
UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE EDUCAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM EDUCAÇÃO MATEMÁTICA E TECNOLÓGICA
CURSO DE DOUTORADO

RICARDO TIBURCIO DOS SANTOS

# A ENGENHARIA DIDÁTICO-INFORMÁTICA: uma metodologia para a produção de software educativo

Recife
2020

RICARDO TIBURCIO DOS SANTOS

# A ENGENHARIA DIDÁTICO-INFORMÁTICA: uma metodologia para a produção de software educativo

Tese apresentada ao Programa de Pós-Graduação em Educação Matemática e Tecnológica do Centro de Educação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do Título de Doutor em Educação Matemática e Tecnológica.

**Área de concentração**: Ensino de Ciências e Matemática.

**Orientador**: Prof. Dr. Franck Gilbert René Bellemain

Recife
2020

Catalogação na fonte
Bibliotecário Danilo Leão, CRB-4/2213

S237e Santos, Ricardo Tiburcio dos.
A engenharia didático-informática: uma metodologia para a produção de software educativo. / Ricardo Tiburcio dos Santos. – Recife, 2020.
194p.

Orientador: Franck Gilbert René Bellemain.
Tese (Doutorado) - Universidade Federal de Pernambuco, CE. Programa de Pós-graduação em Educação Matemática e Tecnológica, 2020.
Inclui Referências.

1.Metodologia. 2. Desenvolvimento de software. 3. Engenharia de software. 4. UFPE - Pós-graduação. I. Bellemain, Franck Gilbert René. (Orientador). II. Título.

371.3 (23. ed.) UFPE (CE2020-074)

RICARDO TIBURCIO DOS SANTOS

**A ENGENHARIA DIDÁTICO-INFORMÁTICA: uma metodologia para a produção de software educativo**

Tese apresentada ao Programa de Pós-Graduação em Educação Matemática e Tecnológica do Centro de Educação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do Título de Doutor em Educação Matemática e Tecnológica.

Aprovado em 15/09/2020.

**BANCA EXAMINADORA**

_____

Prof. Dr. Franck Gilbert René Bellemain (Orientador e Presidente)
Universidade Federal de Pernambuco

_____

Prof. Dr. Sérgio Paulino Abranches (Examinador Interno)
Universidade Federal de Pernambuco

_____

Profa. Dra. Anna Paula de Avelar Brito Lima (Examinadora Externa)
Universidade Federal Rural de Pernambuco

_____

Prof. Dr. Péricles de Lima Sobreira (Examinador Externo)
Université de l'Outaouais et des Laurentides

_____

Profa. Dra. Marilena Bittar (Examinadora Externa)
Universidade Federal do Mato Grosso do Sul

DEDICATÓRIA

Este trabalho é dedicado a **Maura Helena** (*In memorian*), mulher doce e guerreira que tive o orgulho de chamá-la de **Mainha**. Com paixão, brilho nos olhos e empatia, deixou na Terra o seu maior legado: o **amor**.

# AGRADECIMENTOS

A Deus, pelo livre arbítrio.

A minha família, por compreender algumas ausências e me apoiar em toda minha trajetória acadêmica.

Ao Professor Franck Bellemain, pela parceria firmada desde o Mestrado, por ser um orientador excepcional e por me guiar nessa jornada com dedicação, empenho e amizade.

Aos meus amigos por se fazerem presentes em todos os bons e maus momentos, apoiando, incentivando e comemorando cada vitória e conquista.

A Banca Examinadora, constituída pelos professores Sérgio Abranches, Anna Paula Avelar, Marilena Bittar e Péricles Sobreira. Obrigado pelos conhecimentos compartilhados e significativas contribuições para esta pesquisa.

Aos professores e professoras do EDUMATEC, pelos conhecimentos, vivências e contribuições. Em particular a Professora Paula Baltar Bellemain, que acompanhou o início do meu trabalho e sempre teve excelentes intervenções para a melhoria da pesquisa.

Aos colegas de Mestrado e Doutorado que acompanharam minha trajetória e contribuíram efetivamente com o desenvolvimento deste estudo. Em particular, César Thiago, Rosilângela Lucena, Roberto Mariano, Aílson Alzeri, Anderson Douglas, Ana Paula Lima, Edeson Siqueira e Amanda Rodrigues, muito grato por terem compartilhado conhecimentos e experiências ao longo desses anos.

# RESUMO

No contexto de desenvolvimento de software educativo, estudos e investigações apresentam uma problemática em relação à qualidade de alguns produtos vinculada à engenharia a qual são submetidos. Com os resultados da Pesquisa de Mestrado, percebemos que uma solução para a Engenharia de Software Educativo é a modelização dos processos no sentido de utilizar uma metodologia que contemple aspectos teóricos sobre o ensino e a aprendizagem e, também, aspectos tecnológicos levando em consideração os avanços digitais. Nossa primeira solução foi a concepção da Engenharia Didático-Informática – EDI; essa metodologia se constitui da utilização dos procedimentos metodológicos de duas engenharias: Didática e de Software, articulando as análises teóricas e sistemáticas de ambas. Consideramos como hipóteses que: 1. Compreender metodologias de desenvolvimento de software educativo colabora para a criação de modelizações de processos; 2. A realização de um estudo histórico traz subsídios para o aprimoramento da Engenharia Didático-Informática; 3. A análise das utilizações da EDI, na sua primeira versão, produz elementos para aprimorar a modelização do processo de software. Isso posto, esta pesquisa teve por objetivo aperfeiçoar a EDI bem como o modelo de processo de desenvolvimento de software dessa metodologia, assim realizando uma abordagem histórica e analítica. Com a EDI, observamos a possibilidade de suprir algumas carências verificadas nas formas de produzir software educativo. As análises propostas no modelo de processo norteiam a criação de produtos para atender às características específicas dos conhecimentos que serão trabalhados, porém, através do uso dessa metodologia e de discussões em torno da problemática de criação de software, percebemos a necessidade de aperfeiçoamento. O percurso metodológico dessa pesquisa foi iniciado com a análise de como os software Casyopée, Function Probe e Modellus foram desenvolvidos, desde a concepção até a fase de testes dos projetos, afim de observar os padrões de engenharia utilizados – resgate histórico. Também, fez parte da metodologia compreender como a Engenharia Didático-Informática foi utilizada nos projetos de concepção dos software Function Studium, Conics Studium 3D e Magnitude Studium, como meio de obter fundamentação para aprimorar o processo em evolução – abordagem analítica. Os resultados deste estudo indicam que existe uma preocupação em considerar teorias sobre o ensino e a aprendizagem para desenvolver tecnologias digitais educativas e aspectos metodológicos oriundos da Engenharia de Software; com o resgate histórico foram observadas semelhanças entre as engenharias utilizadas e padrões de levantamento de requisitos desses software. Outro resultado é a realização da análise da utilização da EDI. Logo, foi observado que existe a necessidade de aprimoramento dessa

engenharia para que essa possa fornecer informações mais detalhadas de cada parte do processo, visto que algumas etapas não são tão objetivas e simples de serem compreendidas, conforme indicam os pesquisadores que utilizaram essa metodologia. Assim, a Engenharia Didático-Informática foi aperfeiçoada e sua versão atual é apresentada nesta Tese.

**Palavras-chave**: Engenharia Didático-Informática. Desenvolvimento de Software Educativo. Engenharia de Software.

RESUMÉ

Dans le cadre du développement de logiciels pédagogiques, plusieurs études et recherches posent le problème de la qualité de certains produits en conséquence de l'ingénierie dont ils sont fruits. Grâce aux résultats de notre recherche de master, nous avons réalisé qu'une solution pour le génie logiciel éducatif passe par la modélisation des processus par l'utilisation d'une méthodologie qui prend en compte les aspects théoriques sur l'enseignement et l'apprentissage ainsi que les aspects technologiques compte tenu des avancées du numériques. Notre première solution a été la conception de l'Ingénierie Didactique Informatique – IDI, méthodologie qui consiste à utiliser les principes méthodologiques de deux ingénieries : Didactique et Logiciel, articulant les analyses théoriques et méthodologiques de chacune. Nous considérons comme hypothèses que 1. La compréhension de méthodologies de développement de logiciels éducatifs contribue à la création de la modélisation de processus ; 2. La réalisation d'une étude historique apporte des éléments pour l'amélioration de l'ingénierie didactique-informatique ; 3. L'analyse de mises en œuvre de l'IDI, dans sa première version, produit des éléments pour améliorer la modélisation du processus de conception et réalisation de logiciel. Cette recherche visait donc à améliorer l'IDI ainsi que le modèle de processus de développement de logiciel proposé par cette méthodologie en réalisant une approche historique et analytique. Avec l'IDI, nous avons vu qu'il était possible de combler certaines lacunes vérifiées dans les manières de produire des logiciels éducatifs : les analyses proposées dans le modèle de processus guident la création de produits pour répondre aux caractéristiques spécifiques des connaissances qui seront travaillées. Cependant, par la mise en œuvre de cette méthodologie et les discussions relatives au problème de la création de logiciels, nous avons ressenti la nécessité d'améliorations. Le parcours méthodologique de cette recherche a débuté par l'analyse du développement des logiciels Casyopée, Function Probe et Modellus, de la conception à la phase de test des projets, afin de dégager les standards d'ingénierie utilisés – approche historique. Puis elle s'est poursuivie par l'analyse de la mise en œuvre de l'ingénierie didactique-informatique dans les projets de conception des logiciels Function Studium, Conics Studium 3D et Magnitude Studium, afin d'obtenir des éléments d'amélioration du processus en évolution - approche analytique. Les résultats de cette étude indiquent qu'il y a une necessité de considérer les théories sur l'enseignement et l'apprentissage pour développer les technologies éducatives numériques et les aspects méthodologiques du génie logiciel. Dans l'approche historique, des similitudes ont été observées entre l'ingénierie utilisée et les normes pour le cahier des charges de ces logiciels. Un autre résultat est l'analyse de l'utilisation de l'IDI dans laquelle il a été mis

en évidence un besoin d'améliorer cette ingénierie afin qu'elle puisse fournir des informations plus détaillées sur chaque partie du processus. Les chercheurs qui ont utilisés cette méthodologie indiquent que certaines étapes ils ne sont pas aussi objectives et simples à comprendre. En conséquence de ces analyses, l'ingénierie didactique-informatique a été perfectionnée et sa version actuelle est présentée dans cette thèse.

**Mots clés** : Ingénierie Didactique-Informatique. Développement de logiciels éducatifs. Ingénierie des Logiciel.

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE DIAGRAMAS

**SUMÁRIO**

# 1 INTRODUÇÃO

Como desenvolver um software que atenda às especificidades dos conhecimentos matemáticos? Como criar recursos que contribuam efetivamente para a atividade docente? Quais características um artefato tecnológico deve possuir para auxiliar a aprendizagem do estudante? Quais contribuições tecnológicas podem ser úteis para um software educativo?

As respostas destas questões possuem certa fragilidade, visto que não existem respostas objetivas para as mesmas, quando se discute a criação de software educativo – SE, pois, em uma primeira análise, envolve as áreas de Epistemologia, Ensino, Aprendizagem e Tecnologia. A fragilidade em questão reside nas diversas abordagens que são realizadas para a concepção de SE: a centralização em uma dessas áreas em detrimento das demais (BELLEMAIN, BELLEMAIN; GITIRANA, 2014).

A crescente evolução das tecnologias faz com que o ensino e a aprendizagem passem atualmente por uma reconfiguração, não apenas para acompanhar o desenvolvimento das tecnologias de informação e comunicação, mas para que essa evolução favoreça as práticas educativas. As alterações da cultura da sociedade são influenciadas pelo avanço da tecnologia: novas formas de trabalho, de relacionamento pessoal e profissional, de ensinar e aprender, entre outras. Alguns fatores para o crescimento da oferta de cursos à distância, por exemplo, é o progresso das tecnologias digitais, o acesso mais fácil a internet, a disseminação em massa de smartphones, entre outras circunstâncias.

Entretanto, um aspecto a ser considerado sobre a utilização de recursos tecnológicos no ensino e na aprendizagem são as características peculiares dos SE disponíveis. Estudos apresentam a problemática da qualidade da maioria dos software, indicando a ausência de uma engenharia que dê suporte para a construção adequada dos programas com a finalidade de contemplar a maior quantidade possível de características dos saberes que serão trabalhados (BENITTI, SEARA; SCHLINDWEIN, 2005; SANTOS, 2009).

De acordo com Benitti et al (2005, p. 2), “os software educacionais existentes – em sua grande maioria – possuem problemas que dificultam a sua utilização, dentre eles a falta de uma base pedagógica que fundamente a sua construção”. Com isso, compreende-se a relevância dessa base, citada pelos autores, entre outros fatores, como a relevância de inserir na concepção do software fundamentos (teóricos e metodológicos) sobre o ensino e a aprendizagem.

Para Bellemain et al (2014) enunciar a relevância da articulação entre soluções tecnológicas e abordagens teóricas parece evidente, entretanto muitas soluções tecnológicas –

recursos digitais, aplicativos, software, entre outros, para auxiliar na aprendizagem e no ensino não se fundamentam nessa articulação, existe uma dualidade evidente: “buscam explorar a potencialidade da tecnologia, mas desconsideram a produção acadêmica (disciplinar e didática) sobre os conteúdos veiculados pela tecnologia”, por um lado, “ora se apoiam nos conhecimentos sobre os conteúdos e sua didática, mas exploram muito pouco das potencialidades do computador” (p. 4).

Com isso, um questionamento mais amplo pode generalizar o problema da criação de SE: como desenvolver recursos tecnológicos educativos que atendam as demandas da Educação Matemática e Tecnológica? Uma das possíveis respostas são as metodologias de desenvolvimento. Logo, é necessário compreender qual é o paradigma atual de criação de SE: como são desenvolvidos? Quais são as estratégias? Quais são as abordagens? Além de considerar quais são os resultados das pesquisas nessa área.

Nossa primeira resposta para a problemática da criação de SE é a **Engenharia Didático-Informática - EDI.** Essa engenharia foi concebida e publicada, inicialmente, nos estudos de Tiburcio e Bellemain (2018) Tiburcio (2016) e Silva (2016). O termo se constitui da articulação dos procedimentos metodológicos de duas engenharias: A Engenharia Didática[1] (Clássica ou de primeira geração) e a Engenharia de Software (oriunda da Ciência da Computação).

Em particular, os resultados da Pesquisa de Mestrado (TIBURCIO, 2016) expuseram a importância de considerar a conexão entre os saberes sobre ensino e aprendizagem e a exploração de potencialidades tecnológicas. Em Tiburcio (2016), compreendemos a indispensabilidade de associar elementos da Engenharia de Software com as contribuições de estudos e investigações na área da Educação da Matemática.

Tiburcio (2016), em pesquisa de Mestrado, ao verificar a necessidade de relacionar teorias (e construtos teóricos) sobre ensino e aprendizagem com os referenciais da Engenharia de Software apresentou uma proposta de desenvolvimento de SE que articula a Engenharia de Software – ES e Engenharia Didática – ED. Para dessa forma desenvolver produtos tecnológicos que contemplem as minúcias das relações educativas.

Fora observada a relevância de sistematizar os elementos pertinentes das duas engenharias: A ED, com os subsídios de investigação teóricos e experimentais sobre o ensino e a aprendizagem e a ES, com as metodologias de construção de software e técnicas de obtenção de requisitos, objetivando assim construir um processo de desenvolvimento de software

[1] Termo original: Ingénierie Didactique (ARTIGUE; 1996, 2002, 2009; BROUSSEAU; 1976, 1981).

educativos que considere os avanços tecnológicos, mas que não despreze os estudos teóricos realizados que contribuem com as relações educativas.

Desse modo, propusemos, em estudo de Mestrado (TIBURCIO, 2016) uma nova abordagem de metodologia de criação de produtos tecnológicos para o ensino e a aprendizagem de saberes matemáticos (e outros possíveis): A Engenharia Didático-Informática. Contudo, percebemos que era necessário prosseguir com as investigações sobre esta metodologia para aperfeiçoá-la em pesquisa de Doutorado.

Uma das justificativas para a continuidade da referida pesquisa de Mestrado foi a observação de como a EDI estava sendo utilizada em pesquisas no âmbito do desenvolvimento de software educativo. Os software Function Studium (SILVA, 2014), Conics Studium 3D (SIQUEIRA, 2019) e Magnitude Studium (SILVA, 2020), foram desenvolvidos utilizando as premissas e encaminhamentos metodológicos da EDI. O autor da presente Tese participou ativamente das discussões, observou a construção inicial das ideias de cada um dos software citados, indagou e compreendeu as hipóteses dos pesquisadores e analisou como a engenharia foi utilizada em cada um dos projetos.

A motivação para aperfeiçoar a EDI surgiu quando os pesquisadores que a utilizavam elencaram limitações e possibilidades com a aplicação da engenharia, bem como dificultadores e facilitadores desse uso. A princípio, pensávamos em agregar na engenharia outros referenciais teóricos e metodológicos que pudessem auxiliar nos problemas que foram constatados, entretanto, as interações com os pesquisadores e as leituras realizadas das literaturas produzidas quanto ao uso da EDI (artigos, teses de doutorado e dissertação de mestrado), percebemos que as respostas para a evolução estavam na compreensão desses dificultadores e em como superá-los.

Sendo assim, optamos, inicialmente, por realizar uma abordagem analítica, verificando as produções desses pesquisadores quanto a utilização da metodologia em questão a fim de elencar os problemas por eles verificados e encontrar soluções para esses, lançando uma nova versão da Engenharia Didático-Informática. Contudo, ao nos depararmos com essa abordagem, surgiu a ideia de realizar também uma análise histórica, verificando as engenharias as quais outros software foram submetidos com a finalidade de compreender pontos de semelhança e distanciamento e reunir elementos objetivando agregá-los a EDI.

A ideia era que compreender desenvolvimentos de recursos digitais educativos poderia colaborar para a compreensão de como as engenharias de software eram modelizadas.

Começamos a nos indagar e depois constatamos em nossa revisão de literatura que não havia uma engenharia própria para o desenvolvimento de software educativo, porém existem diversos produtos digitais que contribuem efetivamente com as relações de ensino e aprendizagem e esses foram construídos seguindo, minimamente, algum padrão, protocolo ou método. Questões simples surgiram: "se não existe uma engenharia específica para o desenvolvimento de software educativo, como esses software foram desenvolvidos?"; "Como foram idealizados?"; "Consideravam questões teóricas sobre o ensinar e o aprender?"; "Eram fundamentados em teorias didáticas, cognitivas, epistemológicas?"; e ainda "Tentavam sanar dificuldades observadas com os potenciais da tecnologia?".

A partir desses questionamentos e de outros que surgiram ao longo do levantamento de hipóteses da presente pesquisa, percebemos que seria interessante realizar uma abordagem histórica para resgatar as engenharias de alguns software com pesquisas consolidadas na Educação Matemática. Assim, escolhemos o Casyopée (LAGRANGE, 2005), o Function Probe (CONFREY, 1992) e o Modellus (TEODORO, 2002) por serem recursos digitais da mesma tipologia que os softwares que utilizaram a EDI e por terem o contexto acadêmico de investigação e desenvolvimento.

Portanto, definimos assim uma proposta de aperfeiçoamento da Engenharia Didático-Informática, baseando-nos na atualização do referencial teórico que fundamenta essa metodologia, bem como na abordagem analítica, observando como a EDI foi utilizada e na análise histórica, resgatando elementos de engenharias de software consolidados em suas áreas de conhecimento. Consideramos, assim, que a pesquisa realizada tem caráter teórico, visto que nos propusemos a realizar análises de pesquisas de desenvolvimento de software educativo e atualizar os referenciais teóricos e metodológicos.

Pretendemos, com as investigações e estudos realizados, oferecer, com a EDI, uma metodologia de desenvolvimento de software educativo que considere as pesquisas e encaminhamentos sobre o ensino e a aprendizagem aliado com as potencialidades das tecnologias digitais atuais. Assim, a EDI é construída com fundamentação teórica e metodológica na área da Educação Matemática, em particular com a Engenharia Didática, aliada aos avanços tecnológicos e métodos de desenvolvimento de software, utilizando-se dos direcionamentos da Engenharia de Software.

## 1.1 UMA PROPOSTA DE APERFEIÇOAMENTO, OBJETIVOS E METODOLOGIA

Com o exposto, consideramos três hipóteses nesta investigação: 1. Compreender metodologias de desenvolvimento de software educativo colabora para a criação de modelizações de processos; 2. A realização de um estudo histórico traz subsídios para o aprimoramento da Engenharia Didático-Informática; 3. A análise das utilizações da EDI, na sua primeira versão, produz elementos para aprimorar a modelização do processo de software.

### 1.1.1 Objetivo Geral

Esta pesquisa teve por objetivo aperfeiçoar a Engenharia Didático-Informática, bem como o modelo de processo de desenvolvimento de software dessa metodologia, realizando uma abordagem histórica e analítica.

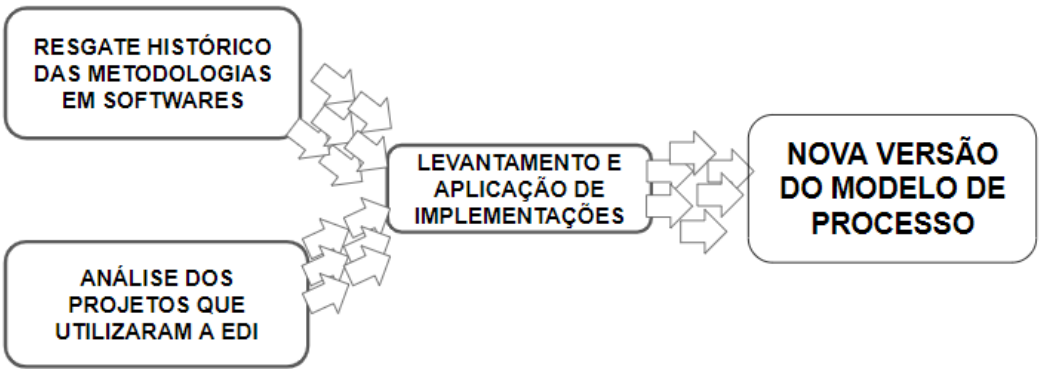
### 1.1.2 Objetivos Específicos

- Analisar a utilização da EDI em projetos de desenvolvimento de software educativos como meio de obter fundamentação para aprimorar o processo em evolução;
- Investigar modelizações de desenvolvimento de software educativo em uma abordagem histórica para obter informações de modo a contribuir com a nova versão da EDI;
- Observar as contribuições dos estudos analisados para lançar nova versão do modelo de processo da EDI.

Os procedimentos metodológicos utilizados neste estudo refletem a problemática da utilização de recursos tecnológicos na Educação Matemática. Assim, temos como problema de pesquisa os métodos de criação que construam software educativo contribuindo de fato na atividade docente e discente de ensinar e aprender.

Desse modo, elencamos algumas etapas metodológicas para alcançar os objetivos delimitados, como pode ser ilustrado no diagrama a seguir:

Diagrama 1 – Esboço da Metodologia




Fonte: o autor.

O "*Resgate Histórico*" consistiu em analisar como foram desenvolvidos alguns software educativos com pesquisas reconhecidas e consolidadas na área da Educação Matemática através de entrevistas online (via Skype) e estudo das literaturas produzidas sobre esses software. Os software escolhidos foram: Casyopée, Function Probe e o Modellus. As justificativas para essas escolhas estão elencadas no Capítulo 5 (Procedimentos Metodológicos), bem como os pesquisadores que foram entrevistados.

A etapa seguinte, "*Análise dos Projetos*" teve por objetivo averiguar as potencialidades e limitações do uso da Engenharia Didático-Informática em projetos de desenvolvimentos de software educativos. Três software foram desenvolvidos utilizando essa metodologia: O Function Studium, o Conics Studium 3D e o Magnitude Studium, que serão detalhados no Capítulo 7.

Já no "*Levantamento e aplicação de implementações*" as informações levantadas nas etapas anteriores foram organizadas e implementadas no modelo de processo da Engenharia Didático-Informática. Foram consideradas as formas de desenvolver software no levantamento histórico bem como as lacunas e sugestões advindas dos pesquisadores ao utilizar a metodologia aqui discutida.

Por último, criamos uma versão atualizada da Engenharia Didático-Informática e do modelo de processo fundamentado nessa metodologia, atualizando questões teóricas e metodológicas, revendo os referenciais que alicerçam essa metodologia e aperfeiçoando-a conforme o objetivo principal dessa pesquisa.

## 1.2 ESTRUTURA DO TEXTO

O primeiro capítulo é a introdução.

No segundo capítulo apresentamos como a Engenharia Didático-Informática foi concebida e aperfeiçoada ao longo dos estudos e pesquisas na área da criação de software educativo: origem, motivação, problemática, características e particularidades.

Em seguida, no Capítulo 3, trazemos a revisão de literatura com reflexões sobre a atual concepção de produtos tecnológicos para o ensino e a aprendizagem, bem como algumas definições importantes para compreender os processos de software.

Os fundamentos teóricos do presente estudo são apresentados no Capítulo 4, em que discutimos as articulações entre a Engenharia Didática e a Engenharia de Software apresentando os principais fundamentos da EDI.

No quinto capítulo expomos a metodologia deste estudo elencando o percurso e a trajetória do desenvolvimento da pesquisa desde sua origem. Apresentamos nesse capítulo a evolução das ideias e como foram pensados os procedimentos da investigação.

Os dois capítulos em sequência, sexto e sétimo, trazem as análises dos dados levantados. No Capítulo 6 apresentamos os resultados das interações e análises realizadas com os pesquisadores que se propuseram a colaborar com esta investigação. Já no Capítulo 7, mostramos os estudos que utilizaram a EDI como aporte metodológico para desenvolver software educativos.

