguir um ritmo que procure atender ao maior número de alunos possível, mas sempre pode ocorrer de alunos que precisam de um pouco mais de tempo ou repetição para absorver o conteúdo possam ficar defasados, enquanto aqueles alunos que já são mais rápidos podem ficar entediados e desmotivados. No entanto a maior vantagem destacada pelos praticantes dessa metodologia é o uso mais eficiente do tempo em sala de aula, como os alunos já estão previamente preparados com o conteúdo necessário o tempo de aula é usado em atividades que permitem a reflexão e a fixação do conteúdo. O que gera um aprendizado mais efetivo.

3. Trabalhos Relacionados

Gerald Gannod, Janet Burge e Michael Helmick da Miami University em Oxford Ohio [Gannod et al. 2008] reforçam a ideia de que a engenharia de software é uma disciplina prática e portanto educar a próxima geração de engenheiros de software requer que os estudantes sejam ativos no processo de ensino e aprendizagem. Os autores afirmam que em um curso de três horas semanais os estudantes devem assistir de três a seis horas de aula em vídeo sob demanda antes do encontro síncrono. Um tempo adicional fora de sala de aula é usado para completar a atividade em sala. O artigo motiva o uso da sala de aula invertida e dá sugestões de como alguns dos cursos do currículo modelo de engenharia de software de 2004 [on Computing Curricula 2004] poderiam incorporar o uso da sala de aula invertida. No caso deste artigo foram usados os guidelines fornecidos para a disciplina *SE201 - Introduction to Software Engineering*, sendo basicamente usar a preparação pré-aula para fazer uma cobertura completa do conteúdo, usando texto e vídeo-aulas, e usar o tempo de aula para um aprofundamento maior em alguns conteúdos que exigem uma aplicação prática, como o levantamento de requisitos ou aplicação de métricas de software. Só não foi aplicado o projeto extra-classe pois isso deixaria a carga horária total de dedicação do estudante muito alta.

Eun Choi da Dongguk University, Coréia do Sul [Choi 2013] seguiu uma abordagem científica para comparar a efetividade da sala de aula invertida com a metodologia de ensino tradicional. Ao final os resultados apresentados pelos alunos dos dois grupos foram comparados. Choi destaca também que a metodologia de ensino tradicional é boa para passar conceitos teóricos para os alunos, mas tem menos eficácia em preparar os

alunos para a aplicação prática desses conceitos. Além disso, também não é eficaz em treinar os alunos em outras habilidades interpessoais (*soft skills*) como: comunicação do trabalho em grupo, colaboração, organização, tomada de decisão, resolução de conflitos, liderança, socialização e concentração.

Tabela 1. Comparação entre as tarefas [Choi 2013]

	Tradicional	Invertida
Número de tarefas	baixo	alto
Cobertura	baixa	alta
Feedback	atrasado	imediato
Profundidade por tarefa	alta	baixa

Ainda segundo Choi [Choi 2013] existe uma grande diferença entre as atividades de aprendizagem do modelo tradicional e do invertido, como mostrado na Tabela 1. O número de tarefas quando se usa o modelo invertido pode ser muito maior do que no modelo tradicional. Como no modelo invertido se busca *feedback* [2] frequente e focado em cada uma das aulas, as tarefas tendem a ser menos complexas, porém mais frequentes do que no modelo tradicional. A cobertura diz respeito ao conjunto total de tópicos abordados na disciplina. As atividades da sala de aula invertida para Engenharia de Software foram planejadas e executadas segundo o que é apresentado na Tabela 2. O aprendizado pré-aula é feito através de vídeos sob demanda, cobrindo aspectos teóricos sobre análise, projeto, codificação, teste, qualidade, métricas e manutenção. As atividades em sala de aula cobrem aspectos práticos desses assuntos.