No oitavo capítulo expomos as contribuições das etapas metodológicas realizadas e apresentamos a nova versão da Engenharia Didático-Informática. Apresentamos uma proposta metodológica útil para desenvolver recursos tecnológicos que auxiliam o ensino e aprendizagem.

Em sequência, no Capítulo 9, fazemos as considerações finais do estudo e encaminhamentos para pesquisas futuras. Por último, finalizamos apresentando as referências.

# 2 O BERÇO DA ENGENHARIA DIDÁTICO-INFORMÁTICA

Neste capítulo apresentamos os resultados da Pesquisa de Mestrado (TIBURCIO, 2016) do autor da presente Tese justificando a continuidade do estudo e discutindo as principais considerações, conclusões e encaminhamentos propostos.

Apresentamos aqui a origem da Engenharia Didático-Informática e como ela foi estruturada sob as óticas de alguns pesquisadores que se empenharam em compreender como o a elaboração de software educativos pode ser guiada considerando contribuições sobre o ensino e a aprendizagem aliados aos avanços das tecnologias computacionais.

## 2.1 COMPREENDER O PROBLEMA E A PROBLEMÁTICA DO DESENVOLVIMENTO

A utilização de recursos tecnológicos digitais para auxiliar o ensino e a aprendizagem é uma realidade global: tutoriais, jogos, simuladores, ambientes de ensino e aprendizagem virtuais, linguagem de programação, robótica, entre outros recursos são utilizados com frequência em ambientes de aprendizagem (virtuais ou presenciais).

Apresentamos aqui a problemática da concepção desses recursos: em nossas investigações (TIBURCIO, 2014, 2016; TIBURCIO; BELLEMAIN, 2016, 2018; BELLEMAIN, SILVA; TIBURCIO, 2017) verificamos dificuldades na produção de software educativo. Nota-se a carência de uma engenharia que dê suporte a construção dos programas, visto que a qualidade de muitos deles é questionada devido a forma como são produzidos (BENITTI et al 2005; SANTOS, 2009).

Na revisão de literatura realizada observamos que era primordial a articulação entre as diversas áreas de conhecimento envolvidas na concepção de software educativo – Cognição, Didática, Design, Epistemologia, etc; bem como a conexão entre contribuições teóricas sobre o ensino e a aprendizagem e os referenciais da Engenharia de Software.

De acordo com Bourque e Fairley (2014), um processo de engenharia de software está preocupado com as atividades de trabalho realizadas pelos engenheiros para desenvolver, manter e operar software, como: requisitos, design, construção, testes, gerenciamento de configurações e outros processos da engenharia. Essa definição pode ser utilizada para a criação de tecnologias educativas, contudo carece de especificações que a Educação exige. Assim, definimos a Engenharia de Software Educativo – ESE como o processo de idealização e criação de interfaces ou artefatos com a finalidade de contribuir para a melhoria das relações de ensino e aprendizagem de áreas distintas do conhecimento (TIBURCIO, 2016).

No contexto educativo consideramos que a contribuição da Engenharia Didática (ARTIGUE; 1996, 2002, 2009) para a criação e análise de sequências de ensino, configura-se como importante referencial metodológico auxiliando professores e pesquisadores no processo de compreensão de como se produzem situações de ensino e como a aprendizagem pode ser facilitada a partir dessas em variados níveis educacionais.

Como metodologia de pesquisa, a Engenharia Didática tem como princípio a observação e análise de sessões de ensino, bem como a idealização e criação prévias dessas sessões. No entanto, os aspectos de criação de recursos digitais, tais como software, programas, jogos, entre

outros, perpassam conhecimentos sobre ensinar e aprender, faz-se necessário características mais específicas de produção de tecnologias.

Isso posto, ao se verificar a relevância de relacionar teorias (e construtos teóricos) com os referenciais da Engenharia de Software para desenvolver produtos tecnológicos computacionais que contemplem as minúcias das relações educativas, em pesquisa de Mestrado, Tiburcio (2016) apresentou uma proposta de metodologia de desenvolvimento de software educativo que articula a Engenharia de Software (BOURQUE; FAIRLEY, 2014; SOMMERVILLE, 2011) e a Engenharia Didática (ARTIGUE, 1996, 2002, 2009; ALMOULOUD**;** COUTINHO, 2008; ALMOULOUD; SILVA, 2012; PERRIN-GLORIAN, 2009): a Engenharia Didático-Informática.

A Engenharia Didático-Informática – EDI, emergiu num cenário de tentativas de modelização de processos de criação de software educativo que considere aspectos teóricos e práticos sobre o desenvolvimento de recursos para o ensino e a aprendizagem da Matemática. O termo "Engenharia Didático-Informática" se constitui da percepção em utilizar os procedimentos metodológicos e reflexões teóricas das duas engenharias mencionadas. Utilizamos tal expressão para designar essa metodologia, pois essa fundamenta uma engenharia de software com os contributos teóricos e metodológicos da Engenharia Didática.

A necessidade de articulação entre as engenharias fora verificada ao serem observadas algumas limitações dessas para a criação de software educativo. As limitações da Engenharia Didática foram verificadas no momento em que tal metodologia não contempla, em suas contribuições, a totalidade das exigências para a concepção de SE. A Engenharia de Software, por sua vez, não reúne especificidades que os software educativos necessitam, quando se observa que os modelos padronizados de desenvolvimento foram criados para produtos comerciais, bancários, domésticos, etc.

Com isso, iremos apresentar nesse capítulo como aconteceu a construção da Engenharia Didático-Informática: uma proposta de engenharia de software educativos fundamentada em contribuições teóricas oriundas da Educação Matemática em conexão com a Engenharia de Software, que tem por finalidade a construção de recursos tecnológicos respaldados em contribuições teóricas e potencialidades tecnológicas.

### 2.1.1 Indicativos do desenvolvimento de software

Estudos indicam que muitos recursos tecnológicos digitais, voltados para a Educação, são desenvolvidos centrados nas possibilidades oriundas da tecnologia ou apenas nas teorias sobre a aprendizagem dos conhecimentos. Mesmo quando há a tentativa de articulação entre teorias educativas e possibilidades tecnológicas, ainda assim, a engenharia é frágil (TIBURCIO, BELLEMAIN; RAMOS, 2015; TIBURCIO; BELLEMAIN, 2018).

Segundo Tiburcio (2016), de forma geral, os software educativos são desenvolvidos das seguintes formas:

1. Utilizando ***metodologias padronizadas*** provenientes da Engenharia de Software (não educativos): método em cascata; Desenvolvimento Iterativo e Incremental, método de Prototipagem; método em Espiral, Metodologias Ágeis, entre outros. Nessa vertente, os interessados no projeto (equipes, professores, pesquisadores, entre outros) utilizam as metodologias citadas, sem realizar alterações, para a concepção dos recursos.

2. Realizando uma ***adaptação de metodologias padronizadas***. Nesse caso, as equipes justificam que as metodologias padronizadas auxiliam na criação de software educativos, porém reconhecem que são necessárias adaptações visto que existem diferenças entre produtos tecnológicos para fins educativos e produtos para outros fins.

***3. Integração de áreas***. Nessa vertente as equipes observam as contribuições teóricas, as tendências de ensino e aprendizagem, o que se discute na academia, orientações de documentos oficiais e tentam conectar com as possibilidades tecnológicas atuais. Essa forma de desenvolver software educativos é característica de produções em ambientes de pesquisa, visto que possui caráter experimental, bem como se preocupa com as teorias atuais sobre ensino, aprendizagem, cognição entre outras áreas, para melhoria das relações de ensinar e aprender.

De acordo com Tiburcio e Bellemain (2018), para desenvolver software educativo havia a lacuna de um referencial teórico-metodológico que aliasse as contribuições teóricas das áreas de ensino e de aprendizagem conectadas aos processos da Engenharia de Software. A percepção desses autores gerou investigações na área da Engenharia de Software Educativos, sendo assim concebida a Engenharia Didático-Informática.

A articulação entre teorias educacionais e possibilidades tecnológicas, para a concepção de software educativo, ainda não é uma realidade sistêmica e definida. Existe a carência de uma metodologia robusta com as devidas indicações de procedimentos para criar os software. Assim, a Engenharia Didático-Informática vem sendo desenvolvida a fim de propiciar um processo de desenvolvimento de software educativo que alie contribuições teóricas e potencialidades tecnológicas referente ao ensino e a aprendizagem com tecnologias digitais. Nas reflexões deste cerne, o quadro teórico-metodológico desta investigação foi constituído pelas Engenharias Didática e de Software, observando as contribuições de ambas para a construção dessa metodologia.

## 2.2 O INÍCIO DA BUSCA PELA ARTICULAÇÃO

A primeira tentativa em articular a Engenharia Didática a Engenharia de Software foi a investigação de Ramos (2014): os resultados de sua pesquisa de Mestrado foram expressivos para a estruturação da EDI. O trabalho teve o objetivo de investigar os aportes dos princípios teórico-metodológicos da Engenharia Didática à Engenharia de Software para a elaboração de uma versão digital de um jogo existente: o Bingo dos Números Racionais, criado no contexto do Projeto Rede (GITIRANA, TELES, BELLEMAIN, CASTRO, ALMEIDA, LIMA, BELLEMAIN, 2013).

O Bingo dos Números Racionais (em sua versão em papel) foi construído utilizando os princípios da Engenharia Didática. Ramos (2014), com o objetivo de construir uma versão digital para esse jogo, aproveitou a Engenharia Didática já realizada na versão em papel e reformulou essa engenharia considerando as "necessidades informáticas" para desenvolver a versão digital. A ideia de "Engenharia Didático-Informática" foi inspirada na existência da ED existente para o bingo, verificando quais seriam os aspectos tecnológicos a serem implementados para desenvolver a versão digital.

Ramos (2014) considerou como questão central de pesquisa investigar os aportes da ED à ESE no caso da criação de situações didáticas digitais em matemática (2014, p. 23). De importância tal que motivou a elaboração de uma articulação formal entre ED e ESE, dando início ao aprofundamento da relação entre essas engenharias culminando na concepção da EDI (TIBURCIO, BELLEMAIN; RAMOS, 2015; TIBURCIO, 2016; TIBURCIO; BELLEMAIN, 2018).

A investigação de Ramos (2014) foi a primeira tentativa de articular formalmente a ED a ESE, o que é possível observar em seus objetivos específicos:

> Investigar os princípios teórico-metodológicos da Engenharia de Software Educativos e os aportes potenciais da Engenharia Didática a esses princípios. Investigar os princípios teórico-metodológicos da engenharia didática aplicada à elicitação dos requisitos de um software educativo para a matemática. Aplicar esses princípios à elicitação dos requisitos de uma versão digital do Bingo dos Racionais. Validar esses princípios na concepção da versão do bingo definida (RAMOS, 2014, p. 23).

Ao analisar a possibilidade de utilizar a Engenharia Didática para auxiliar a concepção de software educativos, Ramos (2014) considerou as análises teóricas da ED como relevantes para nortear a concepção de software, bem como identificar os requisitos de ensino e de aprendizagem de objetos matemáticos.

Os objetivos específicos da investigação de Ramos (2014) revelam a natureza da articulação e integração da ED com a ESE, visto que, como já fora citado, a Engenharia Didática por si só não contempla todos os aspectos necessários para a criação de SE, bem como a Engenharia de Software não considera em sua totalidade aspectos didáticos, cognitivos e epistemológicos dos conhecimentos a serem trabalhados com os produtos de software.

A percepção de Ramos (2014) sobre a Engenharia de Software Educativos nos conduziu a considerar a exigência da modelização de um processo de SE ao perceber a importância de aliar teorias educativas a princípios de concepção de tecnologias. Em seu estudo, a autora observou, nas primeiras utilizações de computadores como ferramentas de ensino e aprendizagem, a preocupação em relacionar saberes: “[...] veremos no ensino programado dos anos 1950-1960, fortes interações entre teorias educativas da época, a concepção e o desenvolvimento de tecnologias computacionais” (p. 25). E ainda:

> Ao final, para o sucesso de um projeto, é importante considerar as três dimensões: tecnológica, educativa e do contexto de uso. Nesse sentido, para poder contemplar todas as dimensões da concepção e desenvolvimento de um software educativo, uma solução é de reunir profissionais das diversas áreas de conhecimento envolvidas nesse processo (RAMOS, 2014, p. 29).

Analisando as articulações propostas e a fundamentação teórica de sua investigação, Ramos (2014) utilizou o que classificou como “esquema” para desenvolver o protótipo do software proposto. De acordo com a autora,

> Entendemos a ESE como um campo de conhecimento em desenvolvimento que emerge de uma abordagem transdisciplinar da ciência da computação e simultaneamente de áreas ligadas à educação e propostas metodológicas para abordar a questão da concepção e do desenvolvimento de software educativos (RAMOS, 2014, p. 45).

Ao estudar a engenharia proposta na investigação aqui discutida, percebemos que esta tem fases não explícitas e algumas características não facilitam a sua utilização por usuários que queiram desenvolver produtos de software. Considerando uma abordagem cíclica, não está compreensível onde iniciar o processo e onde concluí-lo, visto que é essencial uma sequência que contenha, no mínimo, início, meio e fim.

A intenção de relacionar a ED com a ESE não apresentou os elementos comuns e também os complementares em ambas as engenharias, além disso, o desenvolvimento do software foi claramente pautado na Engenharia Didática, onde não ficou compreensível quais são as fases do esquema proposto, bem como os processos (etapas) de criação do Bingo dos Racionais nos resultados da pesquisa. Isso que nos motivou a dar continuidade ao que fora sugerido por Ramos (2014).

No entanto, Ramos (2014) conclui sua investigação observando que o processo de integração entre as duas Engenharias (Didática e de Software) permite evitar que os software educativos, de forma geral, não sejam construídos sem o olhar das contribuições teóricas dentro do campo de Pesquisa da Educação Matemática. Também, que não sejam desconsideradas as contribuições dos estudos da Engenharia de Software.

Com isso, demos início a formalização da articulação da Engenharia Didático-Informática: leituras, discussões, pesquisas e sistematizações com o intuito de lançar a primeira versão dessa metodologia de criação de software educativo.

## 2.3 OS LUGARES DAS ENGENHARIAS NA PRODUÇÃO DE SOFTWARE EDUCATIVO

Para compreender a articulação entre as engenharias, faz-se necessário definir alguns conceitos provenientes da área de produção de tecnologias e da Educação Matemática. Apresentamos nos tópicos a seguir alguns conceitos que devem fundamentar uma pesquisa de desenvolvimento de software educativo.

### 2.3.1 Considerações sobre a Engenharia de Software

Bourque e Fairley (2014) definem a Engenharia de Software como a aplicação de uma abordagem sistemática, disciplinada e quantificável para a concepção, operação e manutenção de software. Com essa definição, compreendemos a criação de um software observando que os mesmos são desenvolvidos por etapas/fases. Esses são os processos metódicos que antecedem a criação de fato: o planejamento, a organização, a delimitação de objetivos, como também a análise do produto final após esse entrar em operação.

Desse modo, consideramos a Engenharia de Software como uma disciplina de engenharia cujo objetivo está centrado nos aspectos da produção de software, desde os estágios iniciais até sua manutenção. Em vista disso, a ES: "[...] não se preocupa apenas com os processos técnicos do desenvolvimento de software. Ela também inclui atividades como gerenciamento de projeto de software e construção de ferramentas, métodos e teorias para apoiar a produção de software" (SOMMERVILLE, 2011, p. 5).

As definições apresentadas estão vinculadas com a ideia de procedimento, método ou percurso para a construção de software. Essa ideia é definida na literatura como "processo de software", que é uma abstração, em forma sequencial, de toda a estrutura da engenharia, indicando os procedimentos, do início ao fim, do projeto de construção dos produtos. Bourque e Fairley (2014) acreditam que os processos de software devem ser especificados para facilitar a compreensão, comunicação e coordenação humana; auxiliar o gerenciamento de projetos de software; medir e melhorar a qualidade dos produtos de software de maneira eficiente; apoiar a melhoria de processos e fornecer uma base para o suporte automatizado da execução do processo.

Os procedimentos para a criação de software poderiam ser generalizados e utilizados para a criação de software educativo, todavia devemos considerar as características específicas e particulares dos recursos tecnológicos produzidos para auxiliar o ensino e a aprendizagem. De acordo com Sommerville (2011), existem diversos tipos de sistemas de software, "não faz

sentido procurar notações, métodos ou técnicas universais para a engenharia de software, porque diferentes tipos de software exigem abordagens diferentes" (p. 2). Ainda, de acordo com o autor, os diversos software necessitam ser construídos com uma engenharia base, embora não necessitem das mesmas técnicas de produção.

Dessa forma, acreditamos que para desenvolver software educativo é necessário uma engenharia específica, visto as finalidades, características e particularidades das ações de ensinar e aprender. Assim, escolhemos o referencial da Engenharia Didática para auxiliar nas questões educativas da produção de recursos tecnológicos que contribuem com o ensino e a aprendizagem.

### 2.3.2 A Engenharia Didática e a criação de sequências de ensino

A primeira investigação que traz à tona a associação entre teorias de ensino e aprendizagem com conhecimentos da área da tecnologia considera que, em sua época, não havia uma articulação formal fazendo com que a Engenharia de Software Educativo fosse considerada como uma disciplina possuindo seus métodos e conceitos específicos (BELLEMAIN, 2000).

Segundo Bellemain (2000), "havia um consenso entre os profissionais das várias áreas envolvidas na criação de tecnologias para a aprendizagem, quanto à necessidade de uma interação estreita entre Educação, Didática, Psicologia Cognitiva, Ciência da Computação, etc." (p. 198). Contudo, ressaltava-se a indispensabilidade de engajar reflexões sobre a especificidade da Engenharia de Software Educativos com a finalidade de criar os próprios métodos e conceitos dessa área.

Ao perceber as contribuições efetivas da Transposição Didática (CHEVALLARD, 1985) para a o ensino e aprendizagem, Balacheff (1994 apud BELLEMAIN 2000) propõe uma adaptação e extensão desse referencial teórico inserindo uma dimensão informática, sendo assim classificado como Transposição Informática (BALACHEFF, 1991, 1994). Segundo Bellemain,

> A transposição didática analisa os fenômenos de transformação do saber de referência em saber a ensinar. A introdução da dimensão informática no estudo destes processos não pode preocupar-se apenas com a encenação do saber a ensinar, uma vez que a introdução do computador participa dessa transformação do saber de referência. Portanto, consideramos primordial investigar a questão da transposição informática, não só do ponto de visto da integração das novas tecnologias no ensino (questão que

> já vem sendo analisada), mas também do ponto de vista da produção dessas novas tecnologias (2000, p. 198).

De modo análogo, observamos na Engenharia Didática Clássica (ARTIGUE, 1996, 2009) a carência de um aprofundamento teórico e metodológico específico para tratar das questões de desenvolvimento de tecnologias digitais para o ensino e a aprendizagem. A Engenharia Didática, segundo Artigue (1996), tem por finalidade analisar e propor situações didáticas: "vista como metodologia de investigação, caracteriza-se por um esquema experimental baseado em *realizações didáticas* na sala de aula, isto é, na concepção, na realização, na observação e na análise de sequências de ensino" (p. 196, *grifo do autor*).

Desse modo, percebemos a possibilidade de desenvolver software educativo utilizando o levantamento teórico e os procedimentos metodológicos da Engenharia Didática. Adotamos como hipótese que o embasamento para construção de situações de ensino poderia ser útil para levantar os requisitos e orientar na produção dos recursos tecnológicos a serem desenvolvidos.

As fases da Engenharia Didática foram idealizadas com o objetivo da produção de sequências de ensino, ainda assim, vislumbramos que a organização metodológica desse referencial poderia ser adaptada a fim de contribuir no desenvolvimento de software educativo. Em específico, um aspecto importante da ED é a delimitação de "dimensões", uma vez que, em nossa investigação de Mestrado (TIBURCIO, 2016), percebemos a carência de tratar com detalhes questões tecnológicas.

As dimensões da Engenharia Didática Clássica têm por objetivo direcionar o levantamento teórico que será realizado para fundamentar a sequência didática a ser construída. São elas: dimensão epistemológica – associada às características do saber em questão; cognitiva – relacionada às características cognitiva do público ao qual se dirige o ensino e didática – referente às características do funcionamento do sistema de ensino (ARTIGUE, 1996, p. 200).

Assim, conjecturamos a possibilidade da criação da dimensão informática, bem como toda fundamentação necessária para que o referencial teórico da Engenharia Didática Clássica pudesse ser utilizado para auxiliar o desenvolvimento de software educativos que considerassem teorias de ensino, aprendizagem, cognição, entre outras, em seu alicerce teórico.

A justificativa da associação entre a Engenharia Didática e de Software ocorre ao passo em que se percebe a ausência de elementos para a criação de produtos que contemplem as exigências do ensino e da aprendizagem da Matemática e das tecnologias (aspectos didáticos,

cognitivos, epistemológicos, tecnológicos, entre outros). Observamos a relevância de reunir os elementos pertinentes das duas engenharias: A ED com os elementos de investigação teóricos e experimentais sobre o ensino e a aprendizagem e a ES com a padronização da elaboração de software e métodos de obtenção de requisitos. Isso tudo com a finalidade de construir um processo de desenvolvimento de software educativos que observem os avanços tecnológicos, mas que não despreze os estudos teóricos realizados para o ensino e a aprendizagem.

Bellemain et al (2014) adota como posicionamento epistemológico considerar que "a concepção e o desenvolvimento de software educativos exige a mobilização de uma engenharia didática específica que deve integrar conceitos e métodos da informática" (p. 6). Acrescenta, ainda, que a concepção de um software educativo tem especificidades que o diferenciam de outros, ratificando a necessidade de uma engenharia particular para construir software educativo.

Dessa maneira, construímos o embasamento teórico inicial que fundamentou a Engenharia Didático-Informática considerando os aportes da Engenharia de Software e Engenharia Didática.

## 2.4 PERCURSO METODOLÓGICO DA INVESTIGAÇÃO DE MESTRADO

A metodologia do estudo de Mestrado (TIBURCIO, 2016) do autor dessa Tese, compôs-se da concepção, criação e análise da Engenharia Didático-Informática. Partindo da problemática da criação de SE observamos a necessidade de criar uma metodologia que fosse útil para desenvolver recursos tecnológicos que considerassem os avanços digitais bem como as teorias de ensino e aprendizagem, conforme exposto até aqui.

O estudo tomou como hipótese que a articulação entre elementos da Educação Matemática, por meio da Engenharia Didática, e a Engenharia de Software poderia fornecer produtos tecnológicos que atendam às especificidades dos saberes a serem ensinados e aprendidos. A revisão de literatura constatou que havia a carência de uma metodologia padronizada para desenvolver recursos tecnológicos com as especificações educativas. Em vista disso, demos início a articulação teórica, levantando elementos indispensáveis para desenvolver esses recursos.

Por conseguinte, percebemos a urgência de articular diversos conhecimentos e esses foram viabilizados com a montagem de uma equipe classificada de pluridisciplinar – no sentido de envolver profissionais de diversas áreas para colaborar com a criação desse processo de

software. Após a versão inicial da metodologia ter sido idealizada e criada, iniciamos um estudo de caso para que a Engenharia Didático-Informática pudesse ser analisada em uso.

É válido ressaltar o caráter colaborativo do estudo de caso realizado, visto que foi possível colocar em prática nossa estrutura de elaboração de software ainda quando a mesma estava sendo criada. Em parceria com Silva (2016), que possuía foco no estudo do desenvolvimento de um recurso tecnológico para o ensino e aprendizagem de funções matemáticas, criamos situações de colaboração em que uma pesquisa utilizava dos resultados imediatos da outra, funcionando como uma retroalimentação.

Silva (2016) pretendia criar um software para ser utilizado com estudantes de cursos de Cálculo, especificamente no conteúdo de taxa de variação, com a finalidade de verificar as contribuições do construto tecnológico para a aprendizagem dos saberes em questão. O autor tinha por objetivo geral "a prototipação e validação de um software para a abordagem da taxa de variação de funções, orientado pelo quadro teórico-metodológico da Engenharia Didático-Informática" (p. 16), com isso além de contribuir com a concepção do modelo de processo baseado na EDI, o pesquisador utilizou os resultados iniciais da construção da EDI proporcionando a possibilidade de retornos imediatos.

Dessa forma, o software Function Studium (SILVA, 2016) foi desenvolvido utilizando as premissas da Engenharia Didático-Informática, sendo percebida a importância dessa metodologia para a construção de software educativo. Ao apresentar o Function Studium e suas principais características os responsáveis pelo desenvolvimento, definiram a engenharia que foi utilizada como "uma metodologia que integra princípios teórico-metodológicos da Didática da Matemática à Engenharia de Software, por meio da integração das primeiras etapas da Engenharia Didática à Engenharia de Software" (SILVA, GITIRANA, TIBURCIO; BELLEMAIN, 2017, p. 691). Ainda segundo os autores, esse processo observa os métodos da engenharia de requisitos integrados com uma Engenharia Didática, sendo assim definida como Engenharia Didático-Informática, na qual são contempladas as potencialidades teóricas (do ensino e aprendizagem) e tecnológicas (da computação).

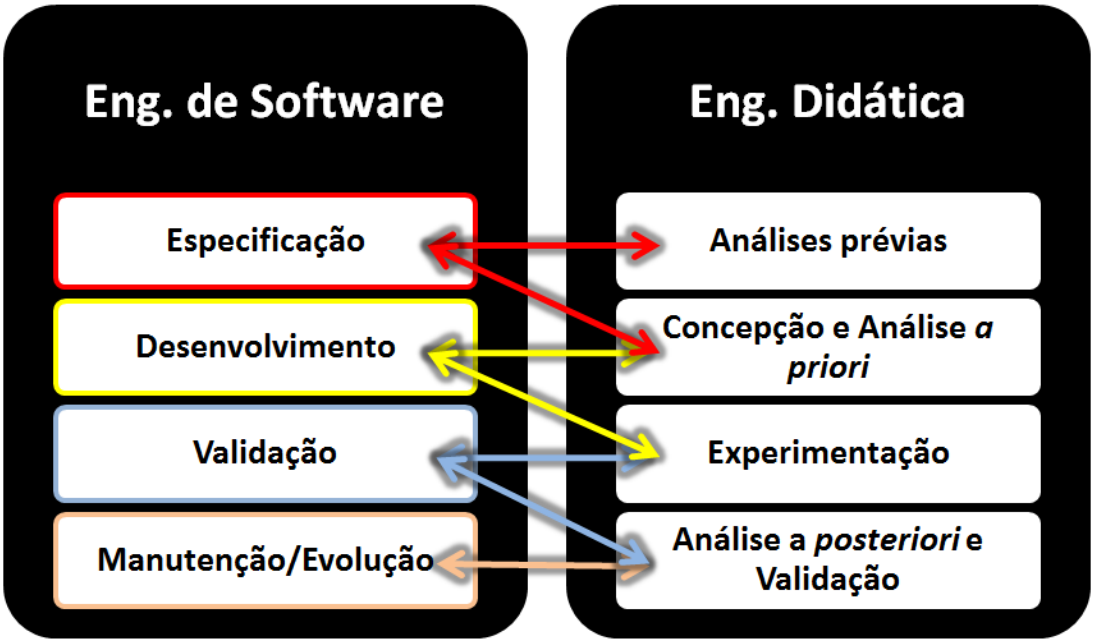
Com a análise dos elementos levantados na experiência de Silva (2016), lançamos a primeira versão do modelo de processo de software da Engenharia Didático-Informática ao mesmo tempo em que foi possível perceber solicitações de evolução do modelo, o que justificou a continuidade da investigação.

## 2.5 PRIMEIROS RESULTADOS DA UTILIZAÇÃO DA ENGENHARIA DIDÁTICO-INFORMÁTICA

Os resultados da investigação de Tiburcio (2016) mostraram que a integração de elementos teóricos sobre o ensino e a aprendizagem da Matemática com as potencialidades tecnológicas atuais é um fator a ser considerado para a elaboração de software educativo. Isso se dá visto que os produtos desenvolvidos com essa perspectiva de articulação podem contribuir para especificação e implementação dos requisitos educativos, assim alcançando as especificidades dos conhecimentos a serem trabalhados com as tecnologias.

Nessa perspectiva, a conexão idealizada entre a Engenharia Didática Clássica (ARTIGUE, 1996, 2009) e a Engenharia de Software (SOMMERVILLE, 2011; BOURQUE; FAIRLEY, 2014) foi realizada. O diagrama a seguir, apresenta a articulação entre as engenharias demonstrando a aproximação de cada uma das fases e como as conectamos.

Diagrama 2 – Comparativo entre as engenharias




Fonte: Tiburcio (2016)

As etapas definidas na Engenharia Didática Clássica são: Análises preliminares, Concepção e Análise a priori, Experimentação e Análise a posteriori e Validação. As etapas da Engenharia de Software servem de fundamentação metodológica para desenvolver os produtos enquanto que as etapas da Engenharia Didática proporcionam reflexões sobre o ensino e a

aprendizagem com o recurso tecnológico em desenvolvimento, configurando assim nossa proposta de complementação entre as engenharias (TIBURCIO, 2016). As relações apresentadas no Quadro 1, entre a Engenharia de Software e a Engenharia Didática foram definidas e apresentadas nos resultados da pesquisa de Mestrado, da seguinte forma:

a) A etapa de Especificação, momento em que se define a tipologia do software, requisitos e características, relaciona-se com as análises iniciais da Engenharia Didática, observando que a metodologia da ED tem por objetivo delimitar variáveis para a compreensão das dificuldades dos estudantes, professores, as características dos conhecimentos, entre outras, a serem trabalhadas na sequência didática, chamados de circunscritores;

b) A etapa de Desenvolvimento leva em consideração o levantamento teórico que foi realizado na Concepção e Análise a priori e inicia-se o processo de experimentação do software. Nesse momento, os requisitos levantados, tanto teóricos quanto tecnológicos, são organizados e articulados para que sejam postos em funcionamento no software em construção. Experimentar o software é uma atividade de pôr em funcionamento em situações de uso com a finalidade de verificar se o que fora proposto em seus objetivos está funcionando.

c) O processo de Validação do software se relaciona com a experimentação da ED e com a Análise a posteriori e Validação. Na ED, a validação das sequências didáticas consiste em confrontar as hipóteses levantadas com a experimentação da sequência. Consideramos que a validação ocorre a partir de duas análises: uma teórica e outra experimental. As situações de uso, funcionalidades e objetivos do SE que foram definidos são confrontados com o que se conseguiu desenvolver na experimentação e, com as conclusões desta investigação, o software é implementado ou o desenvolvimento é concluído. Realizando-se, assim, uma validação interna semelhante ao que ocorre na ED.

d) A etapa de Manutenção/Evolução se relaciona com a Análise a posteriori e Validação no momento em que a análise comparativa fornecida pela ED traz à tona elementos que servem para o aperfeiçoamento e evolução do software

> desenvolvido. Validar o software significa verificar se esse realiza o que se propõe a fazer. Nesta fase, incluem-se os testes, a criação de situações de utilização bem como observações a fim de verificar se o software está em pleno funcionamento. São corrigidos erros e averiguadas se todas as características idealizadas na construção estão em plena performance.

Com o exposto, criamos a estrutura inicial da Engenharia Didático-Informática, apresentando os processos, as etapas, e cada fase para desenvolver software educativo. Tal engenharia fora validada com a criação do software Function Studium e os resultados da sua utilização, bem como as conclusões da pesquisa realizada podem ser acessadas na Dissertação de Mestrado de Silva.

## 2.6 CONSIDERAÇÕES E MOTIVAÇÕES PARA A CONTINUIDADE DO ESTUDO

Como objetivo geral, Tiburcio (2016) pretendia construir, analisar e validar um processo de desenvolvimento de software educativo. Os métodos da Engenharia de Software integrados com a Engenharia Didática sendo observados e articulados em uma análise de potencialidades teóricas e tecnológicas. Foi possível perceber que esse objetivo foi alcançado, no momento em que o processo desenvolvido fora validado no estudo de caso proposto. Ao desenvolver o Function Studium, foi aprimorado o que havia sido proposto na Engenharia Didático-Informática, ainda em construção, chegando a uma primeira versão dessa metodologia.

Os objetivos específicos também foram alcançados, visto que se propôs realizar um panorama do atual desenvolvimento de software educativo identificando potencialidades e limitações. O estudo, também, propôs-se a articular elementos da Educação Matemática e da Engenharia de Software, o que também foi realizado.

Todavia, algumas situações serviram como encaminhamentos futuros para a evolução da EDI. Uma primeira situação que justificou a continuidade do estudo é uma melhor explanação de cada fase da engenharia, bem como delimitações mais explícitas do que deve ser realizado em cada etapa. Estudos posteriores, que utilizaram a EDI para desenvolver software educativo (SIQUEIRA, 2019; SILVA, 2019), expuseram dificuldades em compreender cada parte dessa sistemática. Essas pesquisas revelaram que a descrição de como realizar os procedimentos não foram tão precisas, gerando algumas dificuldades em desenvolver os produtos.

Outra situação que foi considerada como futuro da investigação diz respeito à atualização do referencial teórico. Discute-se, atualmente, a Engenharia Didática de Segunda Geração (ARTIGUE, 2009; ALMOULOUD**;** COUTINHO, 2008; ALMOULOUD; SILVA, 2012; PERRIN-GLORIAN, 2009). São novas perspectivas e há, também, preocupações maiores com a utilização de recursos tecnológicos por professores e estudantes. Consideramos nessa investigação compreender os encaminhamentos da ED de segunda geração.

Outro ponto de atualização é sobre o referencial da Engenharia de Software que também está passando por reformulações para atender as demandas tecnológicas atuais. Na medida em que as tecnologias de informação e comunicação evoluem, igualmente os pesquisadores e estudiosos percebem novas técnicas e métodos para desenvolver os programas, os sites, os jogos, entre outros.

Assim, consideramos que a Engenharia Didático-Informática se constitui de uma metodologia relevante para a elaboração de software educativo, aliando aspectos teóricos sobre o ensino e a aprendizagem com as potencialidades tecnológicas oriundas da Engenharia de Software. Contudo, devem ser consideradas as evoluções sugeridas, o que apresentaremos ao longo desse texto.

# 3 REFLEXÕES SOBRE A PRODUÇÃO DE SOFTWARE EDUCATIVOS

Neste capítulo apresentamos alguns estudos que possuíram interesse em analisar a criação de software para fins educativos.

Destacamos as tentativas de organizar métodos para criar recursos tecnológicos, a fim de auxiliar as relações de ensino e aprendizagem observando as fundamentações teóricas e metodológicas utilizadas pelos pesquisadores. Quanto aos métodos, constatamos que as equipes utilizam princípios da Engenharia de Software (originalmente como foi pensada na Ciência da Computação) e adaptações para realidade educacional.

## 3.1 O ESTADO DA ARTE E A PROBLEMÁTICA DA QUALIDADE

O estado da arte em que se encontra o desenvolvimento de software educativo pode ser resumido em três grupos: 1. a utilização de métodos oriundos da Engenharia de Software - não educativos; 2. a adaptação de métodos; 3. a integração de conhecimentos: teóricos e práticos. Discorreremos aqui sobre esses grupos.

Alguns estudos (BELLEMAIN et al., 2014; BENITTI et al., 2005, SANTOS, 2009) apresentam fundamentos para a construção de jogos, aplicativos, ferramentas de colaboração online, simuladores, micromundos, tutoriais, entre outros. Os resultados dessas pesquisas apresentam possíveis caminhos teóricos e metodológicos para a criação de software. Porém, observando as possibilidades das pesquisas analisadas, concordamos com Tchounikine (2002), no momento em que se observa um paradigma entre a exploração das potencialidades tecnológicas em detrimento das especificidades de aprendizagem.

> Esta visão tecnocêntrica, com base nas possibilidades tecnológicas mais do que nas especificidades de aprendizagem, com máquinas, levou à construção de dispositivos genéricos (disponível a partir de promotores da tecnologia mais relevante por especialistas de EIAH[1]), mas alguns sistemas efetivamente utilizados. O trabalho de investigação dirigida para sistemas cujo valor educativo é claramente reconhecido e que são efetivamente utilizados ou prestes a ser (nós não corremos o risco, propondo uma lista de sistemas que poderia ser exaustiva, mas citamos alguns trabalhos franceses, como o Aplusix, Cabrigéomètre ou Roboteach) não foram concebidos a partir da tecnologia, mas com uma reflexão combinando o estudo dos problemas de capacidades de aprendizagem e de ambientes informatizados. Por isso, estamos interessados em um processo de design de EIAH em significado profundamente pluridisciplinar (TCHOUNIKINE, 2002, p. 8, tradução nossa[2]).

Com isso, ao situar a presente pesquisa na problemática da Engenharia de Software Educativo, faz-se necessário compreender o que é um processo de desenvolvimento e os modelos que existem para a construção de software. Desse modo, percebe-se em Sommerville (2011, p. 5) que a engenharia de software "é uma disciplina da engenharia que se ocupa de

[2]Texto original: Cette vision technocentrée, fondée sur les possibilités technologiques plus que sur les spécificités de l'apprentissage avec des machines, a conduit à la construction de dispositifs génériques (proposés par les promoteurs de la technologie concernée plus que par les spécialistes de l'EIAH) mais a peu de systèmes effectivement utilisés. Les travaux de recherche ayant conduit à des systèmes dont la valeur pédagogique est clairement reconnue et qui sont effectivement utilisés ou en passe de l'être (on ne se risquera pas ici à proposer une liste de systèmes qui ne pourrait être exhaustive, mais on peut, pour citer quelques travaux Français, penser à Aplusix, Cabrigéomètre ou Roboteach) n'ont pas été conçus à partir de la technologie, mais par une réflexion mêlant l'étude des problématiques de l'apprentissage et des possibilités informatiques. Il s'agit donc bien ici de s'intéresser au processus de conception des EIAH dans son acception profondément pluridisciplinaire.

todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a manutenção desse sistema depois que ele entrou em operação".

Apresentada a definição, compreendemos a criação de um software observando não apenas o produto final, mas a atenção para os processos que antecedem o trabalho de criação, ou seja, um planejamento prévio, uma preparação de como o produto será desenvolvido, como também a análise do produto final após o mesmo entrar em operação. De acordo com o IEEE[3] (apud PRESSMAN, 2006, p. 17), engenharia de software é a "aplicação de uma abordagem sistemática, disciplinada e quantificável, para a concepção, operação e manutenção do software; isto é, a aplicação de engenharia ao software".

Ambas as definições supracitadas estão diretamente relacionadas com a ideia de procedimento, metodologia ou percurso para a construção de software, o que a literatura define como "processo de software". Segundo Sommerville (2011), "um processo de software é um conjunto de atividades e resultados associados que geram um produto de software. Essas atividades são, em sua maioria, executadas por engenheiros de software" (p. 7).

Observa-se a importância dos processos de software na medida em que o uso de metodologias e procedimentos podem facilitar o trabalho da equipe envolvida no projeto de construção, seja na análise das exigências dos usuários, na observação do funcionamento, na manutenção, entre outros. A engenharia para produção de software, quando bem fundamentada, tem por objetivo garantir a qualidade dos produtos a serem desenvolvidos. No entanto, ao utilizar o termo "usuários" nota-se a pluralidade de sujeitos que a concepção de software educativo possui: professores, alunos, equipe pedagógica, etc. Concluímos que um dos fatores de grande notoriedade na distinção entre software educativo e não educativo são as necessidades de cada tipo de usuário em relação à sua utilização.

Um software para auxiliar o ensino da Geometria, especificamente para o trabalho com áreas de superfícies, por exemplo, tem características diferentes de outro que sirva para o ensino da taxa de variação de funções na Álgebra, visto que os conhecimentos e as representações que serão projetadas carecem de sistemas de representações diferentes. Um ambiente virtual de aprendizagem difere de um jogo digital, tanto em seus recursos quanto na forma de aquisição do conteúdo, ou seja, é necessário observar os usuários, os objetivos, os conhecimentos, os paradigmas de ensino e de aprendizagem, entre outros fatores a serem considerados.

[3] Institute of Electrical and Electronics Engineers - Instituto de Engenheiros Elétricos e Eletrônicos.

### 3.1.1 Como os software educativos são desenvolvidos?

Partindo da problemática da criação de software educativo, analisamos pesquisas que versassem sobre tal temática. A finalidade era de compreender como se utilizam as metodologias de desenvolvimento atualmente e discutir as sugestões, orientações e o que ainda não foi pesquisado.

Costa e Costa (2013), com o objetivo de discutir um processo de construção de software educativo, "simples, iterativo e incremental", apresentam uma proposta de engenharia que tem por base o Design Centrado no Utilizador – DCU, aliado à prática das metodologias ágeis, configurando assim um método híbrido, segundo os autores. As metodologias de criação de software consideradas ágeis se baseiam em desenvolvimento simples e prático, seguindo princípios metodológicos definidos (explicitaremos alguns princípios nos capítulos seguintes).

De acordo com Costa e Costa (2013, p. 1), o processo desenvolvido é "extremamente útil para garantir a qualidade dos pacotes de software educativo para a Matemática". Os autores acreditam que os recursos tecnológicos contribuem para a aprendizagem, porém, quando analisam a inserção das novas tecnologias nos ambientes de ensino e aprendizagem, questionam a qualidade dos software disponíveis e acreditam que o processo de engenharia de software educativo é relevante para que se tente garantir uma qualidade mínima.

> Contudo, as tecnologias como os tablets, computadores, entre outras, são artefatos potencializadores de aprendizagens significativas, levando os profissionais da educação a procurar conteúdos generalistas ou específicos. Por diferentes fatores, entre eles, a falta de competências de análise e avaliação por parte dos profissionais da educação no que confere aos requisitos técnicos e pedagógicos de determinado software educativo, leva-nos a concordar com Ramos e colaboradores (2005) que a condicionante atrás referida leva, por um lado, à aquisição de conteúdos de qualidade duvidosa e por outro ao pouco cuidado dos editores na garantia da sua qualidade. Desta forma, o desenvolvimento de software educativo de qualidade implica uma avaliação formativa dos protótipos concebidos, pelas equipes, durante o processo de desenvolvimento (COSTA; COSTA, 2013, p. 1).

Dessa maneira, percebe-se que a utilização de teorias de ensino e aprendizagem pode auxiliar a criação de produtos tecnológicos. Os autores, também, acreditam que os procedimentos de elaboração de software educativos aliados às teorias de aprendizagem colaboram com a qualidade do produto a ser gerado. De acordo com eles, os processos não são

capazes de resolver todos os problemas, mas pressupõem a garantia de qualidade dos produtos (COSTA; COSTA, 2013).

Em pesquisas anteriores, os autores acreditavam que formando uma equipe com profissionais técnicos (designers e programadores) e professores da área a que se destinava o software seria suficiente para construir produtos de qualidade. Porém, após o desenvolvimento de dois sistemas, restou claro que existe certa volatilidade dos requisitos educacionais – no sentido de que compreender quais características os recursos devem conter para atender as demandas do ensino e aprendizagem não é uma tarefa simples de ser cumprida. Esses requisitos são fatores de difícil quantificação e requerem boa compreensão do domínio do problema a ser resolvido com soluções tecnológicas (COSTA; COSTA, 2013).

Ao observarmos as dificuldades de execução da proposta de construção de SE apresentada pelos autores, verificamos a importância de acessar as peculiaridades dos saberes a serem trabalhados pelas tecnologias pensadas. Percebermos, assim, a importância do levantamento dos requisitos de forma sistêmica e ancorada em aportes teóricos sobre ensino e aprendizagem, criando na EDI, ainda na Pesquisa de Mestrado (TIBURCIO, 2016), uma etapa específica para que se realize um levantamento teórico robusto. Isso a fim de mapear quais são as contribuições teóricas atuais na área de conhecimentos específicos que os software a serem produzidos irão trabalhar.

Costa e Costa (2013), com o intuito de superar o problema com os requisitos educacionais, observam que a utilização do Design Centrado no Usuário é útil "para descrever os processos de um projeto em que os utilizadores finais têm influência na forma como este é conduzido" (p. 5). Para eles, as contribuições do DCU estão relacionadas com a obtenção direta de informações pelo usuário final.

> Alguns métodos do Design Centrado no Utilizador sondam os utilizadores sobre as necessidades que estes possuem em determinada área educacional, envolvendo-os em partes específicas do processo de desenvolvimento. Por outro lado, existem métodos em que os utilizadores têm uma maior presença, integrando a equipe, isto é, são envolvidos como elementos durante todo o processo (COSTA; COSTA, 2013, p. 5).

Percebemos, em Costa e Costa (2013), alguns indicativos para nortear a construção de software educativo e verificamos que a Engenharia Didático-Informática aparece como uma solução para possíveis lacunas. Em relação à utilização do DCU, verificou-se que a principal

fonte de informações para a análise e obtenção dos requisitos se concentra nos utilizadores dos produtos a serem desenvolvidos. Isso se configura como importante, porém, para a concepção de software educativo, a consulta à pesquisas teóricas sobre o ensino e a aprendizagem, é importante visto que esta pode auxiliar apresentando possibilidades de como ensinar e como aprender com recursos tecnológicos.

Com os resultados da Pesquisa de Mestrado (TIBURCIO, 2016), percebemos o quão prejudicial seria obter os requisitos dos software utilizando como fonte principal apenas os conhecimentos dos usuários, a saber

> Com as discussões realizadas até aqui inferimos que a perspectiva de desenvolvimento de ambientes micromundos tomando por base estudantes, dos diversos níveis, como utilizadores, seria prejudicada caso os requisitos do ambiente fossem adquiridos apenas com a opinião dos utilizadores. Dada a complexidade da configuração dos micromundos, não seria viável obter os requisitos através dos usuários finais, visto que estes ambientes necessitam de situações didáticas para efetivar o aprendizado do conhecimento a ser desenvolvido (TIBURCIO, 2016, p. 29).

Assim, considerando a perspectiva de construção de SE e tomando por base professores e alunos como utilizadores, torna-se difícil o acesso aos requisitos do ambiente através da inserção dos utilizadores, pois dada a complexidade da configuração desses recursos, não seria uma tarefa simples obter os requisitos através apenas da consulta aos usuários finais. Com isso, a EDI auxilia no acesso aos requisitos do software considerando as pesquisas nas dimensões Didática, Epistemológica, Cognitiva e Informática; a fim de explicitar todas as contribuições que essas áreas apresentam sobre o domínio que o software a ser produzido versará. As formas de obtenção de requisitos serão detalhadas nos próximos capítulos.

Outra pesquisa, também na perspectiva da construção de SE, apresenta uma experiência de engenharia de objetos de aprendizagem. Lapolli, Motta, Oliveira e Cruz (2009) consideram importante a utilização das metodologias padronizadas com a finalidade de atender às demandas organizacionais e pedagógicas, porém, acreditam que as metodologias ditas “tradicionais” de desenvolvimento não satisfazem as exigências das propostas atuais, segundo os autores,

> A abordagem de desenvolvimento tradicional já não satisfaz as necessidades das propostas educacionais atuais, principalmente as de domínio complexo, pois nem sempre contribuem para a aprendizagem final. Portanto, é preciso pesquisar

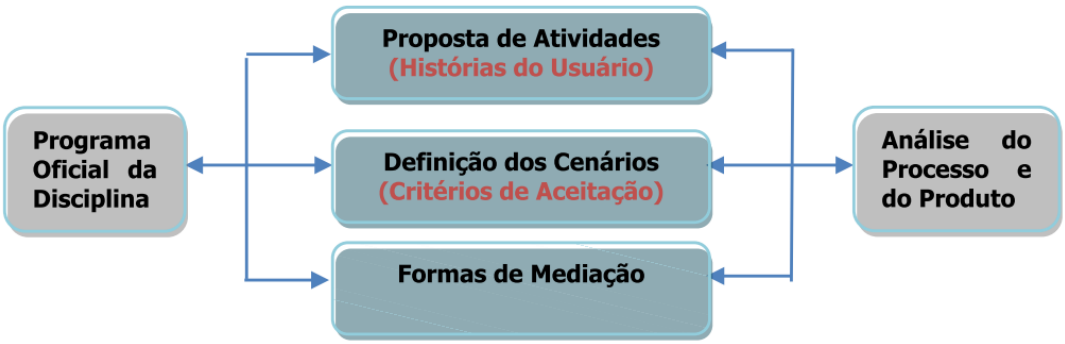
> abordagens de desenvolvimento de recursos educacionais mais adequados a prática pedagógica (LAPOLLI et al., 2009, p. 250).

As metodologias tradicionais são, também, chamadas de “pesadas” ou orientadas a documentação (SOARES, 2004). Essas metodologias surgiram em um contexto de concepção de software muito diferente do atual – quando foram desenvolvidas, o custo de fazer alterações e correções era muito alto visto que o acesso aos computadores era limitado e não existiam ferramentas de apoio ao desenvolvimento do software como temos hoje. Por esse motivo, o software era planejado e documentado antes de ser implementado.

Lapolli et al (2009), em contrapartida aos métodos tradicionais de desenvolvimento, utilizou o Behaviour-Driven Development - BDD (desenvolvimento orientado a comportamento), um processo considerado dentro dos princípios ágeis que engloba a análise de requisitos, até a concepção do software. De acordo com eles, o BDD é a “a união de várias práticas consideradas ágeis e úteis na construção de software, cuja ênfase está nas funcionalidades de alto valor e na redução dos custos de mudança por meio da identificação do que de fato está sendo testado” (LAPOLLI et al., 2009, p. 253).

Baseados nessa metodologia, os autores chegaram ao modelo apresentado no diagrama a seguir:

Diagrama 3 – Modelo de projeto instrucional




Fonte: Lapolli et al. (2009)

Em relação à obtenção de requisitos, foi possível constatar que a consulta sobre a aprendizagem dos saberes técnicos foi realizada através de entrevistas. Percebeu-se que documentos, teorias de aprendizagem, cognitivas ou outras, não foram observadas. Os autores definem sua busca da seguinte forma:

> A fim de identificar os requisitos e funcionalidades que deveriam ser abordados nesse OA, realizamos entrevistas semi-estruturadas com instrutores e alunos no Instituto de Controle do Espaço Aéreo (ICEA), em setembro de 2007, onde apuramos as dificuldades em se construir cognitivamente um modelo do momento em que os conceitos abstratos de Meteorologia influenciam significativamente nos procedimentos operacionais. Na elaboração da proposta de atividades contamos com a colaboração de um controlador de tráfego aéreo com uma gama de conhecimentos, habilidades, convicções e conceitos adquiridos ao longo dos 20 anos de atividade profissional. (LAPOLLI et al., 2009, p. 257).

Do trabalho analisado, conclui-se que a utilização das metodologias ágeis se configura como recurso utilizado rotineiramente para a construção de software, porém a análise dos requisitos não levou em consideração as contribuições das pesquisas já realizadas sobre a aprendizagem dos saberes que estavam envolvidos. À vista disso, percebe-se a relevância da utilização da Engenharia de Requisitos. Com essa metodologia, é possível identificar de forma metódica as necessidades dos usuários e do sistema que será especificada na fundamentação teórica.