Tabela 2. Atividades Práticas - Sala de Aula Invertida [Choi 2013]

#	Pré-Aula	Aula
1		Introdução
2	VA01: Processos	At01: Montando o Time - MBTI - Estudo de caso para processo
3	VA02: Análise de Requisitos	At02: Conceitualização - Conceito do Projeto - Missão
4	VA03: Especificação de Requisitos	At03: Gamestorming - Brainstorming para lista de características
5	VA04: Modelagem Funcional	At04: Análise de Casos de Uso - Escrever casos de uso
6	VA05: Modelagem Estática - Identificando Classes - Diagrama de Classes	At05: Design Studio - Diag de Classes - Classes candidatas - Identificar relacionamentos - Desenhar diagrama de classes
7	VA06: Modelagem Dinâmica - Diagrama de Sequência - Diagrama de Comunicação	At06: Design Studio - Diag de Sequência
8	VA07: Princípios de Projeto - Abstração, Modularização	At07: Aplicando princípios de projeto
9	VA08: Projeto de Interface - Elementos de IU - Estudo de caso de IU	At08: Design Studio - UI/UX
10	VA09: Padrões de Projeto	At09: Aplicando padrões de projeto
11	VA10: Implementação - Estilo de código - Refatoração	At10: Escrevendo código padronizado
12	VA11: Teste (1) - Teste funcional (caixa preta)	At11: Escrevendo casos de teste para particionamento de equivalência
13	VA12: Teste (2) - Teste estrutural (caixa branca) - Teste de sistema	At12: Análise de caminhos de execução e escrever casos de teste

Ashish Sureka et al. [Sureka et al. 2013] da IIITD, Nova Deli - India, apresentam um estudo de caso sobre a aplicação da sala de aula invertida para o ensino de Engenharia de Software. Os autores focam na necessidade de se estimular o trabalho em grupo nesse

[2] Apesar de ser um anglicismo o termo *feedback* é bastante popular e compreendido no Português, por isso foi mantido neste texto. Apenas marcando-o em itálico.

tipo de disciplina e apresentam também um conjunto de questões a ser aplicada aos alunos ao final do semestre como forma de avaliar a percepção desses estudantes sobre o impacto e efetividade da metologia de ensino. Além da aplicação da sala de aula invertida os autores exploram o uso de uma metodologia de avaliação por pares, o uso de clientes reais nos projetos de desenvolvimento e a configuração da sala de aula em um formato de estúdio e não a tradicional organização para aulas expositivas.

Michael Herold et al. [Herold et al. 2012] da Ohio State University - Ohio/USA, também aplicam a sala de aula invertida em sua disciplina de engenharia de software. O material de estudo pré-aula era inicialmente os slides das aulas com a apresentação do professor gravada sobre ela, apenas a voz em uma sequencia única do inicio ao fim. Esse formato se mostrou tedioso para os alunos e foi trocado por seguimentos menores para cada parte da explicação, mas mantendo o formato de slides com voz. Uma contribuição interessante do trabalho foi a quantificação de todo o processo, registrando o tempo de estudo dedicado pelos alunos e os resultados obtidos nos questionários e outras fases do processo.

4. Aplicando Sala de Aula Invertida em Engenharia de Software

O autor vem aplicando a metodologia de sala de aula invertida desde 2018, em duas disciplinas da área de engenharia de software ministradas para dois cursos de graduação, Bacharelado em Sistemas de Informação (integral) e Tecnologia em Análise e Desenvolvimento de Sistemas (noturno). Apesar de serem cursos diferentes as disciplinas são oferecidas exatamente iguais para ambas as turmas. As disciplinas são: Engenharia de Software I com 2 horas de aula semanais e Engenharia de Software II com 4 horas de aulas semanais. Diferentemente do conteúdo apresentado por Choi [Choi 2013] e resumido na Tabela 2 não são abordados os conceitos de modelagem utilizando UML (*Unified Modeling Language [Booch et al. 2005]*) nessas duas disciplinas. Esses conceitos são abordados em outra disciplina específica para tratar o assunto de modelagem de software.

A Tabela 3 apresenta o planejamento da disciplina Engenharia de Software I de 2 créditos, ou seja, 2h semanais por 15 semanas, totalizando 30 horas em sala de aula, sendo ministrada para alunos de segundo semestre. A disciplina é utilizada para introduzir a metodologia de sala de aula invertida para os alunos, assim apenas 10 das 15 aulas são invertidas [3] e a composição da nota final conta com a nota dos relatórios das atividades em grupo, nota dos questionários individuais e duas provas individuais. As provas individuais foram mantidas nessa disciplina para permitir que os alunos fossem avaliados através de um método tradicional, além da forma de avaliá-los pela metodologia ativa.

O material de estudo pré-aula é composto de capítulos de livro ou apostilas específicas para o conteúdo trabalhado, além de vídeo-aulas gravadas com as explicações dos pontos mais importantes de cada tópico. Um plano de melhoria para os próximos oferecimentos é estruturar o caminho de aprendizado dos alunos utilizando alguma ferramenta de autoria como a apresentada em [Veras et al. 2021].