Assim, apresentamos no capítulo a seguir os referenciais teóricos que fundamentam a Engenharia Didático-Informática como metodologia de desenvolvimento de software educativo que alia contribuições teóricas às tecnológicas.

# 4 ALICERCE TEÓRICO DA ENGENHARIA DIDÁTICO-INFORMÁTICA

Estruturamos a fundamentação teórico-metodológica apresentando, inicialmente, quais foram as necessidades que motivaram a continuidade da investigação iniciada no Mestrado (TIBURCIO, 2016), visto que algumas lacunas e indicativos de aperfeiçoamento da EDI foram sugeridas nas considerações finais e nas discussões posteriores ao término da pesquisa.

Apresentamos, nesse capítulo, os princípios das Engenharia de Software e Engenharia Didática que, em uma perspectiva de união, foram organizados para fundamentar a EDI.

## 4.1 NECESSIDADES OBSERVADAS PARA O APERFEIÇOAMENTO DA EDI

Com a utilização da EDI, obtivemos uma primeira resposta às carências observadas nas formas de desenvolver software educativo. As análises teóricas propostas nessa engenharia norteiam a criação de produtos para atender características específicas dos conhecimentos que serão trabalhados nos recursos tecnológicos com fins didáticos.

Em conformidade aos resultados da Pesquisa de Mestrado (TIBURCIO, 2016), consideramos que a construção de software educativo deve pressupor, além dos aportes tecnológicos, elementos oriundos das reflexões a respeito do ensino e da aprendizagem com tecnologias. Isso pois, quando o cenário atual de desenvolvimento de software educativo foi observado, percebeu-se a urgência de aprimoramento do modelo de processo de software justificando, inicialmente, a realização deste estudo. Sendo assim, fez-se necessário conhecer as contribuições atuais das pesquisas na área da Educação Matemática, visto que o modelo foi aperfeiçoado para a concepção e construção de produtos para o ensino e a aprendizagem da Matemática.

Tiburcio (2016) acredita que é importante contemplar os pressupostos da Engenharia de Software com as pesquisas da Educação Matemática. O autor apresenta as relações que podem ser estabelecidas entre a Engenharia Didática – ED (ARTIGUE, 1996, 2002, 2009; PERRIN-GLORIAN, 2009) e a Engenharia de Software (BOURQUE; FAIRLEY 2014; SOMMERVILLE, 2011) para construção de SE.

## 4.2 PREMISSAS DA ENGENHARIA DE SOFTWARE

Quando consideramos um processo de engenharia, de forma genérica, pensamos em atividades de construção, criação, um conjunto de materiais, objetos e pessoas para iniciar e concluir algo. Bourque e Farley (2014) apresentam uma definição interessante de Engenharia de Software fazendo analogias com diversas áreas: “um processo de engenharia consiste em um conjunto de atividades inter-relacionadas que transformam uma ou mais entradas em saídas enquanto consomem recursos para realizar a transformação”[4] (p. 8, tradução nossa).

Para os autores, muitos dos processos usuais das disciplinas tradicionais de engenharia (elétrica, mecânica, civil, química, entre outras) se preocupam em transformar energia e entidades físicas de uma forma para outra, como em uma barragem hidrelétrica que transforma

[4] Texto original: An engineering process consists of a set of interrelated activities that transform one or more inputs into outputs while consuming resources to accomplish the transformation.

energia potencial em energia elétrica ou em uma refinaria de petróleo que usa processos químicos para transformar petróleo em gasolina. Em específico na Engenharia de Software, os processos possuem a preocupação com as atividades de trabalho realizadas pelos engenheiros para desenvolver, manter e operar software; como requisitos, design, construção, teste, gerenciamento de configuração e outros processos da engenharia (BOURQUE; FARLEY, 2014).

Para Sommerville (2011), o mundo moderno não pode existir sem software. Em diversos setores importantes, como a manufatura e a indústria, as atividades são totalmente informatizadas, assim como o sistema financeiro. A área de entretenimento, incluindo a indústria da música, jogos de computador, cinema e televisão, faz uso intensivo de software. Ainda, segundo o autor, existe uma ideia generalizada de que software é simplesmente outra palavra para designar programas de computador. Contudo, ao se referir a Engenharia de Software, não se trata apenas dos programas em si, mas de toda a documentação associada e dados de configurações necessários para fazer os programas operarem corretamente.

Com isso, consideramos, de acordo com Sommerville (2011) e Bourque e Farley (2014), que a Engenharia de Software possui foco em todos os aspectos da produção de software, desde os estágios iniciais da especificação do sistema até sua manutenção, quando o sistema já está sendo utilizado.

## 4.3 COMPOSIÇÃO DAS EQUIPES E MÉTODOS ÁGEIS

Antes de iniciar qualquer projeto de desenvolvimento de software existe a estruturação da equipe que atuará na criação. É necessário, assim, gerir a organização a fim de que os envolvidos compreendam as atribuições individuais e coletivas. Alguns estudos (TIBURCIO; BELLEMAIN, 2018; TCHOUNIKINE, 2004) destacam a importância da constituição de equipes pluridisciplinares e transdisciplinares para o trabalho de criação de software educativo.

O sentido pluridisciplinar traz a perspectiva de articular profissionais de áreas distintas para que cada um possa contribuir com a concepção do software; designers, programadores, usuários, entre outros, contribuindo com os conhecimentos de suas áreas para conceber um produto de qualidade. Já o sentido transdisciplinar traz a perspectiva de que os saberes das áreas envolvidas não sejam apenas reunidos, mas sejam integrados, numa perspectiva de articulação para viabilizar a criação de novos conhecimentos.

Para Tchounikine (2004), a elaboração de tecnologias digitais para o ensino e a aprendizagem é um campo científico que está na interseção de diferentes disciplinas, o que representa um conjunto de desafios únicos. É necessário abordar esse campo específico com uma perspectiva de produzir novos saberes. Para a produção de software com intenções educativas, consideramos a necessidade de a equipe ser constituída por profissionais como Professores, Pesquisadores, Psicólogos, Pedagogos, entre outros, visto que trazem significativas contribuições para as questões de ensino e aprendizagem, além dos Designers, Engenheiros de Software, Programadores, etc.

Tiburcio (2016) observa que o fato de reunir diferentes profissionais não é suficiente para a criação de produtos que atendam às exigências específicas dos conhecimentos envolvidos. A articulação e integração entre a equipe e os saberes devem ser promovidas a partir de uma metodologia de trabalho definida e a perspectiva transdisciplinar é a que traz mais benefícios para a construção de software educativo. Sendo assim, consideramos que alguns dos princípios da Metodologia Ágil contribuem para organizar o trabalho dos participantes e outros encaminhamentos da gestão e criação de software educativo. Em síntese, essas metodologias norteiam o trabalho, as interações, o envolvimento, bem como como cada profissional pode contribuir com a criação efetiva do software.

Os métodos ágeis possuem também características que orientam o ciclo de vida do software na medida em que seus princípios auxiliam a gestão das atividades, a organização da equipe e do desenvolvimento dos produtos (TIBURCIO, 2016). Esses métodos têm como uma das suas premissas considerar a participação do usuário do software que será desenvolvido no processo de criação. Essa característica, em particular, fez-nos compreender a importância da atuação ativa dos usuários des software educativos. Referente aos ambientes micromundos e simuladores, a utilização se dá, na maioria das vezes, quando o professor ou pesquisador cria situações de utilização para que os estudantes ou usuários de forma geral, alcancem os objetivos de aprendizagem. Sendo assim, a participação dos professores, pesquisadores e estudantes (como usuários) da criação de software educativo é essencial.

Além de aproximar os usuários do processo de concepção e elaboração, existem outros princípios norteadores para a utilização das metodologias ágeis, que foram criados por um grupo de pesquisadores com a finalidade de orientar um novo paradigma de criação de software, chamado de Manifesto Ágil. Sommerville (2011) resume esses princípios e os apresentamos na Tabela 1.

Tabela 1 – Os princípios dos métodos ágeis

| PRINCÍPIOS | DESCRIÇÃO |
|---|---|
| Envolvimento do cliente | Os clientes devem estar intimamente envolvidos no processo de desenvolvimento. Seu papel é fornecer e priorizar novos requisitos do sistema e avaliar suas iterações. |
| Entrega incremental | O software é desenvolvido em incrementos com o cliente, especificando os requisitos para serem incluídos em cada um. |
| Pessoas, não processos | As habilidades da equipe de desenvolvimento devem ser reconhecidas e exploradas. Os integrantes devem desenvolver suas próprias maneiras de trabalhar, sem processos prescritivos. |
| Aceitar mudanças | Deve-se ter em mente que os requisitos do sistema vão mudar. Por isso, projete o sistema de maneira a acomodar essas mudanças. |
| Manter a simplicidade | Focalize a simplicidade, tanto do software a ser desenvolvido quanto do processo de desenvolvimento. Sempre que possível, trabalhe ativamente para eliminar a complexidade do sistema. |

Fonte: Adaptado de Sommerville (2011, p. 40)

Com os princípios listados, percebemos a pertinência de incorporá-los na criação de software educativo. A aproximação dos usuários no processo, a consideração dos requisitos levantados pelos usuários, as habilidades e competências da equipe (transdisciplinar) sendo utilizadas durante o trabalho de concepção e criação, a mudança constante de requisitos e a simplicidade para a produção são pilares agregados na Engenharia Didático-Informática.

Na literatura da Engenharia de Software, os usuários finais de software gerais (não educativos), bem como os que investem no desenvolvimento, são chamados de “clientes”. Considerando a utilização de recursos digitais com finalidade educacional, não utilizamos a nomenclatura cliente para designar os usuários, visto que não há uma relação comercial por parte dos estudantes e professores ao adquirir esses recursos.

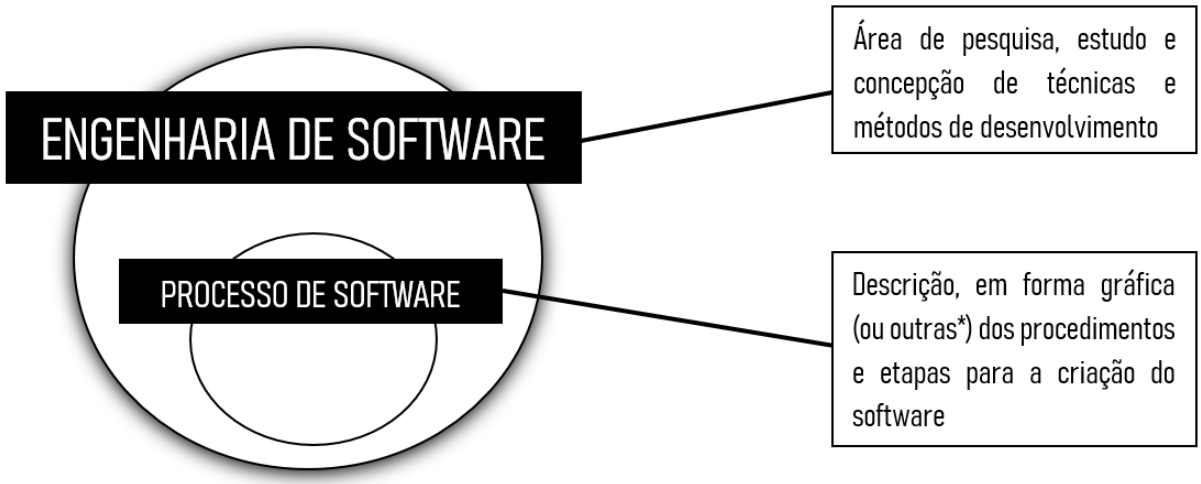
Em sua grande maioria, no âmbito público, os software são adquiridos pelos órgãos regulamentadores (Ministério da Educação, Secretarias Estaduais e Municipais de Educação, entre outros). Já em escolas privadas, as gestões e administrações escolares adquirem os programas e os utilizam com os seus professores, estudantes e equipes de forma geral. Dessa

forma, chamamos de “usuários” os sujeitos que utilizam as tecnologias digitais educativas e não de “clientes”.

## 4.4 MODELOS DE PROCESSO

Uma definição a ser considerada referente à criação de software é a de “modelo de processo”: são modelizações abstratas com todas as etapas para iniciar e concluir a criação dos produtos, também chamadas de ciclo de vida do software. Para melhor compreensão, o Diagrama 4 situa, dentro de uma metodologia de desenvolvimento de software, o lugar dos processos de software.

Diagrama 4 – Engenharia e processo de software




*abstrata/descritiva/ilustrativa

Fonte: o autor.

Existem modelos de processo que são, corriqueiramente, utilizados e bons produtos são desenvolvidos com eles. Ao longo do tempo, os modelos que foram utilizados com maior frequência foram: O *modelo em cascata*, que considera as atividades fundamentais do processo de especificação, desenvolvimento, validação e evolução, e representa cada uma delas como fases distintas; O *desenvolvimento incremental*, que intercala as atividades de especificação, criação e validação e a *engenharia de software orientada a reuso*, que é uma abordagem baseada na existência de um número significativo de componentes reusáveis, o processo de elaboração do sistema concentra-se na integração desses componentes em um sistema já existente em vez de desenvolver um sistema a partir do zero. A Tabela 2 resume alguns dos modelos rotineiramente utilizados e suas características.

Tabela 2 – Vantagens e desvantagens dos métodos de desenvolvimento de software

| Designação | Vantagens | Desvantagens |
|---|---|---|
| Método em Cascata (anos 70) | Funciona bem para equipes tecnicamente mais fracas. É produzida documentação em cada fase. | Estrutura rígida e procedimentos inflexíveis;<br>Não reconhece a necessidade de retornar às fases anteriores e corrigir erros;<br>Versão operacional do sistema apenas disponível numa fase avançada. |
| Método Prototipagem (anos 80) | Apresenta resultados sem necessitar de toda a informação no início do projeto;<br>É útil quando os requisitos mudam rapidamente e o utilizador está relutante em aceitar um conjunto de requisitos;<br>Ajudam a definir os requisitos. | Pode levar a falsas expectativas, isto é, o utilizador muitas vezes pensa que o software está terminado;<br>Pacotes de software pobres devido ao objetivo principal do método, o desenvolvimento rápido;<br>É impossível determinar com exatidão o tempo que o projeto vai demorar a ser desenvolvido;<br>Não há forma de saber o número de iterações que serão necessárias. |
| Método em Espiral (anos 80) | As iterações iniciais do processo de desenvolvimento são menos dispendiosas, permitindo que as tarefas de maior risco sejam concebidas com menor custo;<br>Componente de análise de risco disponibiliza uma ferramenta de medida. | Aplicação complexa, implicando muitos anos de prática para aplicar o método com eficácia. |
| Métodos Ágeis (anos 90) | Os utilizadores (clientes) estão envolvidos ativamente durante o projeto. | Não são adequados para pacotes de software grandes, estáveis e com requisitos bem definidos;<br>Os pedidos informais para melhorias, após cada fase podem gerar confusão. |

Fonte: adaptado de Costa (2012).

De acordo com Sommerville (2011), esses modelos são ditos genéricos, mas não são descrições definitivas, visto que alterações podem ser feitas e um modelo se apoiar em elementos de outros. Na Engenharia Didático-Informática, temos como base, fundamentados na Engenharia de Software, as atividades elementares de processo de software, porém articulando-as sem que essas sejam realizadas como uma prescrição ou receituário. Esses procedimentos estão ilustrados no Diagrama 5.

Diagrama 5 – Atividades elementares da Engenharia de Software – Ciclo de vida


Fonte: o autor

Na etapa de *Especificação*, define-se a tipologia do software, requisitos e características, ou seja, as funcionalidades do software e as restrições em sua operação. O *Desenvolvimento* é a parte prática do processo, momento em que o software é produzido de modo que atenda a suas especificações.

O processo de *Validação* do software é realizado a fim de verificar se esse satisfaz o que se propôs a fazer em seus objetivos e finalidades. Nesta etapa acontecem os testes, a realização das situações de utilização propostas bem como avalia-se o modo que o software reage em funcionamento.

Por conseguinte, a etapa de *Manutenção/evolução* se relaciona à etapa que tem por objetivo dar continuidade aos estudos do software, verificando possíveis melhorias para que novas versões sejam lançadas. Nas seções seguintes, iremos detalhar cada etapa e quais conceitos utilizamos na Engenharia Didático-Informática.

### 4.4.1 Especificação

Também conhecida como engenharia de requisitos, a especificação de software tem por objetivo delimitar características e funcionalidades dos produtos. Essa engenharia se configura como aporte para que se delimitem os atributos do software e dos sistemas.

A engenharia de requisitos, como área de conhecimento, refere-se ao processo de obtenção, análise, especificação e validação de requisitos de software, bem como ao gerenciamento de requisitos durante todo o ciclo de vida do produto de software. Essa área é

amplamente reconhecida entre pesquisadores e profissionais no momento em que esses consideram que os projetos são criticamente vulneráveis quando as atividades relacionadas aos requisitos são mal executadas. Os requisitos expressam as exigências e restrições impostas a um produto de software que contribuem para a solução de algum problema do mundo real (BOURQUE; FARLEY, 2014).

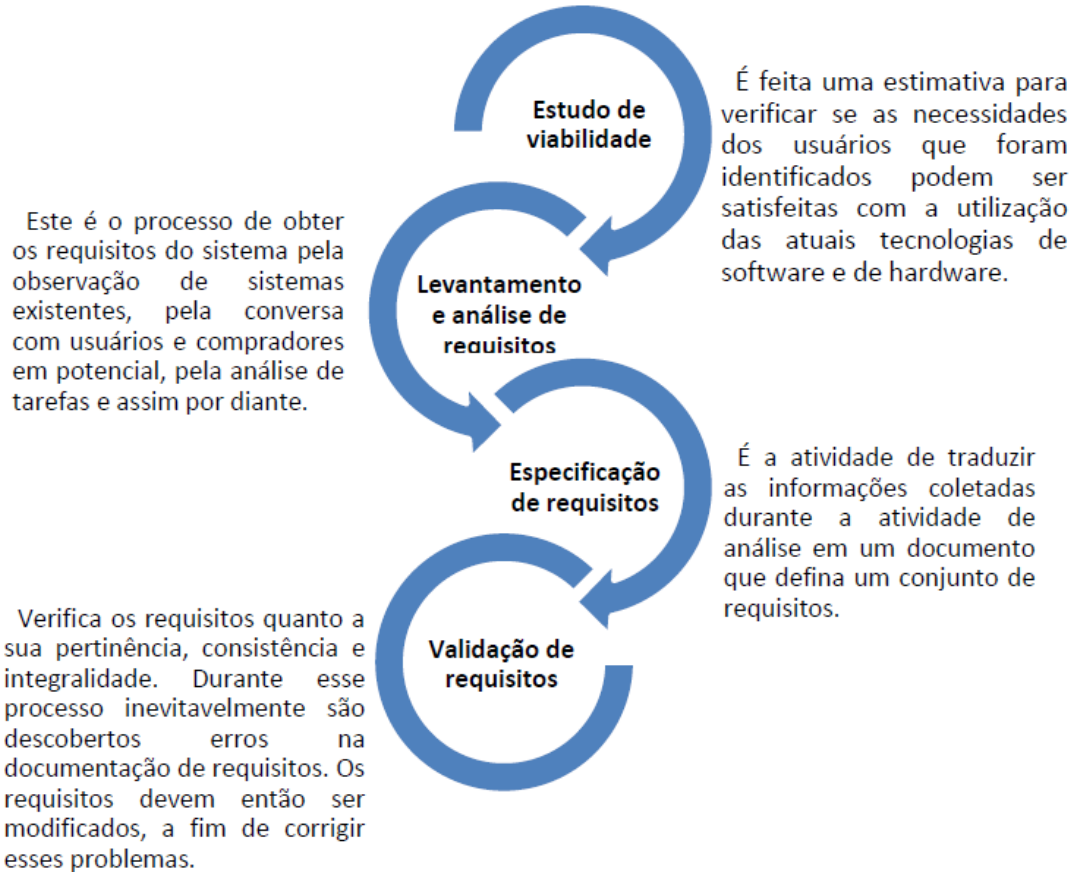
Nuseibeh e Easterbrook (2000) consideram que a principal medida de sucesso de um sistema de software é o grau em que se encontra com o propósito para o qual foi concebido. Para os autores, a engenharia de requisitos é o processo de descobrir a finalidade, através da identificação das partes interessadas e suas necessidades, e documentá-las de uma forma passível de análise, comunicação e posterior implementação.

Alguns autores fazem separações dos requisitos para melhor compreendê-los e buscá-los no processo chamando de elicitação de requisitos. Para Bourque e Farley (2014), os requisitos são classificados em *funcionais* e *não funcionais*. Os funcionais descrevem as funções que o software deve executar; formatar algum texto ou modular um sinal, por exemplo. Às vezes, são conhecidos como capacidades ou recursos. Já os não funcionais são os que agem para restringir a solução. Requisitos não funcionais são, algumas vezes, conhecidos como restrições ou requisitos de qualidade.

Em contrapartida, Sommerville (2011) sugere uma classificação diferente: os *requisitos do usuário* e os *requisitos do sistema*. Os do usuário são declarações, em linguagem natural e também em diagramas, sobre as funções que o sistema deve fornecer e as restrições sob as quais deve operar; os requisitos do sistema, estabelecem detalhadamente as funções e as restrições de sistema. O documento de requisitos de sistema, algumas vezes chamado de especificação funcional, deve ser preciso.

Iremos considerar nessa pesquisa os requisitos segundo Sommerville (2011), visto que pensamos nos usuários como os estudantes, professores, pesquisadores e interessados de uma forma geral na utilização de recursos tecnológicos para auxiliar no ensino e na aprendizagem. Em resumo, o processo de captação de requisitos está representado no Diagrama 6.

Diagrama 6 – Engenharia de Requisitos (adaptado de SOMMERVILLE, 2011)




Fonte: Tiburcio (2016)

### 4.4.2 Desenvolvimento

Definidos os requisitos do software, uma versão preliminar com as características básicas, idealizadas com a análise e compreensão dos requisitos, é lançada – o protótipo. Um protótipo é uma versão inicial de um sistema de software, usado para demonstrar conceitos, experimentar opções de projeto e descobrir mais sobre o problema e suas possíveis soluções (SOMMERVILLE, 2011).

A prototipagem (ou prototipação) é geralmente um meio utilizado para validar a interpretação do engenheiro de software dos requisitos elicitados, bem como para obter novos requisitos. A vantagem dos protótipos é que eles podem facilitar a interpretação das suposições do engenheiro e, quando necessário, fornecer feedback útil sobre o motivo de estarem errados. Por exemplo: o comportamento dinâmico de uma interface do usuário pode ser melhor compreendido por meio de um protótipo dinâmico do que por descrição textual ou modelos gráficos (BOURQUE; FARLEY, 2014).

Protótipos permitem aos usuários ver quão bem (ou não) o sistema dá suporte a seu trabalho. Eles podem obter novas ideias para requisitos e encontrar pontos fortes e fracos do software; podem, então, propor novos requisitos do sistema. Além disso, a construção do protótipo pode revelar erros e omissões nos requisitos propostos.

Construído o protótipo do software, dá-se continuidade ao desenvolvimento que pode ser ramificado em dois processos fundamentais: o *design* e a *construção* do software. O *design de software*, para Bourque e Farley (2014), é considerado como o processo de definição da arquitetura, componentes, interfaces e outras características de um sistema ou componente. É, também, a atividade do ciclo de vida de engenharia de software na qual os requisitos são analisados para produzir uma descrição da estrutura interna do software que servirá de base para sua construção. O resultado do design descreve a arquitetura do software - isto é, como o software é decomposto e organizado em componentes e as interfaces entre esses componentes. Também deve descrever os componentes em um nível de detalhe que permita sua construção.

O design desempenha um papel importante na criação do software: durante essa etapa os engenheiros produzem vários modelos que formam um tipo de planta da solução a ser implementada. Podem-se analisar e avaliar esses modelos para determinar se eles permitirão cumprir os vários requisitos. Também, é possível examinar e avaliar soluções e compensações alternativas. Finalmente, usam-se os modelos resultantes para planejar atividades de desenvolvimento subsequentes, como verificação e validação do sistema, além de usá-los como entradas e como ponto de partida da construção e teste (BOURQUE; FARLEY, 2014).

O termo *construção de software* se refere à criação detalhada de software por meio de uma combinação de códigos (linguagem de programação), verificação (se o que foi programado está funcionando), teste de unidade (análise das entradas e saídas – se um comando executado por um usuário tem a correspondência esperada no software), teste de integração (análise dos comandos de forma coletiva – se os comandos interagem entre si respondendo conforme o esperado) e depuração (*debugging* em inglês – procura-se os defeitos e tenta-se reduzi-los ou eliminá-los) (BOURQUE; FARLEY, 2014).

Na etapa de construção, utiliza-se o resultado do design e finaliza-se essa etapa com o início dos testes. Os limites entre design, construção e testes (se houver) variarão dependendo dos processos do ciclo de vida do software que são usados em um projeto.

De acordo com Bourque e Farley (2014), existem alguns fundamentos para a etapa de construção de software: 1. minimizar a complexidade; 2. antecipar mudanças; 3. construção para verificação; 4. reuso; 5. padrões em construção.

1. *Minimizar a complexidade* é enfatizar a criação de códigos simples e legíveis para que todos os envolvidos no projeto compreendam as etapas do desenvolvimento e as funcionalidades a serem desenvolvidas e testadas;
2. *Antecipar mudanças* auxilia os engenheiros de software a criar um produto extensível, o que significa que eles podem aprimorar o produto sem interromper a estrutura existente;
3. *Construir para verificação* significa construir software de forma que as falhas possam ser facilmente encontradas pelos engenheiros que escrevem o software, bem como pelos testadores e usuários durante atividades operacionais e de teste independentes.
4. *Reuso* se refere ao uso de componentes existentes na solução de diferentes problemas. A reutilização sistemática pode permitir melhorias significativas de produtividade, qualidade e custo de software.
5. *Padrões em construção* sejam externos (criados por organizações internacionais) ou internos (no nível corporativo), que afetam diretamente os problemas de construção, incluem: métodos de comunicação; linguagens de programação; padrões de codificação; plataformas e ferramentas.

### 4.4.3 Validação

A etapa de validação, chamada na literatura de “verificação e validação” – V&V, tem como objetivo ajudar os desenvolvedores a criar qualidade no sistema durante o ciclo de vida. Segundo Bourque e Farley (2014), os processos de V&V fornecem uma avaliação objetiva de produtos e processos. Essa avaliação demonstra se os requisitos são corretos, completos, precisos, consistentes e testáveis. Com isso, esses processos determinam se os produtos estão em conformidade com os requisitos e se satisfazem o uso pretendido e as exigências dos usuários.

A verificação é uma tentativa de garantir que o produto seja construído corretamente, no sentido de que atendam às especificações impostas a eles. A validação é uma tentativa de garantir que o produto certo seja construído – ou seja, o produto cumpra seu objetivo específico. (BOURQUE; FARLEY, 2014).

Sommerville (2014) apresenta um sistema de validação de software baseado em testes organizados em três estágios: 1 *testes de desenvolvimento*; 2 *testes de sistema*; 3 *testes de aceitação*. Na ordem, os componentes do sistema são testados, em seguida, o sistema integrado é testado e, finalmente, o sistema é testado com os dados do usuário. Tem-se como ideal que os defeitos de componentes sejam descobertos no início do processo e os problemas de interface encontrados quando o sistema é integrado. No entanto, quando os defeitos são descobertos, o programa deve ser depurado (encontrar e reduzir os defeitos) e isso pode requerer que outros estágios do processo de testes sejam repetidos.

Nos *testes de desenvolvimento,* os componentes do sistema são testados pelas pessoas que o desenvolveram. Cada componente é avaliado de forma independente, separado dos outros. Os componentes podem ser entidades simples como funções do software ou classes de menus e funcionalidades; ou podem ser agrupamentos dessas entidades. No estágio de *testes de sistema,* os componentes são integrados para criar um sistema completo. Esse processo se preocupa em encontrar os erros resultantes das interações inesperadas entre componentes e problemas de interface. Também, visa mostrar que o sistema satisfaz seus requisitos funcionais e não funcionais, bem como testar as propriedades emergentes do sistema.

Por último, nos *testes de aceitação,* antes que o sistema seja aceito para uso operacional, ele deve ser testado com dados fornecidos pelos usuários e não com dados advindos de testes simulados. O teste de aceitação pode revelar erros e omissões na definição dos requisitos do sistema, pois dados reais exercitam o sistema de formas diferentes dos dados de teste.

### 4.4.4 Manutenção/Evolução

A manutenção é um desenvolvimento evolutivo em que as decisões são auxiliadas pela compreensão do que acontece com o software ao longo do tempo. Essa manutenção sustenta o produto de software ao longo de seu ciclo de vida. Os pedidos de modificação realizados pela equipe de desenvolvedores ou usuários devem ser registrados e rastreados; o impacto das alterações propostas é determinado; o código e outros artefatos são modificados; os testes são conduzidos e uma nova versão do produto de software é lançada (BOURQUE; FARLEY, 2014).

Existem, atualmente, quatro categorias (tipos) de manutenção: a corretiva, a adaptativa, a perfectiva e a preventiva:

> • Manutenção corretiva: modificação reativa (ou reparos) de um produto de software executado após a entrega para corrigir problemas descobertos. Incluída nesta categoria está a manutenção de emergência, que é uma modificação não programada realizada para manter temporariamente um produto de software operacional, pendente de manutenção corretiva. • Manutenção adaptativa: modificação de um produto de software executado após a entrega para manter um produto de software utilizável em um ambiente alterado ou em mudança. Por exemplo, o sistema operacional pode ser atualizado e algumas alterações no software podem ser necessárias. • Manutenção perfectiva: modificação de um produto de software após a entrega para fornecer aprimoramentos aos usuários, melhoria da documentação do programa e recodificação para melhorar o desempenho, a manutenção ou outros atributos do software. • Manutenção preventiva: modificação de um produto de software após a entrega para detectar e corrigir falhas latentes no produto de software antes que elas se tornem falhas operacionais (BOURQUE; FARLEY, 2014, p. 107, *tradução nossa*[5]).

A etapa de manutenção/evolução é contínua, no sentido de prover novos requisitos e possibilidades de atualizações do software. Ao utilizar as atuais tecnologias digitais de informação e comunicação, como smartphones, computadores, tablets, por exemplo, percebe-se que os software (aplicativos) são atualizados com uma frequência intensa. Novas versões, novas funcionalidades, novos menus, correção de erros, enfim, são diversas possibilidades a cada atualização. A interação do usuário com os sistemas e aplicativos proporciona feedbacks importantes para as equipes de construção que estão trabalhando para a evolução dos produtos.

Zaina (2014) observa "a influência que os aspectos da Interação Humano-Computador – IHC possuem na qualidade de um produto vem deixando os limites dos muros acadêmicos para se tornar um tema fortemente discutido" (p. 1). Acrescenta, ainda, que existe um cuidado na experiência do usuário para com o produto de software. Assim, destacamos a relevância da compreensão das necessidades dos usuários observadas durante a utilização dos produtos digitais na área educativa: a experiência do usuário e os retornos que eles podem apresentar com a utilização são úteis para o aprimoramento do software e, também, da engenharia que os produz.

---

[5] • Corrective maintenance: reactive modification (or repairs) of a software product performed after delivery to correct discovered problems. Included in this category is emergency maintenance, which is an unscheduled modification performed to temporarily keep a software product operational pending corrective maintenance.
• Adaptive maintenance: modification of a software product performed after delivery to keep a software product usable in a changed or changing environment. For example, the operating system might be upgraded and some changes to the software may be necessary.
• Perfective maintenance: modification of a software product after delivery to provide enhancements for users, improvement of program documentation, and recoding to improve software performance, maintainability, or other software attributes.
• Preventive maintenance: modification of a software product after delivery to detect and correct latent faults in the software product before they become operational faults.

## 4.5 O PRINCIPAL PRODUTO DA ENGENHARIA DIDÁTICO-INFORMÁTICA

Ao observar o estado da arte do desenvolvimento de recursos computacionais para a educação, encontramos estudos revelando que um dos modos de desenvolver SE é adaptando engenharias que desenvolvem outros tipos de software (BENITTI et al, 2005; LAPOLLI et al, 2009; COSTA, 2012; COSTA; COSTA, 2013). Concordamos que, de fato, existe a exigência da realização dessa adaptação. As especificidades dos software educativos os distinguem dos demais ao considerar as relações de ensino e aprendizagem, visto que para a elaboração de produtos tecnológicos com o objetivo didático, percebemos a necessidade latente de uma engenharia específica. Observa-se diversas características que estes devem possuir e percebe-se a utilização de métodos (e adaptações) não específicos para se produzir produtos educativos.

Diante disso, consentimos com Santos (2009) quando ressalta a indispensabilidade de compreender as singularidades do trabalho de concepção e criação de software educativo. O autor observa que o engenheiro de software educativo tem diante de si um sistema que se difere de outros tipos. O sistema de SE não é fechado onde os usuários e proprietários interagem entre si utilizando procedimentos pré-estabelecidos de forma previsível e traduzível em operações automatizadas e informatizadas. Para o autor, por mais simples que seja um sistema educativo, ele "traduz e delimita conhecimentos em processo dinâmico de comunicação e percepção. Consequentemente, o engenheiro de software educativos deve lidar com um conjunto de aspectos subjetivos que caracterizam o fenômeno educativo" (SANTOS, 2009, p. 18).

Assim, põe-se uma das justificativas quanto a relevância de engenharias específicas para a criação de produtos tecnológicos com fins educativos. Isso posto, para delimitar um processo de Engenharia de Software Educativo - ESE é válido compreender o que é o produto dessa engenharia.

Software educativo é todo aquele programa computacional que tem por objetivo auxiliar no ensino e na aprendizagem de diversas habilidades, competências e saberes. De acordo com Almeida e Almeida (2015), para que um programa seja considerado educativo é necessário que esse possua recursos projetados com a **intenção e a finalidade** de serem usados em contexto de ensino-aprendizagem (p. 6, grifo nosso).

Mesmo caracterizando os software educativos, ainda existem diversas tipologias que possuem finalidades diferentes nas relações educativas. Uma forma de classificação utilizada, frequentemente, é a que considera as teorias sobre ensino e aprendizagem da época em que os

software foram construídos, dando ênfase aos paradigmas desses referenciais teóricos. De acordo com Bellemain (2012),

> A diversidade de software disponíveis associa-se obviamente à diversidade dos conteúdos abordados. Mas, os software educativos distinguem-se também por suas formas de abordar os conteúdos. Assim, podemos constatar diversos tipos de software educativos que são, do nosso ponto de vista, frutos da combinação de vários elementos, como: a aplicação de um certo modelo de ensino-aprendizagem, a concretização de um certo projeto didático do autor, a escolha do tipo de situação onde o software deve intervir (2002, p. 51).

Resumimos na Tabela 3 uma tipologia de software educativo de acordo com Bellemain (2002), referente a algumas teorias cognitivas de grande relevância no cenário educativo.

Tabela 3 – Software e os paradigmas de ensino e aprendizagem

| FUNDAMENTO TEÓRICO | CARACTERÍSTICAS DOS SOFTWARE |
|---|---|
| Behaviorismo | Tentam resolver a questão da aprendizagem com sistema de treinamento, de aprendizagem por tentativa/erro. Nessa categoria aparecem os primeiros software educativos que foram desenvolvidos e os sistemas de treinamento. |
| Instrucionismo | Consistem na automatização, pelo computador, das formas tradicionais de ensino. A questão da aprendizagem é considerada como uma questão de transmissão de informação. O computador contribui permitindo uma organização não linear da informação (utilização de hipertextos) e a criação de múltiplas formas de apresentação dos conhecimentos, incluindo animações. |
| Construtivismo | Criam, no âmbito computacional, condições e ferramentas para expressão e resoluções de problemas. Contribuem para a aprendizagem permitindo as manipulações pelo sujeito dos objetos e relações através dos operadores e favorecendo assim a construção dos conhecimentos sobre esses objetos e relações. |

Fonte: adaptado de Bellemain (2002).

Em consequência dos paradigmas cognitivos, compreendemos a complexidade da ESE quando essa se preocupa em desenvolver produtos que contribuam com a aprendizagem de

competências e habilidades; auxiliem o ensino do Professor; possibilitem o auto aprendizado; sirvam como fonte de pesquisa e estudo para professores e estudantes; entre diversas outras possibilidades que os software educativos podem oferecer.

A Engenharia Didático-Informática, ancorada em aportes teóricos sobre o ensino e a aprendizagem, bem como aportes sobre auxílios e benefícios do uso das tecnologias para ensinar e aprender, possibilita a concepção (no sentido de ideias), desenvolvimento (no sentido da criação) e avaliação (no sentido de análise) de software classificados como micromundos.

A origem do termo micromundo remonta ao início dos anos 70, no contexto dos estudos sobre Inteligência Artificial, tendo como forte referência os trabalhos de Marvin Lee Minsky e Seymour Papert. Nessa comunidade, o termo foi inicialmente usado para definir um sistema que permite simular ou reproduzir um domínio do mundo real e que tem como objetivo abordar e resolver uma classe de problemas. É assim classificado pois tem o sentido de que é uma redução do mundo real, ou seja, uma pequena parte da realidade (NOSS e HOYLES, 1996 apud BELLEMAIN, 2002).

Um dos pioneiros do estudo destes ambientes foi Seymour Papert, que, em síntese,

> É um dos fundadores do laboratório de inteligência artificial do Massachussetts Institute of Technology (MIT). Foi pioneiro no estudo do uso de computadores na educação. Desenvolveu, no final dos anos mil novecentos e sessenta, a linguagem Logo, que fazia uso da programação de computadores no aprendizado da criança. Nos anos 1980, Papert desenvolve o brinquedo Lego-Logo, uma espécie de robô infantil, que faz uso de motores, sensores e outros componentes eletrônicos nos mundialmente consagrados blocos Lego. Papert foi influenciado por educadores como Jean Piaget, Lev Vigotsky, Montessori, John Dewey e Paulo Freire. Seu primeiro livro foi Mindstorms: Children, Computers and Powerful Ideas, de 1980, versando sobre computadores e educação. Mindstorm (tempestade da mente, ou paixão do pensamento, em tradução livre) é um neologismo (coisa que Papert adora) que remete a brainstorm (tempestade cerebral). O livro A Máquina das Crianças foi publicado treze anos depois (SILVEIRA, 2012).

De acordo com Silveira (2012), Papert utilizou o computador no processo de aprendizagem da criança, com ênfase para o ambiente logo. Dessas experiências surgiu o conceito de micromundo como ambientes que simulavam operações concretas de uma pessoa no seu mundo real, através de operações abstratas em programas de computador.

Segundo Papert (1980), em relação às pequenas partes da realidade que podem ser trabalhadas nestes ambientes,

> Existem limites para cada uma dessas fatias da realidade. E vou sugerir que, de uma maneira muito geral, não apenas no contexto do computador, mas provavelmente em toda a aprendizagem importante, um mecanismo essencial e central é limitar-se a um pequeno pedaço da realidade que é simples o suficiente para entender. É olhando pequenas fatias da realidade de cada vez que você aprende a entender as maiores complexidades do mundo inteiro, o mundo macro (PAPERT, 1980, p.81; tradução nossa[6]).

Papert formulou uma teoria cognitiva chamada de Construcionismo: considerando características do Construtivismo, mas com algumas oposições. O autor considerou que a aprendizagem é uma construção de relações entre o conhecimento antigo e o novo, em interações com outras pessoas, enquanto cria artefatos de relevância social. O Construcionismo se concentra na natureza conectada do saber com suas dimensões pessoais e sociais (BELLEMAIN, 2002).

Especificamente para a aprendizagem de conhecimentos matemáticos, um micromundo deve fornecer uma semântica dinâmica para um sistema formal: um sistema composto de objetos (saberes matemáticos), relações entre objetos e operações que transformam objetos e relações. Na prática, um micromundo matemático incorpora uma exibição gráfica que mostra uma visualização dos objetos iniciais. Ou seja: um micromundo matemático constitui o núcleo de um sistema de axioma intuitivo onde os alunos podem definir novos objetos e operações e investigar interativamente suas propriedades (BELLEMAIN, 2002).

De um ponto de vista prático, a integração de um micromundo para a aprendizagem necessita da criação de situações didáticas específicas:

a) favorecendo a manipulação pelo aluno das novas representações dos objetos matemáticos introduzidos pelo software,

b) permitindo organizar o ensino em torno da resolução de problemas e da experimentação (BELLEMAIN, 2002).

O objetivo do micromundo é o de criar situações problematizadoras nas quais os saberes matemáticos sejam indispensáveis para solucionar o que fora proposto como problema. As contribuições dos micromundos no ensino acontecem, apenas, quando os sujeitos interessados em ensinar elaboram situações de uso para os sujeitos interessados em aprender.

---

[6]Texto original: There are limits for each of these slices of reality. And I'm going to suggest that in a very general way, not only in the computer context but probably in all important learning, an essential and central mechanism is to confine yourself to a little piece of reality that is simple enough to understand. It's by looking at little slices of reality at a time that you learn to understand the greater complexities of the whole world, the macroworld.

As situações de uso dos micromundos se baseiam na resolução de problemas, que deve ser considerada como central na elaboração das situações para os usuários: “A resolução de problemas é um motor da manipulação dos elementos do micromundo e o motor da construção dos conhecimentos que justifica essa manipulação” (BELLEMAIN, 2002, p.60). De acordo com o autor,

> Escolher os bons problemas para ensinar com um micromundo não é somente escolher problemas interessantes do ponto de vista das noções em jogo. Os problemas devem ser adaptados a uma exploração com um micromundo do ponto de vista das estratégias de resolução que ele permite desenvolver. Com os novos sistemas de representação introduzidos pelos micromundos, novas formas de colocar os problemas aparecem. Por exemplo, na situação de construção de um quadrado apresentada não se trata do problema da produção do desenho de um quadrado, mas de um procedimento de construção de um quadrado. Assim, no caso do Cabri-géomètre, que também é o caso da maioria dos software de geometria dinâmica, diversos tipos de uso na sala de aula foram desenvolvidos correspondentes a diversas formas de problemas. Entre outros, existem problemas em torno de construções geométricas (Capponi, 2000), de simulações (Moreira Baltar, 1996), de caixas pretas (Charriere, 1996) (BELLEMAIN, 2002, p.60).

Nessa perspectiva, consideramos que a Engenharia Didático-Informática apresenta em seu alicerce teórico e metodológico condições necessárias para que micromundos sejam desenvolvidos dentro do paradigma Construtivista. Entretanto, considerando outras tipologias de software e diferentes contextos de pesquisas na área da Engenharia de Software Educativos, a EDI pode auxiliar na concepção de outros produtos, visto que as análises teóricas que podem ser realizadas fornecem requisitos teóricos e tecnológicos para qualquer tipo de software.

Com isso, discutimos o atual desenvolvimento de software educativo, bem como a percepção de articular elementos da área da tecnologia aos do ensino e aprendizagem com as definições e encaminhamentos da Engenharia Didática que apresentamos nas sessões a seguir.

## 4.6 PRIMEIRA E SEGUNDA GERAÇÃO DA ENGENHARIA DIDÁTICA

De acordo com Artigue (2002 apud CHEVALLARD, 1982), a expressão "Engenharia Didática" apareceu na didática da Matemática francesa no início dos anos 80 como um meio de responder a duas questões fundamentais:

> • como levar em conta a complexidade das aulas nas metodologias de pesquisa?

> • como pensar sobre as relações entre pesquisa e ação no sistema educacional?[7] (ARTIGUE, 2002; p. 60, tradução nossa)

Com isso, definiu-se o principal objetivo da Engenharia Didática,

> "[...] o de classificar uma forma do trabalho didático: comparável ao trabalho do engenheiro que, para realizar um projeto preciso, deve se apoiar nos conhecimentos científicos do seu domínio, aceitar submeter-se a um controle de tipo científico, mas, ao mesmo tempo, se encontra obrigado a trabalhar sobre objetos muito mais complexos do que os objetos depurados da ciência e, portanto a estudar de uma forma prática, com todos os meios ao seu alcance, problemas de que a ciência não quer ou ainda não é capaz de se encarregar (ARTIGUE, 1996).

A Engenharia Didática se tornou uma das referências como metodologia de pesquisa após a síntese de Michele Artigue nos anos 90, que foi tomada como ponto de partida e diretriz definitiva por inúmeras pesquisas (PERRIN-GLORIAN, 2009). Para Almouloud e Coutinho (2008), a Engenharia Didática tem por características: 1. ser um esquema experimental baseado em realizações didáticas em sala de aula, isto é, na concepção, realização, observação e análise de sessões de ensino; 2. ser pesquisa experimental devido ao registro em que se situa e modo de validação que lhe são associados: a comparação entre análise a priori e análise a posteriori (p. 65).

Como metodologia de pesquisa, a ED difere dos métodos experimentais, então usuais na educação, por seu modo de validação. Esse modo é interno e baseia-se na comparação entre uma análise a priori na qual várias hipóteses são empregadas e uma análise a posteriori que se baseia nos dados resultantes da implementação real. Seus vínculos com a Teoria das Situações Didáticas são expressos, em particular, na concepção e análise a priori da engenharia. As escolhas que governam o design, tanto no nível de todo o projeto quanto no nível de cada uma das situações, são explicadas mostrando as variáveis didáticas em que atuam, os ambientes que essas escolhas determinam, buscando antecipar as possíveis interações de alunos genéricos com esses ambientes e seus possíveis efeitos em termos de construção do conhecimento, em um funcionamento inicialmente supostamente didático (ARTIGUE, 2002).

---

[7] Texto original:
• comment prendre en compte la complexité de la classe dans les méthodologies de recherche?
• comment penser les relations entre recherche et action sur le système d'enseignement?

A Engenharia Didática de primeira geração (ou clássica) consiste em determinar dispositivos de ensino comunicáveis e reprodutíveis. Já na Engenharia Didática de segunda geração, o objetivo maior é a produção de recursos que podem ser utilizados pelo professor em sua aula, ou para a formação de professores (inicial ou continuada), fazendo com que os docentes aprendam a matemática ou compreendam a matemática para o ensino (AMOULOUD; SILVA, 2012).

A ED de segunda geração foi percebida a partir do momento em que a ênfase dos estudos baseados nesse referencial seguia rumos, ora interessados na pesquisa dos fenômenos didáticos, ora na efetiva participação dos professores em sala de aula (despertando o interesse na formação inicial e continuada dos docentes), bem como a produção de recursos para o ensino e a aprendizagem. A segunda geração começou a se solidificar quando se construíam situações de ensino que poderiam ser reproduzidas, porém o papel do professor que utilizara essas situações não era questionado. De acordo com Artigue (2002), "O professor, parceiro do pesquisador no projeto e teste de engenharia didática ou potencial usuário dos resultados da pesquisa, permanece uma figura transparente, pouco questionada" (p. 66, tradução nossa[8]). O que pode ser verificado a seguir,

> O uso de produtos de engenharia didática no ensino regular, com todas as suas restrições, não é considerado objeto de estudo, provavelmente com uma certa ilusão de transparência no uso de situações: o papel de como o professor não foi estudado como tal, uma vez que a teoria não havia espaço para isso no momento, parece haver uma suposição implícita de que o uso de tais situações está ao alcance dos professores, pelo menos professores experientes (PERRIN-GLORIAN, 2009, p. 60, tradução nossa[9]).

Ainda de acordo com Perrin-Glorian (2009), a produção e disseminação de recursos para o ensino e a formação de professores se tornou uma questão de pesquisa relevante com a criação de ferramentas de comunicação por computador. Os institutos de pesquisa de Educação Matemática franceses tiveram interesse no estudo e produção desses recursos. Por um lado, a

[8] Texto original: L'enseignant, partenaire du chercheur dans la conception et l'expérimentation d'ingénieries didactiques ou utilisateur potentiel des résultats de la recherche, reste une figure transparente, peu questionnée.
[9] Texto original: L'utilisation des produits de l'ingénierie didactique dans l'enseignement ordinaire avec toutes ses contraintes n'est pas prise comme objet d'étude, probablement avec une certaine illusion de la transparence de l'usage des situations : le rôle de l'enseignant n'ayant pas été étudié en tant que tel puisque la théorie ne lui fait pas de place à l'époque, il semble qu'implicitement on fait l'hypothèse que l'usage de telles situations est à la portée des enseignants, au moins des enseignants expérimentés.

produção direta de recursos por meio de pesquisa-ação, por outro lado, a transposição da engenharia didática de pesquisa voltada também para questões mais teóricas. Consequentemente, podemos evidenciar as principais diferenças entre as gerações da Engenharia Didática.

Nota-se um avanço, a partir de estudos e pesquisas, no conhecimento do campo da Educação e nas necessidades da formação de professores com as pesquisas que utilizam a Engenharia Didática como referencial teórico-metodológico. Com esse avanço, é possível construir engenharias didáticas que visam produzir recursos para a Educação e também estudar o impacto que a pesquisa em didática pode causar. Esse conjunto é classificado como Engenharia Didática de segunda geração (PERRIN-GLORIAN, 2009). Na Tabela 4, algumas das principais características e diferenças entre a primeira e a segunda geração estão resumidas.

Tabela 4 – Engenharias de 1° e 2° geração, objetivos e aspectos centrais

| | **Objetivo(s)** | **Aspectos centrais** |
|---|---|---|
| **ED 1ª geração** | Elaborar e estudar propostas de transposição didática para o ensino. | Metodologia de pesquisa e produto. |
| **ED 2ª geração** | Determinar os princípios que comandam a engenharia que se quer transformar em recurso para o ensino regular e estudar as condições de sua divulgação. | Três funções não independentes: a investigação, o desenvolvimento e a formação de professores por meio da análise;<br>Necessita de vários níveis de construção. |

Fonte: Amouloud e Silva (2012, p. 32).

Nota-se na ED de segunda geração um viés de criação de recursos, seja para o ensino, para o planejamento do professor ou para a formação (inicial ou continuada) docente. Essa geração da ED tem por característica ser, simultaneamente, uma engenharia para a concepção de recursos e para a formação de professores envolvidos nesse projeto.