A Tabela 4 apresenta o planejamento da disciplina Engenharia de Software II de 4 créditos, ou seja, 4h semanais por 15 semanas, totalizando 60 horas em sala de aula, sendo ministrada para alunos de terceiro semestre. Como os alunos já estão habituados com a

[3] As aulas #1, #2 e #14 não são invertidas e a as aulas #8 e #15 são para a aplicação das provas.

Tabela 3. Planejamento - Engenharia de Software I

#	Pré-Aula	Aula
1		Introdução - A história da Engenharia de Software
2		- A natureza do software - Princípios da engenharia de software
3	Paradigmas de Desenvolvimento de software - Leitura de apostila - Q01 - Paradigmas	At01: Mudança de Paradigma - Desenvolver Procedimental e analisar Mudança para Orientado a Objetos
4	Modelos de Processo - Leitura Cap. Livro - Cascata - Leitura Artigo W. Royce - VA02 - Modelo Cascata - Q02 - Modelo Cascata	At02: Desenvolvimento Cascata - Desenvolvimento de um pequeno sistema seguindo as fases do modelo cascata
5	Modelo Espiral - Leitura Cap. Livro - Espiral - Leitura Artigo B. Boehm - Q03 - Modelo Espiral	At03: Comparação Cascata x Espiral
6	Processo Unificado - UP - Leitura RUP Made Easy - Leitura Cap 5 R. Wazlawick - VA04 - RUP - Q04 - RUP	At04:Elaboração de perguntas sobre RUP - Competição entre as equipes
7	Desenvolvimento Ágil - Leitura eXtreme Programming - VA05a - XP - Leitura Scrum - VA05b - Scrum	At05: eXtreme Programming - Aplicação e avaliação de Pair Programming
8		P1 - Avaliação parcial 1
9	Requisitos de Software - Leitura Requisitos de Sw - VA06 - Requisitos	At06: Formulário de Requisitos de Software
10	Técnicas de Levantamento de Requisitos - VA07 - Levantamento de Req.	At07: Aplicando Entrevista Estruturada
11	Análise de Risco - Leitura Cap Livro	At08: Aplicando análise de risco
12	Planejamento de Projeto - Leitura Cap Livro - Q09 - Plan. Projeto	At09: Desenvolvimento o projeto de Arquitetura
13	Teste de Software - Leitura apostila - VA10 - Introd Teste de Sw	At10: Elaboração de casos de teste
14		Revisão para Prova
15		P2 - Avaliação parcial 2

rotina da sala de aula invertida nesta disciplina praticamente todas as aulas são invertidas. Com exceção de primeira pois nessa aula são passadas as instruções de como a disciplina será desenvolvida ao longo do semestre. A avaliação da disciplina é dividida em uma parte individual e uma parte para as atividades em grupo. Todas as aulas tem a entrega de uma atividade e na maioria das aulas os alunos devem responder a um questionário pré-aula com questões sobre o conteúdo da aula da semana. É aplicado também um questionário inicial e um final para avaliar a evolução dos alunos sobre os conceitos trabalhados ao longo da disciplina, esses questionários são avaliados e valem um bônus na nota final. O conjunto de questões é exatamente o mesmo, sendo apresentado em ordem diferente.

Ao final da aula de uma dada semana os alunos já têm acesso ao conteúdo a ser estudado para a semana seguinte. E também ao questionário relacionado a esse conteúdo. Eles podem escolher qualquer momento da semana para responder ao questionário, que geralmente tem de 5 a 15 perguntas de múltipla escolha. Apenas uma tentativa pode ser feita para responder o questionário, sendo que as perguntas e as alternativas são embaralhadas para evitar que um aluno consulte as respostas com outro. Uma vez iniciado o questionário o aluno tem 1 hora para finalizá-lo. O resultado final só estará disponível para os alunos depois que o período em que o questionário está habilitado estiver encerrado.

No início do encontro da aula é feito um breve resumo do conteúdo estudado, destacando os pontos mais importantes e aqueles que serão mais relevantes para a atividade do dia. Em seguida o enunciado da atividade é disponibilizado para os alunos e é feita uma breve explicação da atividade em si. A partir desse momentos os grupos, sorteados previamente, se organizam fisicamente na sala de aula e iniciam os trabalhos. Qualquer

Tabela 4. Planejamento - Engenharia de Software II

#	Pré-Aula	Aula
1		Apresentação da disciplina e de conteúdo programático. Qi - Questionário inicial A1 - Qualidade de Código
2	Aspectos humanos da engenharia de software - Leitura Cap. Livro - V2 - Vídeo Aula - Q2 - Questionário	Aspectos Humanos da Eng. de Software A2 - Selecionando um Engenheiro de Software Efetivo
3	Projeto de Interface - Leitura Cap. Livro - V3 - Video Aula - Simplicity Sells - David Pogue - Q3 - Questionário	A3 - Avaliando a Interface de um sistema real (regras de ouro de Mandel)
4	Revisão técnica formal - Leitura Cap. Livro - V4 - Video Aula - Q4 - Questionário	A4 - Aplicando Revisão Técnica Formal - Desenvolvimento de um pequeno programa. - Preparação para a revisão (checklist) - Condução da reunião e elaboração do relatório.
5	Garantia de Qualidade de Software - Leitura Cap. Livro - Q5 - Questionário	A5 - Modelagem de um processo de SQA utilizando BPMN - Business Process Model and Notation
6	Modelagem de Sistemas - Leitura Apostila	A6 - Modelagem de um diagrama de classes
7	Análise e Projeto OO - Leitura apostila	A7 - Do modelo para o código
8	Estratégias de Teste de Software - Leitura Cap. Livro - Leitura de Artigo (teste automatizado) - Q8 - Questionário	A8 - Teste Unitário utilizando JUnit
9	Teste de Aplicações Convencionais - Leitura Cap. Livro - Q9 - Questionário	A9 - Comparanto teste funcional e teste estrutural
10	Teste de Aplicativos Web - Leitura Cap. Livro - Q10 - Questionário	A10 - Avaliação de não conformidades em uma aplicação Web
11	Testes de Aplicações Móveis - Leitura Apostila - Q11 - Questionário	A11 - Teste de usabilidade de aplicativos móveis
12	Gestão de Configuração e Versão - Leitura Cap. Livro - Q12 - Questionário	A12 - Atividade utilizando GIT
13	Métricas de Produto - Leitura Cap. Livro - Q13 - Questionário	A13 - Aplicação das métricas CK a um sistema real em Java
14	Manutenção de Software - Leitura Cap. Livro - Q14 - Questionário	A14 - Propor manutenção corretira, adaptativa e perfectiva a um software real.
15	Melhoria de Processo de Software - Leitura Cap. Livro - Q15 - Questionário	Qf - Questionário final A15 - Gincana sobre CMMI

nova instrução ou comunicação a partir desse ponto é feita somente para os líderes de cada equipe (também sorteados) e esses devem repassar as instruções para os outros membros.

O líder de cada equipe tem um papel diferenciado, ele é o responsável por dividir as tarefas, organizar o relatório, cobrar os membros das equipes, lembrar os outros membros de responder a chamada e alinhar com os outros líderes. Esse papel diferenciado é destacado para os aluno assim como o fato de que essa atuação também faz parte da disciplina de engenharia de software.

O enunciado das atividades é bem detalhado e mostra claramente a sequencia das atividades que devem ser executadas e quais são os entregáveis. Ao final cada grupo produz um relatório com todos os entregáveis daquela atividade e devem submeter pelo Moodle [Dougiamas and Taylor 2003]. Em geral o prazo para essa entrega é 30 minutos antes do final da aula. Depois que todos os grupos fazem a entrega, uma das equipes é sorteada para fazer a apresentação do que foi feito e um *feedback* geral é passado para a turma. Com isso os alunos não levam nenhuma tarefa para casa em relação a atividade realizada em sala, ficam apenas aguardando o *feedback* individualizado para o relatório entregue e a nota. Enquanto isso já vão se preparando para a próxima aula, lendo o material, assistindo as vídeo aulas e respondendo o questionário seguinte.