Sobre as três funções apresentadas na Tabela: investigação, desenvolvimento e formação; Perrin-Glorian (2009) destaca na função de pesquisa a tentativa de fazer aparecer fenômenos didáticos e estudá-los, busca-se produzir conhecimento didático. A função de desenvolvimento é manifestada pela produção de recursos utilizáveis por professores (ou formadores de professores). A função de formação visa a aprendizagem profissional dos

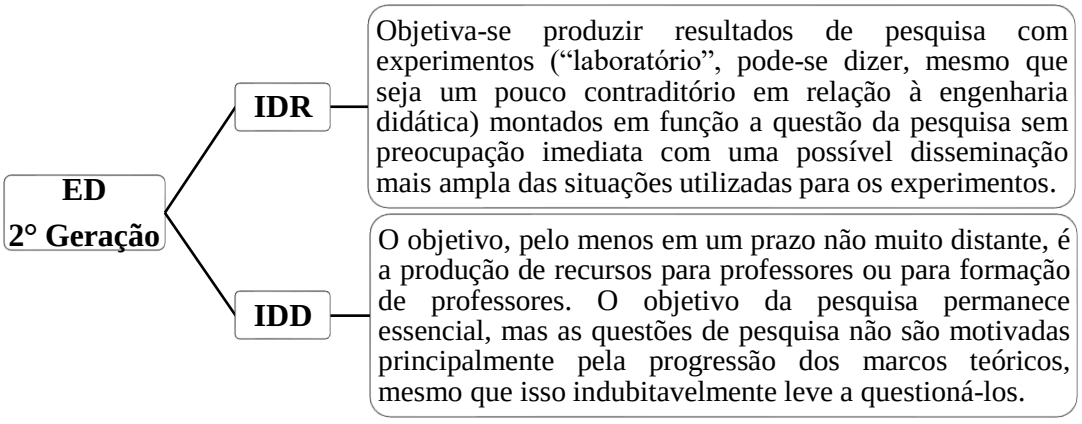
professores (e não apenas a aprendizagem de estudantes) através da análise, produção e modificação de situações para os alunos. Com isso, iremos apresentar algumas ramificações importantes presentes na segunda geração da ED e que contribuem, significativamente, para a estruturação dessa pesquisa.

### 4.6.1 A segunda geração e suas ramificações

A fim de compreender as ramificações, bem como os objetivos a ED de segunda geração, observamos que os propósitos tanto do professor quanto do pesquisador que utiliza esse referencial devem ser ponderados: "[...] devemos considerar o objeto do trabalho, o objetivo principal da pesquisa que leva à produção dessa engenharia e distinguir a engenharia didática que tem como objetivo a pesquisa e a que visa desenvolver recursos para professores (e formação de professores)." (PERRIN-GLORIAN, p 69, tradução nossa[10]).

Delimita-se, assim, segundo Perrin-Glorian (2009), duas ramificações da segunda geração da ED: "*L'Ingénierie Didactique pour la Recherche*"- IDR – Engenharia Didática para Investigação e "*L'Ingénierie Didactique pour le Développement et la Formation*" - IDD[11] – Engenharia Didática para o Desenvolvimento e Formação. Resume-se no Diagrama 7, a seguir, algumas das principais características dessas vertentes.

Diagrama 7 – Comparativo entre IDD e IDR




[10] Texto original: il nous faut prendre en compte l'objet du travail, l'objectif premier de la recherche qui conduit à produire cette ingénierie et distinguer l'ingénierie didactique qui a pour seul objectif la recherche et l'ingénierie didactique qui vise le développement de ressources pour les professeurs (et la formation).

[11] Pesquisas no Brasil que utilizam o referencial da Engenharia Didática traduzem "recherche" como "investigação", embora a tradução literal seja "pesquisa"; também adotaremos a tradução habitual para facilitar a leitura e compreensão, bem como as siglas IDR e IDD, que não estão traduzidas.

Fonte: elaborado pelo autor. Adaptado de Perrin-Glorian (2009, p. 69, tradução nossa[12]).

Na IDR, há uma clara intenção de fazer surgir fenômenos didáticos e estudá-los, utilizando situações experimentais fundamentadas nas questões de pesquisa. Não há, de imediato, a preocupação de replicações futuras das situações que estão em estudo. Em contrapartida, na IDD, o principal foco é a produção de recursos para professores ou para a formação docente, segundo Almouloud e Silva (2012). Os autores acrescentam ainda que,

> [...] os conhecimentos dos alunos são controlados teoricamente em todos os casos, mais é uma variável mais ou menos fixada na IDR, enquanto no caso da IDD é necessário prever adaptações dessas situações e meios para conduzi-los. O papel do professor é controlado pela teoria, no caso da IDR. Enquanto na IDD, uma flexibilidade nas decisões deve ser prevista. E, por fim, as exigências institucionais podem ser negligenciadas no caso da IDR, são incontornáveis no caso da IDD e consequentemente devem ser levadas em consideração teoricamente (ALMOULOUD; SILVA, 2012, p. 28).

Dessa forma, ao utilizar a Engenharia Didática como referencial teórico e metodológico, observamos que seriam necessárias algumas características tanto da primeira quanto da segunda geração. Isso visto que construir um software educativo deve levar em consideração a utilização do produto para o ensino e a aprendizagem, mas também deve compreender a função do professor planejando, elaborando as situações de uso e verificando como o recurso digital auxiliará suas aulas.

Sabemos que são duas vias importantes: pensar nas situações de utilização e nos saberes mobilizados pelo professor/pesquisador no momento de criar as situações de uso do software. A ED de segunda geração apresenta etapas que envolvem a formação do professor, seja ela inicial ou continuada, o que não cabe, temporalmente, num processo de desenvolvimento de um recurso digital. Embora a efetiva participação docente na criação de software seja uma prerrogativa que consideramos, trabalhar a formação docente é um processo que abrange diversas outras etapas.

---

[12] Texto original : L*'ingénierie didactique pour la recherche (1DR),* on vise à produire des résultats de recherche avec des expérimentations (« de laboratoire » pourrait-on dire même si c'est un peu antinomique de l'ingénierie didactique) montées en fonction de la question de recherche sans souci immédiat d'une éventuelle diffusion plus large des situations utilisées pour les expérimentations [...].
*L'ingénierie didactique pour le développement et la formation (IDD),* l'objectif au moins dans un terme pas trop éloigné, est la production de ressource(s) pour les enseignants ou pour la formation des enseignants. L'objectif de recherche reste essentiel mais les questions de recherche ne sont pas motivées en premier lieu par la progression des cadres théoriques même si cela amènera sans doute à les questionner.

Com o exposto, verificamos as principais características das ramificações da Engenharia Didática de segunda geração (IDR e IDD). Porém, acrescentamos ainda a existência de outras vertentes da segunda geração: *Engenharia Fenomenotécnica*, *Engenharia de Formação*, *Engenharia Didática Profissional*, *Engenharia dos Domínios de Experiência* e *Engenharia Didática do Percurso de Estudo e Pesquisa*. Por opção de alinhamento teórico, bem como para auxiliar nos objetivos desta pesquisa, identificamos que os indicativos teóricos e metodológicos da ED de primeira geração aliados com a IDD são os que mais contribuem para a criação de software educativo. Na sessão a seguir, explicitam-se essas contribuições.

## 4.7 EDI: UNIÃO DE ELEMENTOS DE DUAS ENGENHARIAS

Todas as "versões" da Engenharia Didática têm como ponto de unificação, segundo Artigue (2009), fazer parte do design, implementação e avaliação de dispositivos didáticos com objetivos designados, claramente fundamentados em bases teóricas. Os objetivos são diversos: exploração de organizações matemáticas e/ou didáticas não existentes até o momento; teste de hipóteses ou construções teóricas; estudo do funcionamento de sistemas didáticos sob determinadas condições; produção de recursos para ensinar este ou aquele conceito ou tema; estabelecer sistemas de formação de professores; apoiar ou preparar desenvolvimentos curriculares locais ou globais.

Segundo Perrin-Glorian (2009), para utilizar a Engenharia Didática na educação regular é necessário garantir sua validade no nível experimental, ou seja, é necessário planejar pelo menos dois níveis de engenharia (talvez mais), com objetivos diferentes. Para autora, é a combinação desses dois níveis que deve ser considerada em uma engenharia didática para a criação de recursos e formação docente, são eles:

- Primeiro nível em condições experimentais "protegidas" específicas para testar a validade teórica das situações (ou seja, sua capacidade de produzir o conhecimento esperado, visando pelo menos um teorema da existência) e identificar as escolhas fundamentais da engenharia: é essencial, em referência ao saber direcionado, o que se enquadra no contexto escolhido e pode ser alterado, adaptado.

- Segundo nível para estudar a adaptabilidade das situações ao ensino, a negociação da primeira engenharia; o desvio da implementação e as transformações realizadas são objeto de estudo das repercussões na própria engenharia didática, compreensão do funcionamento do conhecimento em questão no sistema escolar (professor, alunos, ...).

Ao observar essa construção para utilização da Engenharia Didática, percebemos que a Engenheira de Software Educativo tem diante de si a complexidade de compreender os fenômenos didáticos e também os tecnológicos. Para Bellemain et al (2014), a ESE, como área de pesquisa, deve se apoiar nos conhecimentos científicos do seu domínio. Assim utilizando uma metodologia científica para desenvolver seus produtos, trabalhando com objetos complexos, pois a concepção de software educativo é um processo que tem especificidades muito diferentes da construção de aplicativos comerciais, bancários ou domésticos, percebidos os objetivos e peculiaridades distintas, conforme já fora mencionado.

Analisando as demandas do processo de criação de software educativo, elegemos como hipótese que a utilização da Engenharia Didática articulada com a Engenharia de Software é um caminho teórico-metodológico para essa criação. Enquanto a ED trata da construção de sequências de ensino-aprendizagem a partir da utilização de conceitos e resultados de pesquisa, a ES coordena os conhecimentos sobre como as ferramentas digitais podem auxiliar atendendo aos requisitos educativos. Consideramos, de acordo com Bellemain et al (2014), que "a concepção e o desenvolvimento de software educativos exige a mobilização de uma engenharia didática específica que deve integrar conceitos e métodos da informática" (p. 6).

Na concepção de um software, existem idas e voltas para verificar um controle teórico da construção e realização dessas sessões. O referencial teórico é testado ao mesmo tempo que as situações são desenvolvidas e realizadas. Obviamente, o controle teórico está relacionado à questão de pesquisa e ao(s) arcabouço(s) teórico(s) utilizado(s) (PERRIN-GLORIAN, 2009).

Tanto na criação de situações de ensino quanto na de software educativo existe uma busca por "situações fundamentais". Para Artigue (2002), referente ao conhecimento a ser trabalhado em uma situação de ensino, a questão, principal objeto de debate, é a existência, para qualquer conceito, de uma situação fundamental, isso é, uma situação característica do saber em jogo, no sentido de que ela é ideal. Já no campo da Engenharia de Software, Sommerville (2011) destaca que na fase de especificação dos requisitos do sistema, busca-se produzir um documento que satisfaça as necessidades dos usuários finais que serão expressadas nas situações de utilização do produto a ser desenvolvido.

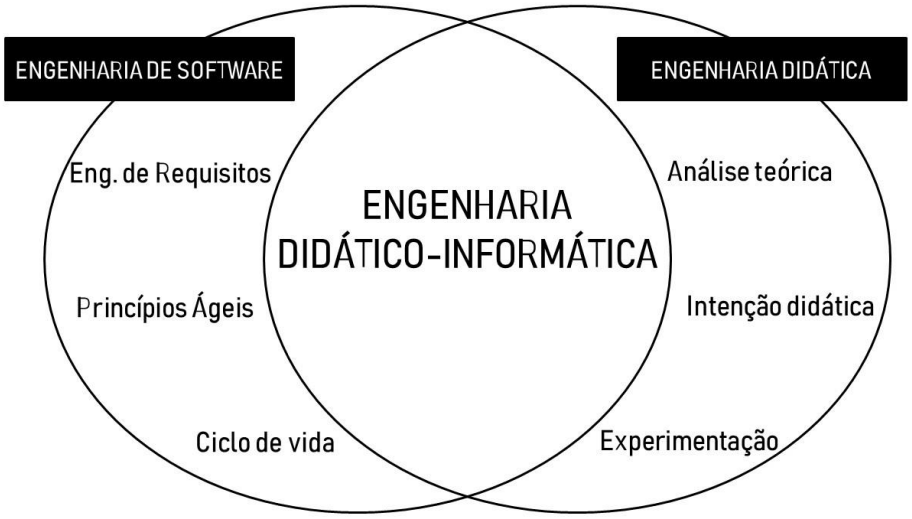
Analisando experiências educacionais com uso de tecnologias computacionais, consideremos, inicialmente, os pesquisadores debruçados no estudo de um determinado software. Em linhas gerais, analisam-se as possibilidades de utilização do recurso para determinado campo de conhecimentos e verificam-se quais são as potencialidades, limitações, benefícios, falhas e problemas gerais encontrados com o uso. Essa investigação, com foco nas

situações, gera um feedback e diversas atualizações podem ser realizadas enriquecendo a atualização do software com as observações realizadas. Ao considerarmos o uso de software educativo por professores, é simples observar que em suas atividades em sala de aula e também de planejamento, esses profissionais conseguem verificar novas possibilidades e formas de manuseio dos produtos os adequando corriqueiramente para as realidades de seus estudantes.

Essas atividades de investigação com foco na utilização são essenciais, contudo, a busca por situações fundamentais não é o propósito maior no desenvolvimento nem de sequencias de ensino nem de software. Para Artigue (2002), a questão que então se coloca ao pesquisador é: sob o quadro das restrições existentes; organizar as mediações do professor de maneira que na resolução das situações postas; a divisão de responsabilidades entre professor e alunos é otimizada, no sentido da devolução aos estudantes de uma máxima responsabilidade. Se ele se situa no quadro da Teoria das Situações Didáticas – referencial teórico base da Engenharia Didática, não há mais então uma situação única, mas uma sucessão de situações. Na verdade, cada mediação do professor é refletida por uma evolução do problema a ser resolvido.

Com isso, concluímos que é importante situar o lugar da Engenharia Didático-Informática dentro dos estudos de concepção, produção e utilização de tecnologias computacionais para o ensino e a aprendizagem: a EDI é uma ***metodologia*** para a produção de software educativo que possui como ***métodos*** princípios da Engenharia de Software e da Engenharia Didática, bem como os ***processos*** de criação de situações de utilização e situações didáticas propostos nessas duas engenharias, conforme o Diagrama 8.

Diagrama 8 – Aportes da EDI




Fonte: o autor.

Em nossa visão, desenvolver software educativo é um processo de união entre os métodos, técnicas e análises da Engenharia de Software com as teorias, orientações e referenciais da Educação. Dentre os diversos referenciais metodológicos e teóricos no campo educacional, sobre a construção de situações de ensino e aprendizagem, sobre pesquisa, técnicas de elaboração de aulas e sequências de ensino, escolhemos a Engenharia Didática visto sua estrutura que alia pesquisa ao ensino para compreender as relações educativas.

Assim, da Engenharia de Software consideramos os avanços tecnológicos, as técnicas de levantamento de requisitos, a estrutura e organização das equipes de desenvolvimento – inserindo o usuário no processo e as etapas organizacionais para a concepção e construção do software. Já a Engenharia Didática auxilia na compreensão de como as tecnologias podem ser utilizadas para contribuir com o ensino e a aprendizagem, trazendo a reflexão sobre aspectos didáticos, cognitivos, epistemológicos e de outras naturezas, que possam auxiliar no levantamento dos requisitos do software, na criação de situações de uso, bem como na análise da utilização e na validação do produto em elaboração.

Inicialmente, no levantamento de requisitos, consideramos que os ***princípios ágeis*** inserem os usuários na concepção. Observe que a efetiva participação do professor, do pesquisador, do estudante e de outros usuários envolvidos no projeto e uso de software educativo cria um ambiente rico em conhecimentos, experiências e retornos da utilização. Ao considerar a concepção do software educativo, temos: o professor que elabora as situações de ensino, o pesquisador que procura entender as relações da utilização de tecnologias educacionais e o estudante como usuário final para auxiliar no processo de entender benefícios e limitações do software com sua experiência de uso.

As características, fundamentos e pressupostos da Engenharia Didática fazem emergir reflexões a serem consideradas no desenvolvimento de software educativo. Quanto à ***análise teórica***, percebemos o auxílio na compreensão dos requisitos iniciais – essa é a primeira conexão com a Engenharia de Software, pois os ***requisitos*** de um software educativo são pautados em uma fundamentação que deve contribuir para a aprendizagem, para o ensino e para auxiliar na complexidade dos saberes. É essa análise que nos possibilita compreender as necessidades e demandas do ensino e aprendizagem e como as tecnologias podem auxilar a superiar os desafios e obstáculos.

Referente à ***intenção didática***, utilizamos essa premissa visto que software educativo tem como função principal auxiliar no ensino e na aprendizagem de saberes específicos pois

quando se idealiza a produção de um software uma das primeiras circunstâncias a serem compreendidas é a efetiva contribuição do produto para auxiliar as relações didáticas.

Na ***experimentação*** do software, considerando a multiplicidade de usuários, quando é realizada, acreditamos que é considerável criar situações que sejam reprodutíveis – que outros usuários do software ,também, possam utilizá-las com a mínima instrução possível. Na Engenharia Didática, fala-se de situações reprodutíveis que podem ser reproduzidas em contextos e com usuários diversos.

Por último, referente ao ***ciclo de vida***, as etapas são necessárias para guiar a concepção bem como a evolução do software. Por mais que o desenvolvimento não seja um processo rígido quanto as suas etapas, é necessário ter uma organização formal mínima para inicar a criação do produto e concluí-la. Essas relações foram estabalecidas quando comparamos as etapas das duas engenharias aqui discutidas. À vista disso, nos próximos capítulos, detalharemos cada fase da EDI a partir dos estudos realizados.

Em particular, quanto ao tempo dedicado pelas equipes para a concepção, desenvolvimento, produção e utilização dos software a serem criados, é válido ressaltar que o processo de validação dos produtos deve considerar o tempo que os estudos e pesquisas possuem para serem concluídos. Os projetos de desenvolvimento de software discutidos na presente pesquisa estavam inseridos em estudos de mestrado, doutorado, em grupos de pesquisa e estudos acadêmicos. Assim, o tempo para conclusão das investigações, bem como do desenvolvimento dos produtos, as análises, utilizações e conclusões é limitado, observado o tempo de conclusão dos cursos que os pesquisadores/estudantes estão inseridos, as demandas de desenvolvimento de software que podem ser longas e, também, a permanência dos pesquisadores interessados nos grupos. Com isso, é necessário ponderar o início do projeto de desenvolvimento de software educativo, considerando questões de viabilidade temporal, analisando a real possibilidade de concluir o que se pretende.

# 5 PROCEDIMENTOS METODOLÓGICOS

Neste capítulo apresentamos a metodologia utilizada para a realização do presente estudo. Consideramos, na revisão de literatura, a preocupação com a análise dos paradigmas atuais de desenvolvimento de software educativo. Posterior a isso, realizamos um resgate histórico – investigando as engenharias a quais os software Casyopée, Function Studium e Modellus foram submetidos a fim de serem criados.

Em seguida, realizamos uma abordagem analítica com a finalidade de compreender como os projetos do Conics Studium 3D, Function Studium e Magnitude Studium utilizaram a Engenharia Didático-Informática para serem desenvolvidos. Desse modo, apresentamos aqui os instrumentos de coleta de dados, os questionamentos e como essas análises foram efetuadas.

## 5.1 VALIDAÇÃO DO CONHECIMENTO

Os padrões de pesquisa na área da Educação Matemática contemplam, em sua essência, características e particularidades do ensino e aprendizagem. Segundo Garcia e Machado (2007, p. 1), o objeto de estudo da Educação Matemática "é a compreensão, a interpretação e a descrição de fenômenos referentes ao ensino e à aprendizagem da matemática, nos diversos níveis de escolaridade, tanto na sua dimensão teórica, quanto prática".

Ao investigar a criação e a utilização de software educativo, Bellemain et al (2017) propôs um estudo sobre como se considera válido o conhecimento produzido na Engenharia de Software Educativo. O estudo de Tiburcio (2016) foi analisado e, em síntese, a estrutura da investigação pode ser descrita na Tabela 5.

Tabela 5 – Etapas do estudo de Tiburcio (2016)

| | |
|---|---|
| PROBLEMA | Desenvolvimento de software educativos que não levam em consideração características do ensino e aprendizagem dos conhecimentos envolvidos. |
| HIPÓTESE | A articulação entre os conhecimentos da Educação Matemática e a Engenharia de Software pode fornecer produtos que atendam às especificidades dos conhecimentos. |
| ESTUDO TEÓRICO /FUNDAMENTAÇÃO | Engenharia de Software (Metodologias Ágeis + Engenharia de Requisitos; Engenharia de Software Educativos; Engenharia Didática. |
| DESENVOLVIMENTO | Montagem de equipe para desenvolvimento; Estudo teórico; Estudo de caso. |
| TESTES | Estudo de caso sobre Taxa de Variação de Funções Polinomiais. |
| ESTUDO DOS RESULTADOS (VALIDAÇÃO) | Confronto com a hipótese; Verificação da utilidade do processo criado. |

Fonte: Bellemain et al (2017).

Neste cenário, percebe-se que a concepção de software educativo não está totalmente inserida na Educação Matemática, pois, além de existirem particularidades tecnológicas que necessitam de fundamentação teórica e prática proveniente da Engenharia de Software, a criação de produtos tecnológicos não é algo específico da área. Etapas metodológicas como desenvolvimento, testes e validação de software são comuns em pesquisas da área da Engenharia de Software.

A partir dessa percepção, definimos os procedimentos metodológicos dessa pesquisa como teóricos, no sentido de aprimorar uma metodologia de concepção e criação de software a partir de análises investigativas e observacionais. A evolução/aperfeiçoamento da Engenharia Didático-Informática foi baseada em atualizações dos referenciais teóricos-metodológicos, em uma abordagem histórica e na investigação de projetos de desenvolvimento que utilizaram o modelo de processo de software da EDI para a produção de recursos tecnológicos.

## 5.2 ENGENHARIAS APLICADAS EM PRODUTOS COM PESQUISAS CONSOLIDADAS

Nesta etapa analisamos as engenharias de software as quais os produtos listados na Tabela 6 foram submetidos a fim de serem criados em seus respectivos espaços e ao longo do tempo. Esses software possuem resultados relevantes em pesquisas na área da Educação Matemática, uma das justificativas para a escolha dos mesmos.

Tabela 6 – Software a serem estudados e seus desenvolvedores

| NOME | DESENVOLVEDORES |
|---|---|
| CASYOPEE | Jean Baptiste Lagrange e colaboradores |
| FUNCTION PROBE | Jere Confrey e colaboradores |
| MODELLUS | Vitor Teodoro e colaboradores |

Fonte: o autor.

Justificamos a escolha dos software listados na Tabela 6 considerando suas tipologia, visto que os ambientes de simulação/micromundos são o principal produto da Engenharia Didático-Informática. Além disso, eles têm características comuns: foram desenvolvidos em ambientes de pesquisa e trabalham com a pluralidade de representações matemáticas. Contactamos os desenvolvedores desses produtos e esses se colocaram à disposição para colaborar com esse estudo. Esclarecemos a seguir algumas particularidades de cada um desses software e seus desenvolvedores.

O software Casyopée foi desenvolvido no âmbito de um grupo de pesquisa que possui o mesmo nome; no cerne de investigações para auxiliar o ensino e a aprendizagem de Álgebra e Análise para o Ensino Médio, liderados pelo pesquisador Jean Baptiste Lagrange. O Casyopée

é um ambiente que permite simulações e criação de situações matemáticas. Tem por objetivo facilitar as representações numéricas, gráficas e algébricas de funções matemáticas, também, relacionando-as com a Geometria.

A escolha desse software se justifica pelas relações que podem ser feitas entre a Álgebra e a Geometria, bem como o fato de, também, ser um ambiente micromundo que possui pesquisas consolidadas na área da Educação Matemática e um desenvolvimento robusto pautado em pesquisas sobre o auxílio ao ensino e a aprendizagem de saberes matemáticos. De igual modo aos demais software aqui discutidos, surgiu no ambiente acadêmico, fruto de pesquisas e investigações.

O Function Probe foi desenvolvido no âmbito das investigações do Grupo de Pesquisa em Educação Matemática da Universidade de Cornell sob a supervisão de Jere Confrey. Também, fora desenvolvido no ambiente acadêmico e é fruto de investigações de pesquisadores na problemática da utilização de tecnologias para auxiliar o ensino e a aprendizagem da Matemática.

Um dos principais recursos do software, que reitera a justificativa por analisar a engenharia, é a articulação entre as representações gráfica, tabular e algébrica. Esse software também necessita da criação de situações problemas para que possa ser utilizado com a finalidade de contribuir para o aprendizado, o que, igualmente, o enquadra na classificação de micromundo.

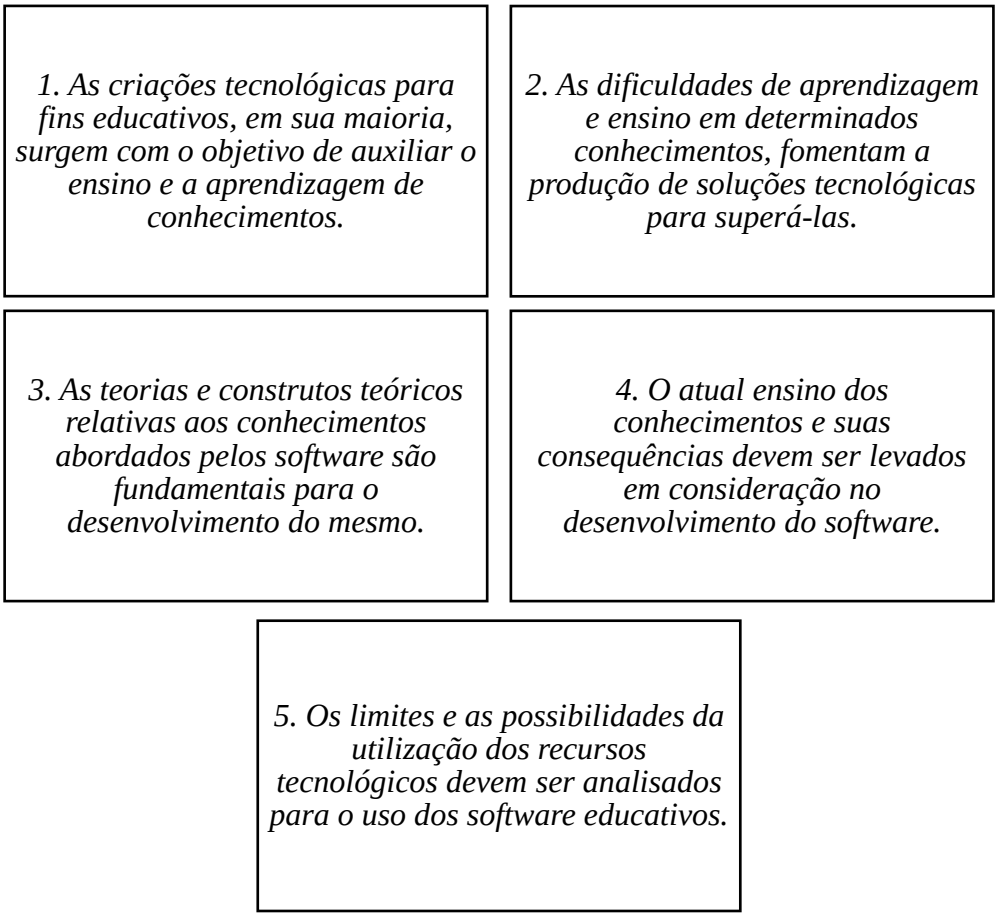
O Modellus foi desenvolvido por um grupo de pesquisadores sob o comando de Vitor Teodoro, da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa. Seu objetivo inicial era auxiliar o ensino dos conhecimentos da Física, contudo, as possibilidades de trabalhar com saberes da Matemática foram tantas que diversas pesquisas foram realizadas para essa área. No mesmo sentido dos outros software que foram analisados nessa pesquisa, o Modellus é um criador de situações problemas e um ambiente de simulações. Visto que seu objetivo inicial de trabalhar com a Física, esse possui diversas funcionalidades que simulam situações reais.

Os software escolhidos possuem características comuns, porém formas particulares de abordar as áreas de conhecimentos que atuam. Ressaltamos que as diversas pesquisas e os resultados de investigações sobre a utilização desses programas na Educação Matemática serviram como fonte de obter mais informações além das que obtivemos com a realização das entrevistas.

Sendo assim, o resgate histórico pretendia levantar elementos e características

pertinentes das engenharias dos software em questão. Construímos um Questionário/Entrevista para ter acesso a algumas particularidades desses projetos. A construção do questionário levou em consideração algumas hipóteses que julgamos pertinentes no desenvolvimento de software educativos. Organizamos essas em três diagramas (Diagramas 9, 10 e 11).

Diagrama 9 – Facetas do desenvolvimento de software educativo - Parte 1



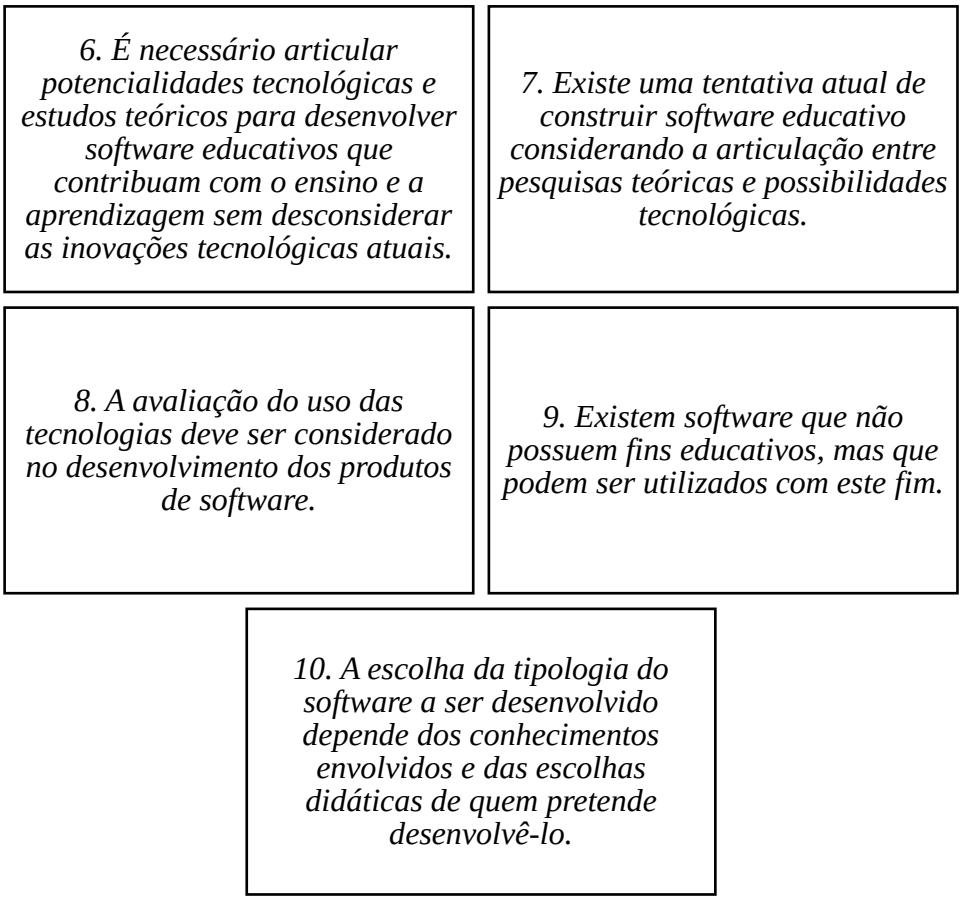

Fonte: o autor.

Os cinco primeiros itens refletem nosso objetivo de, com as interações com os desenvolvedores, identificar as necessidades do ensino e da aprendizagem as quais os software estavam sendo construídos para superar. Com isso, pretendíamos compreender como os desenvolvedores analisaram (ou não) teorias didáticas e cognitivas, ou construtos teóricos que versassem sobre a aprendizagem e o ensino a fim de conceber os produtos tecnológicos com fundamentação teórica pautada na Educação Matemática.

É importante, também, observar que na construção de um software educativo há sempre uma intenção didática. De acordo com Almeida e Almeida (2015), software educativos são definidos como programas que possuem recursos projetados com a intenção e a finalidade de

serem usados no contexto do ensino e da aprendizagem. Sendo assim, buscamos compreender, com os questionamentos formulados na entrevista/questionário, quais foram os objetivos dos software desenvolvidos no sentido de auxiliar as relações educativas.

Diagrama 10 – Facetas do desenvolvimento de software educativo - Parte 2




Fonte: o autor.

Quando consideramos limites e possibilidades, em consonância com os itens seis e sete, do Diagrama 10, dos recursos tecnológicos, compreendemos que as tecnologias devem ser utilizadas analisando quais serão os benefícios que esse uso trará. Recursos como papel e lápis, por exemplo, continuam sendo indispensáveis, contudo, verificados os amplos auxílios no ensino e aprendizagem advindos das novas tecnologias, é coerente considerar em quais aspectos há efetiva contribuição. É necessário, assim, que se avaliem os benefícios que o software trará, como consta no oitavo item.

No nono item trazemos a questão da adaptação de software que é realizada corriqueiramente. Mesmo possuindo conhecimento prévio das equipes de desenvolvedores, bem como dos produtos, sabendo que os software escolhidos para serem investigados nesta

pesquisa foram construídos com fins educativos, faz-se necessário tornar explícito o que os desenvolvedores idealizaram antes e durante da concepção dos mesmos. Tivemos como finalidade verificar as reais intenções e também conhecer os fins para quais os software foram criados.

Referente à tipologia do software, consideramos o décimo item para elaborar alguns questionamentos. Nesse sentido, uma importante questão surge e nos levou a pensar como os desenvolvedores escolheram a tipologia do software a ser construído: o pesquisador escolhe o tipo de software antes de se debruçar nos saberes que serão abordados ou, após análises consistentes desses saberes, o pesquisador delimita que tipo de software é adequado para contribuir com o ensino e a aprendizagem na área? Tivemos por objetivo responder esse questionamento ao longo das entrevistas e interações.

Observando a problemática da produção de conhecimento especializado destinado aos professores, com o objetivo de auxiliá-los na escolha e avaliação de software para a realização das atividades de ensino e planejamento de suas aulas, Tavares (2017) realizou uma investigação com o objetivo de classificar os software educativos e concluiu que existem duas classificações possíveis: quanto aos seus objetivos e quanto ao tipo de aprendizagem.

Quanto aos objetivos, os autores resumem os tipos com a Figura 1.

Figura 1 – Classificação dos Software Educacionais por objetivos


Fonte: Tavares (2017).

Quanto à aprendizagem o autor resume com a seguinte tabela:

Tabela 7 – Classificação dos Software Educacionais por objetivos


Fonte: Tavares (2017)

Assim, observando o levantamento realizado por Tavares (2017), entendemos que para a elaboração de cada tipo de software devemos ter uma engenharia compatível a essa construção como afirma Tchounikine (2009) apresentando a existência de engenharias específicas para um tipo de objeto X, onde X = situação de aprendizagem à distância, X = cenário para a aprendizagem colaborativa, X = simulação, X = geometria dinâmica, etc. Trata-se de um X de diversas naturezas e os elementos de engenharia correspondentes igualmente (TIBURCIO et al, 2015).

Diagrama 11 – Facetas do desenvolvimento de software educativo - Parte 3

| | |
|---|---|
| *11. A multiplicidade de sujeitos e de conhecimentos envolvidos no desenvolvimento de um software pode contribuir para a qualidade do mesmo.* | *12. A utilização de modelos de processo de software ajuda na compreensão do que é necessário para que o produto atenda às necessidades observadas.* |

Fonte: o autor.

O décimo primeiro item reflete a composição do grupo de desenvolvedores. Consideramos como ideal a ser alcançado a composição de uma equipe transdisciplinar, visto

que acreditamos que na concepção e desenvolvimento de software educativo é importante reunir especialistas das diversas áreas sabendo que estes podem trazer contribuições para o melhor aproveitamento do software (TIBURCIO, 2016). Procuramos compreender como as equipes foram compostas a fim de analisar como e quais os saberes das diversas áreas foram articulados nos projetos.

O último item que consideramos para a elaboração da entrevista foi referente ao processo de engenharia de software propriamente dito. A Engenharia de Software é uma área que se propõe a estabelecer princípios teóricos e metodológicos para a concepção, o desenvolvimento e a validação de software. No caso mais específico da produção de software educativo, a engenharia deve articular elementos de educação, de informática, de conteúdos disciplinares e de didática desses conteúdos.

Segundo Almeida e Almeida (2015), "produzir um software educativo não se limita a um bom gerenciamento de projeto aliado a uma boa ideia de docentes aguardando o apoio da tecnologia da informática" (p. 35). A análise de um sistema computacional com finalidades educacionais enfrenta o desafio de incorporar os mecanismos pedagógicos e didáticos que constituem a base de todo instrumento de ensino e de aprendizagem.

Em vista disso, apresentamos a seguir os questionamentos que foram realizados aos desenvolvedores dos software educativos escolhidos na etapa do resgate histórico e os objetivos de cada interação.

Tabela 8 – Primeiros questionamentos

| **QUESTIONAMENTOS INICIAIS** (PERGUNTAS GERAIS) |
|---|
| Como surgiu a necessidade do desenvolvimento do software? |
| O que se pretendia com o software? |
| Quais problemas existiam que o software se apresentaria como solução? |

Fonte: o autor.

O primeiro grupo de questionamentos teve por objetivo compreender o contexto da concepção dos software, bem como entender quais eram as demandas e finalidades com o uso do produto após criado. Partimos da hipótese de que existem motivações para a concepção de

algum produto, sejam elas acadêmicas, com fins de resolver problemas ou auxiliar em algo.

Tabela 9 – Equipe de desenvolvimento

| **SOBRE A EQUIPE DE DESENVOLVIMENTO** |
|---|
| Quais eram as características da equipe? |
| Foram considerados profissionais de diversas áreas para compor a equipe? |
| Existiam especialistas em quais áreas? |

Fonte: o autor.

Com esse bloco de questionamentos, objetivamos compreender a composição das equipes, as áreas de formação dos membros e se a pluralidade de áreas havia sido considerada para a elaboração dos produtos.

Tabela 10 – Auxílio ao ensino e a aprendizagem

| **OBJETIVOS DE ENSINO E APRENDIZAGEM** |
|---|
| O software possuía fins de auxiliar o ensino e a aprendizagem? |
| Foram observadas as etapas na construção de um conceito, pelo sujeito, para desenvolver o software? |
| Foram elencadas dificuldades de ensino e aprendizagem e estas foram consideradas no desenvolvimento do software? |
| Quais recursos disponíveis no software foram desenvolvidos para atender às especificidades do ensino e da aprendizagem? |

Fonte: o autor.

Neste bloco, procurou-se verificar os objetivos dos software referentes ao ensino e a aprendizagem: se haviam sido observadas dificuldades e o produto poderia ser concebido como uma possível solução para superar tais dificuldades e, também, compreender quais recursos haviam sido idealizados para auxiliar o ensino e a aprendizagem.

Tabela 11 – Sobre o desenvolvimento dos software

| QUESTÕES TECNOLÓGICAS |
| --- |
| Em que linguagem foi desenvolvido o software? |
| Que tipo de prototipação foi utilizada para o desenvolvimento? |
| Quais características das versões iniciais foram implementadas até a primeira versão ser lançada? |
| Na prototipação do software os usuários participaram dos testes e experimentação? |

Fonte: o autor.

Nesta etapa da entrevista, procuramos compreender aspectos de programação e prototipação. Qual linguagem de programação foi utilizada, como o protótipo foi desenvolvido e as implementações que surgiram até a primeira versão. Além disso, buscamos saber se os usuários finais tiveram participação no processo de desenvolvimento do software.

Por fim, tentamos compreender e relacionar as etapas das engenharias aos pressupostos que a Engenharia Didático-Informática apresenta em seus fundamentos. As questões pretendiam verificar se os pesquisadores consideraram estudos teóricos sobre o ensino, a cognição, a epistemologia, os avanços tecnológicos e tudo o que pode contribuir para a aprendizagem e o ensino dos saberes a serem trabalhados com as tecnologias digitais.

Com o instrumento finalizado, contactamos os pesquisadores, verificamos as disponibilidades dos mesmos e iniciamos as entrevistas e as interações. Os resultados são apresentados no capítulo a seguir.

## 5.3 ESTUDO DA UTILIZAÇÃO DA ENGENHARIA DIDÁTICO-INFORMÁTICA

O segundo momento da metodologia dessa investigação consistiu na análise das utilizações da Engenharia Didático-Informática por alguns pesquisadores com o objetivo de desenvolver projetos de software. Tivemos como meta organizar os principais indicativos, sugestões, críticas, comentários, etc., oriundos dos projetos que utilizaram a EDI como metodologia para a concepção e construção de software educativo.

No âmbito de realização de pesquisas e investigações, a Engenharia Didático-Informática foi utilizada como metodologia para embasar o desenvolvimento de software

educativo para o ensino e a aprendizagem de saberes matemáticos. Acompanhando essas experiências, foi possível obter informações de suma importância para aprimorar a EDI, observando as limitações e as possibilidades apresentadas pelos pesquisadores no que se refere a utilização efetiva dessa metodologia.

Com isso, apresentamos, na Tabela 12, os pesquisadores e suas respectivas produções utilizando a EDI e detalhamos, em ordem cronológica, no Capítulo 7, como cada investigação trouxe elementos para a evolução dessa metodologia.

Tabela 12 – Desenvolvimento de software com a metodologia da EDI

| PESQUISADORES | SOFTWARE | TIPO DE ESTUDO |
|---|---|---|
| SILVA, 2016 | Function Studium | Dissertação de Mestrado |
| SIQUEIRA, 2019 | Conics Studium 3D | Tese de Doutorado |
| SILVA, 2019 | Magnitude Studium | Tese de Doutorado |

Fonte: o autor.

Analisamos os projetos de software listados na Tabela 12 observando como se deu a utilização da EDI. Todos os projetos foram idealizados no ambiente acadêmico, proporcionando assim uma aproximação maior com os objetos e objetivos de estudo e pesquisa de cada pesquisador e a possibilidade de interações de forma mais rápida entre os desenvolvedores, pesquisadores e envolvidos no projeto de forma geral.

Decidimos por analisar como os pesquisadores utilizaram cada procedimento compreendendo as etapas indicadas na EDI e o quadro de análise utilizado está resumido na Tabela 13.

Tabela 13 – Itens de análise da utilização da EDI em projetos de software

| ITENS | DESCRIÇÃO |
|---|---|
| **Delimitação do Campo** | Intenção de verificar qual a problemática e o contexto da necessidade de desenvolver um recurso tecnológico que foi verificada pelos pesquisadores; identificar qual tipologia de software compreendida como adequada, bem como a especificação de funcionalidades e objetivos com a utilização do software a ser desenvolvido. |
| **Análises preliminares** | Averiguar como os pesquisadores fizeram os levantamentos teóricos considerando as dimensões da EDI (Cognitiva, Epistemológica, Didática e Informática). Investigar quais referenciais teóricos foram adotados pelos pesquisadores a fim de compreender as contribuições que esses trazem para o ensino e a aprendizagem utilizando recursos tecnológicos. |
| **Análise de requisitos** | Compreender como foi realizado o levantamento de requisitos e se esses contribuíram para a criação dos software. Verificar quais foram os métodos utilizados pelos pesquisadores, quais documentações utilizaram e como expressaram esses requisitos para serem compreendidos em linguagem de programação. |
| **Análise a priori e Prototipação** | Observar as hipóteses levantadas para auxiliar o ensino e a aprendizagem com a utilização do software em desenvolvimento; examinar as contribuições do levantamento teórico realizado em confronto com os requisitos e as situações de utilização idealizadas pela equipe. Entender o processo de construção do protótipo também verificando as situações de uso. |
| **Experimentação** | Observar como foi realizada a experimentação: quais os sujeitos, quais as situações, como foram organizadas as sessões, quais as contribuições e limitações da EDI para orientar quanto ao uso do software em desenvolvimento. |
| **Análise a posteriori e Validação** | Entender como o processo de análise da utilização do software foi realizado pelos pesquisadores durante e após a experimentação. Verificar o confronto com as hipóteses e se as situações idealizadas atenderam às expectativas e objetivos. Observar como a validação do software ocorreu, quais métodos foram utilizados e identificar se os objetivos das investigações dos pesquisadores foram alcançados. |
| **Sugestões de implementações** | Verificar as sugestões dadas pelos pesquisadores para implementar/modificar/aperfeiçoar a EDI. Compreender quais os itens da metodologia que carecem de avanços e melhorias para que seja utilizado em outros projetos de desenvolvimento. |

Fonte: o autor.

Considerando as discussões, os estudos e as pesquisas que utilizaram a Engenharia Didático-Informática, percebemos que existe uma evolução em curso do que se construiu ao longo dos anos. Desde as primeiras investigações até as Teses de doutoramento concluídas

utilizando a EDI, percebemos o quanto houve avanços nesse estudo.

Assim, os procedimentos metodológicos dessa pesquisa foram direcionados a evoluir não apenas o modelo de processo da Engenharia Didático-Informática, como de incrementar nessa metodologia contribuições para desenvolver produtos tecnológicos alicerçados em conhecimentos da área do ensino e aprendizagem, bem como as contribuições das tecnologias digitais atuais. A EDI é uma metodologia que tem a preocupação de não apenas criar tecnologias digitais, mas pensar em como essas podem ser utilizadas para efetivamente contribuir com as relações educativas.

À vista disso, apresentamos nos dois capítulos seguintes os resultados da investigação em consonância com a metodologia que fora utilizada.

# 6 RESGATE DE ENGENHARIAS DE PRODUTOS CONSOLIDADOS

Apresentamos neste capítulo os resultados das interações com os líderes das equipes de desenvolvimento dos software Modellus, Casyopée e Function Probe, nesta ordem. Fomos além das respostas dos questionamentos realizados trazendo referências bibliográficas indicadas pelos pesquisadores para obter informações detalhadas sobre os projetos de concepção e criação.

Cada software é descrito de forma separada. Inicialmente, expomos o contexto da elaboração, bem como as funcionalidades básicas e depois compartimentamos os questionamentos em blocos para melhor compreensão das interações e respostas.

Após a análise das entrevistas realizadas com os desenvolvedores dos software aqui discutidos, organizamos nesta seção as principais características das engenharias às quais os produtos foram submetidos para serem criados e colocados em funcionamento. Discutimos nesse capítulo as técnicas, métodos e fundamentos dessas engenharias, bem como as peculiaridades que auxiliam na compreensão da criação de software educativo. As respostas dos desenvolvedores, bem como as literaturas que esses indicaram para a obtenção de informações detalhadas de cada etapa dos projetos de criação desses produtos foram consideradas para melhor compreensão dos processos de desenvolvimento e dar subsídios para aprimorar a EDI.

Como detalhado nos procedimentos metodológicos dessa pesquisa, os questionamentos realizados foram organizados em grupos. No início, fazemos uma apresentação do software e algumas de suas funcionalidades e, posteriormente, elencamos as respostas das interações agrupadas da seguinte forma: Bloco 01 – Questionamentos gerais; Bloco 02 – Equipe de desenvolvedores; Bloco 03 – Objetivos de ensino e aprendizagem; Bloco 04 – Questões técnicas/metodológicas.

## 6.1 MODELLUS

O programa Modellus é um software livre criado para fins educacionais. Inicialmente com foco no ensino de Física e Matemática, estendeu-se, posteriormente, para outras ciências. Foi desenvolvido, a princípio, por Vitor Duarte Teodoro (Professor da Universidade de Lisboa em Portugal) com a colaboração de João Paulo Duque Vieira e Felipe Costa Clérico. De sua primeira versão até as atuais, o programa já passou por várias atualizações. As versões mais usadas são 4.05 e 4.01 (ANDRADE, 2016, p. 47).

Além disso, o Modellus é uma ferramenta de computador que permite alunos e professores realizarem experimentos conceituais usando modelos matemáticos expressos em funções, derivadas, taxas de variação, equações diferenciais e equações às diferenças. Segundo Teodoro (2002), “do ponto de vista computacional, o Modellus pode ser considerado como um micromundo de computador para estudantes e professores, com base em uma metáfora não programática: na ‘Janela Modelo’” (p. 192, *tradução nossa*[13]). Os usuários podem escrever modelos matemáticos, quase sempre da mesma maneira que ele escreveria no papel. Não há um novo idioma a ser aprendido, apenas algumas regras de sintaxe sobre como escrever funções

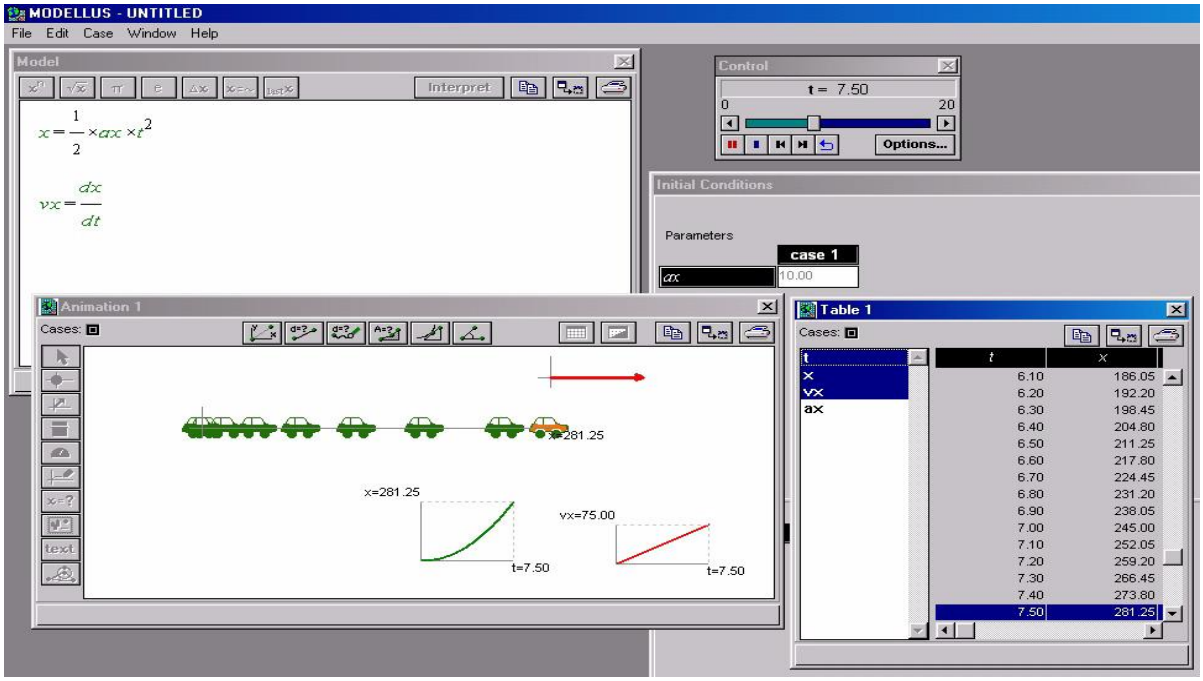
[13] Texto original: From a computational point of view, Modellus can be regarded as a computer microworld for both students and teachers alike, based on a non-programming metaphor: in the “Model window”.

explícitas, equações diferenciais e iterações.