4.1. Vantagens

Uma das principais vantagens da aplicação da metodologia de sala de aula invertida, de modo geral, é a de permitir que cada aluno tenha o seu próprio ritmo para estudar e absorver o conteúdo teórico de forma adequada. Em uma sala de aula tradicional o professor deve calibrar o ritmo das explicações para atender o maior número de alunos, em geral, utilizando sua experiência e avaliando o comportamento dos alunos para saber dosar sua velocidade. Isso pode ser um grande desafio, especialmente em turmas mais heterogêneas. Os alunos mais rápidos sempre terão que esperar pelos outros, o que pode desmotivá-los e os alunos mais lentos terão dificuldade para acompanhar o ritmo da sala, o que pode comprometer o seu entendimento do conteúdo. Além de se ajustar ao ritmo do aluno a apresentação do conteúdo pré-aula também pode ser feita de formas diferentes, permitindo que o aluno se ajuste ao formato que lhe é mais apropriado. Pode-se utilizar material escrito para leitura, áudio em formato de podcasts, vídeos indicados ou produzidos pelo próprio professor. Esses vários formatos podem ser usados de maneira combinada.

O tempo de aula é utilizado em atividades mais relevantes ao aprendizado [Tucker 2012] e onde a presença do professor pode ser melhor aproveitada. No caso específico de engenharia de software isso permite a utilização da aplicação prática do conteúdo, destacada como essencial no ensino dessa disciplina [Marques Samary et al. 2015]. Além de colocar o conteúdo teórico em prática o aluno tem a oportunidade de trocar experiências com outros alunos o que pode ser tão produtivo e eficaz quanto a interação com o próprio professor. Experimentos controlados mostram que alunos expostos à metodologia têm notas significativamente maiores do que alunos que fizeram uma mesma disciplina de forma tradicional [Day and Foley 2006].

4.1.1. Pontos de Atenção

A implantação de uma metodologia nova em uma disciplina sempre gera um esforço adicional de planejamento e preparação, principalmente nos primeiros oferecimentos. Isso não é diferente com a sala de aula invertida. É necessário fazer uma curadoria do material de estudo, tanto para material de leitura quanto para vídeos e *podcasts* já disponíveis na Internet. Pode também ser necessário que se produzam vídeos específicos sobre o conteúdo que precisa ser explicado. Além disso, é preciso se investir tempo e muita criatividade para se planejar as atividades práticas que serão conduzidas ao longo do semestre. Não se pode menosprezar a dificuldade desse trabalho e certamente ajustes precisarão ser feitos em oferecimentos subsequentes.

Mesmo quando esse esforço de preparação já está superado a metodologia exige tempo de dedicação do professor em um ritmo semanal, principalmente para se fazer a correção e dar *feedback* nos relatórios produzidos pelos alunos durante a realização das atividades práticas. Essa dedicação de tempo é adicional ao tempo em sala de aula propriamente dito e é, em geral, superior ao tempo de preparação de aula normalmente dedicado pelo professor, especialmente a partir do segundo oferecimento da disciplina. É importante que o docente tenha ciência desse impacto em termos de tempo de dedicação antes de iniciar a transição para a metodologia. O *feedback* sobre o que foi desenvolvido em aula é fundamental. Um *feedback* geral sobre o que foi desenvolvido pode ser feito ao final da aula síncrona, logo depois dos grupos terem feito a entrega dos relatórios.

A formulação das atividades práticas é de extrema importância [Auster and Wylie 2006]. Elas devem ter um objetivo claro e devem se apoiar no conteúdo teórico estudado em pré-aula. Não se deve formular atividades apenas para manter os alunos ocupados. Para isso é importante que o enunciado das atividades seja revisado a cada novo oferecimento e caso os objetivos da aula não tenham sido atingidos com a atividade anterior uma nova formulação deve ser feita para o novo oferecimento. Deve-se promover a participação de todos os integrantes das equipes na atividade, inclusive, se possível, com responsabilidades diferentes entre eles. Deve-se estimular que os integrantes das equipes interajam entre si e com as outras equipes (se possível). Os líderes de cada equipe ficam com o papel de organizadores e negociadores dentro da própria equipe e entre as equipes.

5. Conclusão

Foi apresentada a experiência de aplicação da metodologia sala de aula invertida em disciplinas da área de engenharia de software. Com a metodologia pretende-se aproveitar melhor o tempo do professor em sala de aula, permitindo que a aplicação prática do conteúdo estudado possa ser supervisionada e orientada e que a absorção do conteúdo teórico possa ser feito no ritmo e estilo de aprendizado de cada aluno, o que permite um aproveitamento mais uniforme.

Uma série de vantagens foram observadas, sendo que a melhor retenção do conteúdo trabalhado é o principal destaque. Além de uma maior motivação dos alunos em participar das aulas. Alguns pontos de atenção também merecem destaque, principalmente em relação à dosagem do trabalho adicional exigido dos alunos e também do professor. O aluno precisa se preparar antes do encontro síncrono, lendo o material disponibilizado ou assistindo aos vídeos explicativos. O professor precisa antecipar a preparação e o planejamento das aulas, ter criatividade para elaborar boas atividades práticas, além de precisar dar *feedback* semanal em todo o material produzido pelas alunos.

A maior parte do *feedback* recebido dos alunos é positivo, geralmente citam que o conteúdo trabalhado foi melhor fixado devido à aplicação imediata em atividades práticas. Ex-alunos relatam que se lembram das aulas e aplicaram o conhecimento em seu trabalho, vários semestres depois de terem passado pela disciplina. Em torno de 2% dos alunos dão um *feedback* negativo, que geralmente é algo do tipo: ”a disciplina é muito puxada, com muito conteúdo para ser estudado e muita demanda durante as aulas” o que, no fundo, é verdade. De qualquer forma o *feedback* recebido de ex-alunos é bastante positivo e motiva a continuidade e evolução da adoção da metodologia. O *feedback* dos alunos é obtido ao final do semestre através de um processo de avaliação formal das disciplinas do curso. O *feedback* obtido de ex-alunos é espontâneo, geralmente recebido em eventos ou através de redes sociais como o LinkedIn.

Referências

Auster, E. R. and Wylie, K. K. (2006). Creating active learning in the classroom: A systematic approach. *Journal of Management Education*, 30(2):333–353.

Bergmann, J. and Sams, A. (2012). *Flip Your Classroom: Reach Every Student in Every Class Every Day*. International Society for Technology in Education.

Booch, G. (2019). The history of software engineering. *IEEE Software*, 35(5):108–114.

Booch, G., Rumbaugh, J., and Jacobson, I. (2005). *Unified Modeling Language User Guide, The (2nd Edition) (Addison-Wesley Object Technology Series)*. Addison-Wesley Professional.

Choi, E.-M. (2013). Applying inverted classroom to software engineering education. *International Journal of e-Education, e-Business, e-Management and e-Learning*.

Day, J. A. and Foley, J. D. (2006). Evaluating a web lecture intervention in a human–computer interaction course. *IEEE Transactions on Education*, 49(4):420–431.

Dougiamas, M. and Taylor, P. C. (2003). Moodle: Using learning communities to create an open source course management system. In *Proceedings of the EDMEDIA 2003 Conference, Honolulu, Hawaii*.

Gannod, G. C., Burge, J. E., and Helmick, M. T. (2008). Using the inverted classroom to teach software engineering. In *Proceedings of the 30th International Conference on Software Engineering*, ICSE '08, page 777–786, New York, NY, USA. Association for Computing Machinery.

Herold, M. J., Lynch, T. D., Ramnath, R., and Ramanathan, J. (2012). Student and instructor experiences in the inverted classroom. In *2012 Frontiers in Education Conference Proceedings*, pages 1–6.

Marques Samary, M., Quispe, A., and Ochoa, S. (2015). A systematic mapping study on practical approaches to teaching software engineering. *Proceedings - Frontiers in Education Conference, FIE*, 2015.

on Computing Curricula, T. J. T. F. (2004). Curriculum guidelines for undergraduate degree programs in software engineering. Technical report, New York, NY, USA.

Pressman, R. (1982). *Software Engineering: A Practitioner's Approach*. McGraw Hill.

Sureka, A., Gupta, M., Sarkar, D., and Chaudhary, V. (2013). A case-study on teaching undergraduate-level software engineering course using inverted-classroom, large-group, real-client and studio-based instruction model. *CoRR*, abs/1309.0714.

Tucker, B. (2012). The flipped classroom: Online instruction at home frees class time for learning. *Education Next*, 12:82.

Veras, N. L., Rocha, L. S., and Viana, W. (2020). Flipped classroom in software engineering: A systematic mapping study. In *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*, SBES '20, page 720–729, New York, NY, USA. Association for Computing Machinery.

Veras, N. L., Rocha, L. S., and Viana, W. (2021). An authoring tool to support flipped classroom in software engineering teaching. In *Proceedings of the 17th ACM Conference on International Computing Education Research*, ICER 2021, page 403–404, New York, NY, USA. Association for Computing Machinery.