Ainda de acordo com Teodoro (2002), as funcionalidades do Modellus e suas principais características, em síntese, podem ser assim descritas: 1. É uma ferramenta de software para criar e explorar várias representações de modelos matemáticos usando funções, equações diferenciais e equações iterativas; 2. Possui um ambiente de múltiplas janelas. Em uma das janelas, o usuário pode escrever um modelo, escrevendo as equações como estão escritas no papel; em outras janelas, o usuário pode criar e interagir com animações dos modelos, usando objetos abstratos, como vetores e gráficos, ou objetos mais concretos, como vídeos e fotos; 3. A comunicação com o usuário é baseada no conceito de "espelho intelectual"- o software atua como um espelho do que o usuário pensa (TEODORO, 2002).

Quanto às várias representações e às múltiplas janelas, a Figura 2 ilustra “múltiplas representações” de um movimento acelerado no Modellus. No software, de acordo com Teodoro (2002), podem ser representadas, simultaneamente, equações, tabelas, gráficos, trajetórias, estroboscopia e vetores).

Figura 2 – Múltiplas representações no Modellus



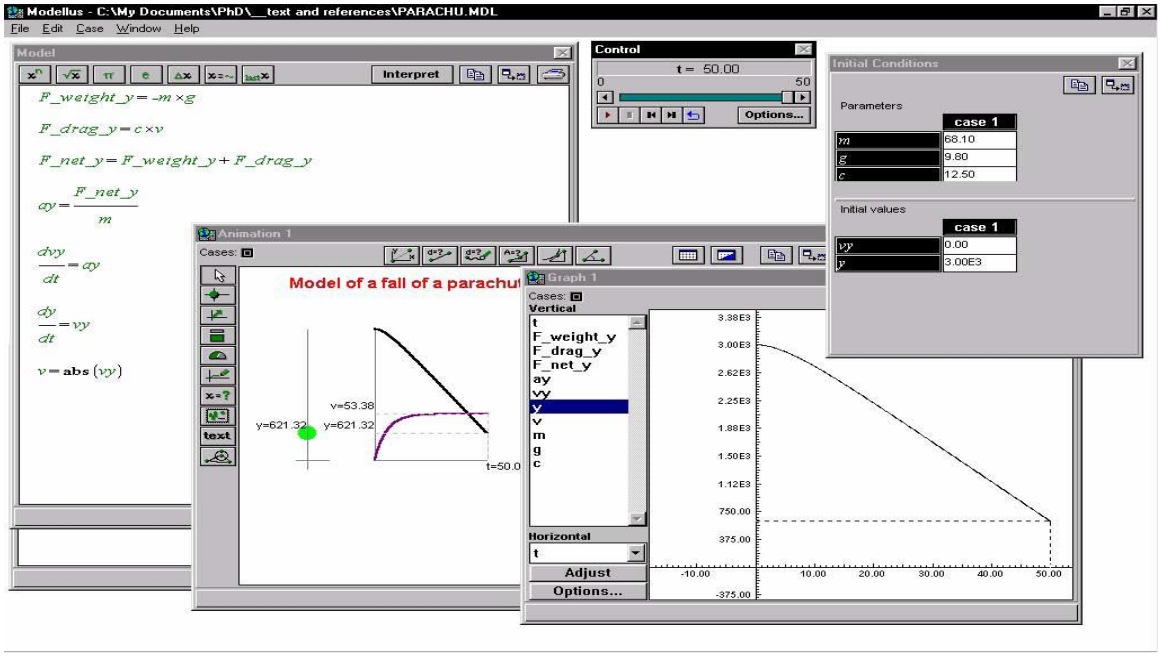

Fonte: Teodoro (2002, p. 24).

Ao utilizar o termo “espelho intelectual”, Teodoro (2002) ilustra esse conceito apresentando o relato da experiência de um estudante ao utilizar o software: “o programa me

deu uma nova visão sobre o movimento que até agora eu só havia alcançado graças à projeção mental" (p. 181, *tradução nossa*[14]). Relatos de outros estudantes mostram o quão importante era para eles explorar múltiplas representações, a fim de tornar a Física mais concreta, no sentido de possibilitar a compreensão de conceitos abstratos de forma a vê-los acontecendo na prática.

A Figura 3 ilustra a situação do salto do paraquedista (questão utilizada rotineiramente no ensino da Física) em que o software Modellus proporciona a compreensão com diversas representações: "Modellus usa uma abordagem completamente diferente da modelagem habitual" (p. 73, *tradução nossa*[15]).

Figura 3 – Abordagem do salto do paraquedista no Modellus




Fonte: Teodoro (2002, p. 73).

Bloco 01 – Questionamentos gerais

O conceito do software surgiu de múltiplas influências: experiência de ensino, experiência em design de ambientes de aprendizado de computador e literatura sobre

[14] Texto original: This program gave me a new vision about motion that until now I only had reached thanks to mental projection. Intrinsically, it is a mirror that reflects our reasoning and makes possible to see it over and over again.
[15] Texto original: Modellus uses a completely different approach to modelling. A model of a parachutist can look like the following in Modellus.

dificuldades de aprendizado. Quanto a experiência em ambientes de aprendizagem, Vitor Teodoro, primeiro líder do projeto de desenvolvimento, destaca que por muitos anos, em sua experiência docente, teve contato direto com os alunos intrigados com o significado, formas e implicações dos modelos matemáticos. Como supervisor e formador de professores, também, por muitos anos, o autor destaca que: os professores (de Física e Matemática) têm muitas dificuldades com os modelos matemáticos. Além disso, tanto alunos quanto professores têm poucas oportunidades de criar objetos em ambientes de aprendizagem, principalmente objetos abstratos, como modelos matemáticos. Assim, o Modellus foi constituído como uma ferramenta para ajudar a mudar essa situação (TEODORO, 2002).

A proposta de Teodoro (2002), quanto a utilização do software Modellus, tal qual a pesquisa realizada sobre a utilização de recursos tecnológicos para o ensino e a aprendizagem, apresentou uma nova perspectiva na aprendizagem e na definição dos currículos de Física, em que o computador e, em particular, o uso do computador como ferramenta de modelação, é considerado como um instrumento chave no processo de aprendizagem. Essa perspectiva assume que:

> (1) a aprendizagem é um processo ativo de criação de significados a partir de representações; (2) a aprendizagem decorre numa comunidade de prática em que os estudantes aprendem a partir do seu próprio esforço e a partir de orientação externa; (3) a aprendizagem é um processo de familiarização com conceitos, com ligações entre conceitos e com representações; (4) as interfaces baseadas na manipulação direta permitem aos estudantes explorar conceitos concreto-abstratos, como é o caso dos conceitos físicos, mesmo quando possuem uma competência reduzida na utilização de computadores (TEODORO, 2002, p. 15).

Bloco 02 – Equipe de desenvolvedores e métodos

A metodologia de desenvolvimento do Modellus, considerou cinco fundamentos, segundo Teodoro (2002):

1. O desenvolvimento é um projeto de equipe, envolvendo diferentes especialistas: pelo menos, designers de software, programadores e professores experientes;
2. O software deve ser projetado após a identificação das experiências de aprendizagem mais relevantes em um determinado domínio. Uma experiência relevante está relacionada ao processo de formação de conceitos, seja porque dá fundamentação para incluir conceitos ou porque mostra uma visão conflituosa com o pensamento ingênuo.

3. Como o aprendizado ocorre em muitas configurações diferentes, o software deve ser validado nas diferentes configurações em que ocorre a aprendizagem: não pode ser projetado apenas para salas de aula. Com a crescente difusão de computadores, em casa, em centros de recursos, em bibliotecas, etc., os alunos podem ter experiências com software exploratório em muitos lugares diferentes, fora das salas de aula e fora da supervisão de professores.
4. O software deve ser baseado em interfaces gráficas e de manipulação direta, nas quais o usuário controla suas ações diretamente, não mediado por comandos escritos.
5. Como todos os software, o design de ambientes exploratórios é um processo iterativo com melhorias sucessivas baseadas em observação do usuário ou estudos de usabilidade e feedback de alunos, professores e desenvolvedores de currículo.

Bloco 03 – Objetivos de ensino e aprendizagem

A enorme literatura sobre concepções alternativas e dificuldades de aprendizagem de ciências específicas e conceitos matemáticos também teve um papel significativo na especificação da concepção do Modellus. Foram considerados os primeiros estudos que identificaram conceitos errôneos persistentes e dificuldades de aprendizado associadas à força e movimento, por exemplo, a outros estudos que mostraram como era difícil para os alunos dar sentido aos gráficos, além de pesquisas que criaram categorias e teorias sobre conceitos errôneos e dificuldades de aprendizagem (TEODORO, 2002).

Para Teodoro (2002), as ferramentas de computador podem ser ferramentas cognitivas poderosas para ajudar a ensinar, a entender e facilitar a mudança conceitual. Para isso, o software deve: lidar com conceitos errados; promover aprendizado ativo e descoberta; usar representações dinâmicas e interativas; permitir simulações desenvolvidas pelos alunos e fornecer ambientes de suporte.

Quanto aos recursos para auxiliar o ensino e a aprendizagem, duas características essenciais do Modellus são as múltiplas representações e a manipulação direta. Várias representações significam que o usuário pode criar, ver e explorar diferentes representações do mesmo modelo. Enquanto a manipulação direta significa que o usuário pode interagir diretamente com as representações, usando o mouse e uma interface gráfica comum, sem a mediação da linguagem escrita.

A importância das várias representações é destacada pelo autor quando enfatiza a construção de conhecimentos pelos alunos: movendo-se entre várias representações mentais,

incluindo formas simbólicas matemáticas, descritivas, experimentais, fenomenais e conceituais, por meio de uma complexa rede de interações, onde se desenvolvem ideias por meio de diálogos, a fim de esclarecer conceitos e estabelecer conexões, assistidas por um professor que assume o papel de orientador para apoiar a aprendizagem cognitiva (TEODORO, 2002).

Conforme Teodoro (2002), o Modellus incorpora modos expressivos e exploratórios de atividades de aprendizagem. Em uma atividade de aprendizagem expressiva, os alunos podem criar seus próprios modelos e criar maneiras de representá-los. Em um modo exploratório, os alunos podem usar modelos e representações feitos por outros, analisando como as coisas se relacionam. Professores, assim como designers e desenvolvedores de currículo, podem usá-lo como uma linguagem de criação para criar representações visuais.

Quanto à atividade docente, Teodoro (2002) afirma que "os professores estão de acordo sobre a importância da modelação na aprendizagem da Física (e da Matemática) e com os aspectos mais importantes das propostas sobre integração da modelação como um componente essencial do currículo" (p. 16). Para o autor, as escolas mudam muito lentamente e há muitas e variadas razões para isso. Os currículos tradicionais, com ênfase na aprendizagem mecânica e de fatos pontuais, apenas podem ser modificados se as escolas tiverem acesso a novas e poderosas visões sobre a aprendizagem e a novas ferramentas que deem suportem à aprendizagem conceitual significativa e que sejam tão comuns e fáceis de utilizar como o papel e o lápis.

Bloco 04 – Questões técnicas/metodológicas

Um dos resultados da pesquisa de construção do software Modellus é a estrutura metódica para projetar recursos tecnológicos digitais do tipo exploratório – como são todos os software discutidos nesse estudo. Esta estrutura é apresentada no Diagrama 12, a seguir, e foi construída utilizando o Diagrama em V de Gowin, possuindo duas linhas de abordagem: uma metodológica e a outra teórica.

Diagrama 12 – Um modelo para orientar o design de ambientes exploratórios de aprendizado de computador para Ciências (e Matemática)


Fonte: Adaptado de Teodoro (2002, p. 121).

Apresentamos os pontos metodológicos do diagrama em V no bloco de questões anterior. Quanto a linha teórica, as três orientações podem ser compreendidas do seguinte modo: o projeto de software dessa categoria (micromundos) deve se basear em pesquisas sobre formação de conceitos, incluindo conceitos errôneos, a fim de identificar experiências relevantes de aprendizagem e fontes de dificuldades na formação de conceitos. Isso garante uma ***base didática*** para o software (TEODORO, 2002).

Uma visão específica da aprendizagem deve ser assumida, ainda assim, Teodoro (2002) acredita que o entendimento completo de um conhecimento nunca será alcançado. Porém, a visão de aprendizagem adotada na concepção do produto auxilia na compreensão do que se pretende aprender. O autor considera que entender ideias científicas é um processo de enculturação. Esse processo é facilitado quando o aprendizado ocorre na "zona de desenvolvimento proximal", conforme Vygotsky. Isso garante uma ***base cognitiva*** para o design das experiências de aprendizado baseadas no software.

Por último, a ciência é assumida como um processo para produzir conhecimento que depende tanto de fazer observações cuidadosas dos fenômenos quanto de inventar teorias para dar sentido a essas observações. A ciência trata de criar, testar e comunicar representações sobre o mundo, como teorias e modelos, especialmente modelos matemáticos. Isso assegura que as experiências de aprendizado baseadas no software devem reforçar a produção de saberes a partir de observações e o teste e aprimoramento de modelos e teorias, em vez da simples apresentação de conhecimento.

Como "saída" do modelo (diagrama em V), sete questões são consideradas (TEODORO, 2002):

1) Esse tipo de software, por si só, tem uso limitado. É necessário que seu uso seja considerado como parte dos pacotes de aprendizagem, a utilização envolve a exploração do que os alunos já sabem, não do que eles não sabem. Materiais escritos, com um design atraente, são essenciais para apresentar linhas de argumentação, orientar as instruções e orientar o trabalho pessoal e em grupo. O software deve servir como um complemento aos livros, permitindo que os alunos explorem o que leem e discutem, oferecendo a eles os recursos que nenhum livro possui.

2) Equilibrar aprendizado exploratório e instrução direta é uma questão fundamental no design de pacotes de aprendizado e na criação de bons ambientes de aprendizado. Pesquisas mostram que a aprendizagem exploratória é difícil. Os professores devem sempre ter em mente que os alunos terão sérias dificuldades em explorar adequadamente, sem orientação docente, o que ainda não sabem. O equilíbrio entre aprendizado exploratório e instrução direta deve ser gerenciado pelo desenvolvedor do currículo e pelo professor.

3) O software exploratório deve suportar a progressão conceitual com base na natureza do assunto e na pesquisa didática. Os usuários devem ser capazes de fazer explorações simples ou mais complexas, sem grandes dificuldades, à medida em que seus conhecimentos aumentam.

4) Os objetos típicos apresentados ao usuário em um ambiente exploratório por computador para aprender ciências são objetos abstratos-concretos. Os usuários podem manipular diretamente vetores, partículas, forças, gráficos, equações, figuras geométricas etc. Esses objetos são concretos apenas no computador, contudo, na realidade, são abstratos, pois são construções mentais, representações mentais de certas propriedades de objetos reais ou imaginários. Os ambientes exploratórios devem permitir que os usuários explorem como esses objetos funcionam, tornando as ações e as consequências das ações transparentes para o usuário.

5) Múltiplas representações são um dos recursos mais importantes do software exploratório. Esse recurso oferece aos alunos a possibilidade de interagir com diferentes representações coordenadas de um fenômeno, como equações, gráficos, animações, vídeos, etc. Independente da importância do uso de múltiplas representações, elas devem ser usadas parcimoniosamente, pois seus abusos podem facilmente levar a sobrecarga de informações nos alunos – múltiplas representações são essenciais para ajudar os alunos a progredir da fidelidade perceptiva (o que observam) à fidelidade conceitual (como a ciência descreve e representa o fenômeno).

6) O software exploratório deve permitir que os alunos concentrem o raciocínio em descrições qualitativas, não em algoritmos e cálculos.

7) O software exploratório deve ter uma interface autoexplicativa: o domínio exploratório deve ser evidente para o usuário iniciante e as funcionalidades do software devem ser facilmente acessíveis sem treinamento formal sobre o uso do software.

Segundo Teodoro (2002), a interface do Modellus segue uma interface padrão do Windows, de acordo com as Diretrizes de Interface do Windows (MICROSOFT, 1995 apud TEODORO, 2002), com pequenas exceções. Essas exceções são: 1. Não é possível fechar ou minimizar janelas, exceto a janela principal, pois cada arquivo do software pode ter várias janelas; 2. É possível ocultar janelas (botão), sem fechá-las; fechar uma janela destrói seu conteúdo; ocultar uma janela preserva o conteúdo da janela; 3. Não é possível imprimir um arquivo: o usuário pode imprimir apenas o conteúdo de cada janela; a melhor maneira de imprimir dados é copiar o conteúdo de uma ou mais janelas para um arquivo de processador de texto, permitindo que o usuário adicione comentários e notas com facilidade aos dados do programa.

Quanto a prototipação, por se tratar de uma evolução, não foram utilizados protótipos para a construção da versão aqui discutida do Modellus.

A validação da versão do Modellus aqui discutida foi realizada com dois grupos de estudantes: um da Educação Básica e outro do Ensino Superior, ambos de Portugal. O primeiro estudo foi realizado com doze alunos do ensino médio que haviam terminado o 11º Ano, reunidos na Faculdade de Ciências e Tecnologia por uma semana, trabalhando em pares em um laboratório de informática.

Esses alunos responderam por e-mail uma chamada para participação em um Curso de Verão sobre Modelagem com Computadores, publicado em um conhecido jornal semanal

português. A maioria dos estudantes possuía um conhecimento bom ou razoável sobre como usar um computador, mas apenas metade deles considerava bom ou razoável seu domínio da Física e da Matemática. As questões de pesquisa deste estudo foram: "Os alunos podem criar seus próprios modelos e animações? Quais vantagens e desvantagens os alunos identificam ao usar o Modellus para aprender modelos matemáticos simples de movimento?" (TEODORO, p. 178, *tradução nossa*[16]).

Os tipos de atividades propostas aos estudantes eram relativamente exigentes. Eles foram convidados a usar modelos em contextos experimentais reais, algo a que não estavam acostumados. Todos os alunos usaram gráficos em tempo real, registro de dados e sensores pela primeira vez. Foi, também, a primeira vez que eles fizeram medições reais para criar modelos de movimento. Todos os alunos foram capazes, com apoio, mas trabalhando sozinhos, em pares, de criar os modelos solicitados, computando escalas de fatores, encontrando parâmetros, criando funções (incluindo funções definidas por partes) e comparando os modelos com dados reais.

Os estudantes consideraram o Modellus fácil de usar, graças à sua interface simples e ao ambiente de aprendizado onde poderiam, facilmente, obter suporte e discutir dificuldades. Ficou, igualmente, claro para os alunos a importância do conhecimento prévio ao usar ferramentas exploratórias como o software. Eles estavam conscientes de que só se poderia fazer uso significativo se tivessem habilidade suficiente no significado dos parâmetros e no uso específico dos diferentes modelos matemáticos (TEODORO, 2002).

O primeiro estudo realizado para a validação do Modellus, de acordo com Teodoro (2002), mostrou que os alunos do Ensino Médio podem começar a criar modelos com o software após uma breve introdução a seus recursos se tiverem conhecimentos prévios da Física e da Matemática necessários para criá-los. Também, mostrou que os alunos reconhecem que esse software pode ser uma ferramenta importante para eles pensarem sobre como a Física descreve o movimento usando modelos matemáticos. Os estudantes não identificaram claramente nenhuma desvantagem significativa – as desvantagens que mencionaram podem ser facilmente superadas com bons ambientes onde a aprendizagem significativa é promovida.

O segundo estudo foi realizado com dez estudantes universitários de graduação que haviam acabado de terminar o terceiro semestre de um bacharelado em Ensino de Ciências

[16] Texto original: Can students create their own models and animations? What advantages and disadvantages do students identify when using Modellus to learn simple mathematical models of motion?

Naturais na Faculdade de Ciências e Tecnologia. Esses alunos foram voluntários para um curso curto de pré-cálculo e cinemática básica que ocorreu no intervalo entre dois semestres. O curso foi oferecido pelo departamento de Educação em Ciências para ajudar os alunos a superar dificuldades nos conceitos básicos necessários para ter sucesso nos cursos universitários de Física e Matemática. Seis estudantes relataram ter conhecimentos básicos ou razoáveis sobre como usar um computador e quatro se consideram fracos ou ruins. As questões de pesquisa específicas para este estudo foram:

> Os alunos podem criar seus próprios modelos e animações? Os alunos concordam que o Modellus pode promover uma abordagem mais integrada da física e da matemática? Os alunos concordam que o Modellus pode ajudá-los a trabalhar mais concretamente com objetos formais? Quais diferenças os alunos identificam ao resolver problemas sem e com Modellus? (TEODORO, 2002, p. 182, *tradução nossa*[17]).

Os resultados deste experimento mostraram, segundo Teodoro (2002), que estudantes de graduação com pouca formação em Física podem usar facilmente o Modellus para criar seus próprios modelos com funções lineares, quadráticas e paramétricas. Os alunos concordaram que o software apoia uma abordagem mais integrada da Física e da Matemática do que na escola. Eles reconheceram a importância do conhecimento anterior para tirar proveito do software. As vantagens estão relacionadas aos recursos de visualização que podem ajudar a melhorar as habilidades abstratas e o raciocínio.

## 6.2 CASYOPÉE

Casyopée é o nome de um software, mas também o de um grupo de pesquisa composto por professores acadêmicos e do ensino médio com foco nas relações de ensino e aprendizagem de funções matemáticas. Os experimentos realizados pelo grupo acompanharam uma reflexão didática sobre o ensino de funções com o uso de uma ferramenta de Tecnologia da Informação e Comunicação para o Ensino – TICE.

Os primeiros trabalhos do grupo estavam orientados em torno de algumas questões: “como usar um software de álgebra computacional para ensinar matemática, quais são os

[17] Texto original: Can students create their own models and animations? Do students agree that Modellus can promote a more integrated approach to physics and mathematics? Do students agree that Modellus can help them work more concretely with formal objects? What differences do students identify when solving problems without and with Modellus?

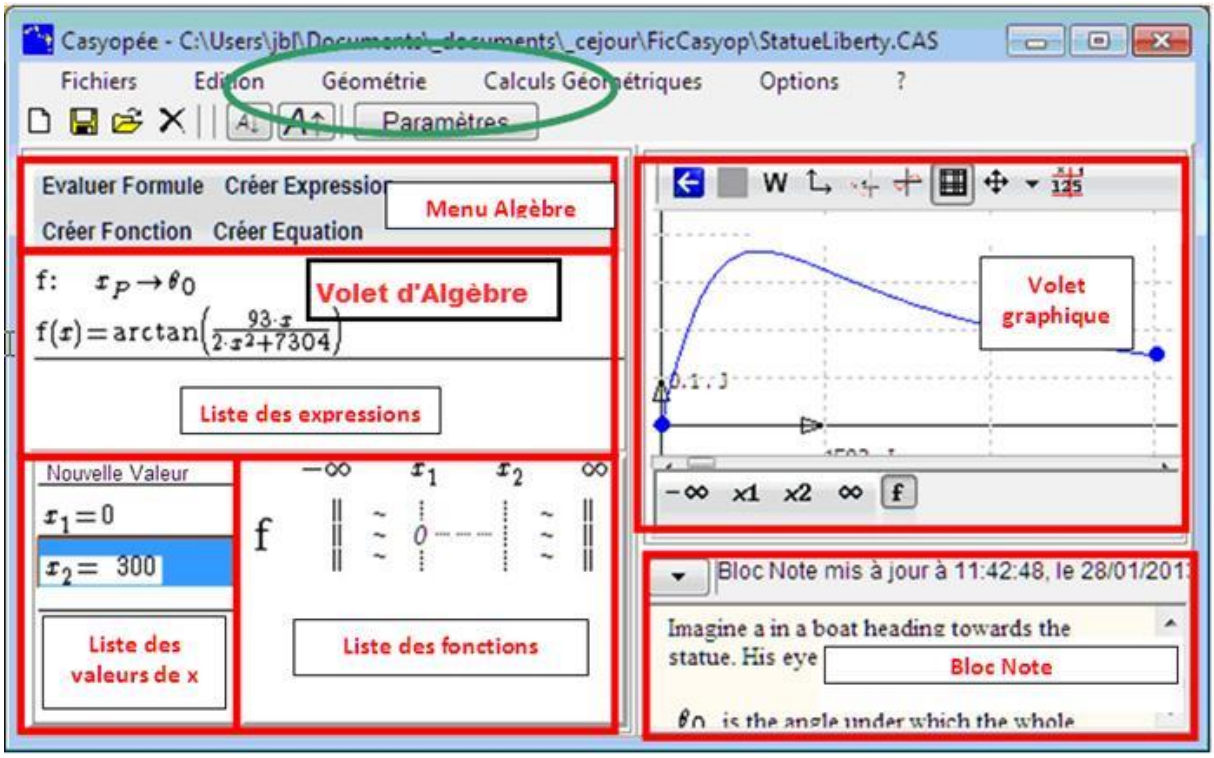
conceitos que facilitarão o aprendizado?" e "Como integrá-lo à atividade dos alunos, como fazer uma ferramenta familiar que eles saberão usar?" (CASYOPÉE, 2019, *tradução nossa*[18]). De acordo com a equipe, as constatações das pesquisas realizadas levaram o grupo a duas conclusões: Os alunos precisam se familiarizar com as linhas de comando do programa que devem aprender a usar e o professor dirige as atividades com muita frequência e deixa pouca iniciativa para os alunos.

Após essas observações, Jean Baptiste Lagrange, Bernard Le Feuvre e Xavier Meyrier decidiram estudar a possibilidade da criação de uma ferramenta tecnológica e a chamaram de Casyopée – **Ca**lcul **sy**mbolique **o**ffrant des **p**ossibilités à l'**é**lève et l'**e**nseignant (Cálculo simbólico oferecendo possibilidades para o aluno e o professor). Nesse software, as ferramentas algébricas não exigem linhas de comando e devem ser semelhantes aos gestos habituais de papel/lápis de alunos. Jean Baptiste Lagrange é o designer e desenvolvedor em colaboração com professores que conhecem os requisitos institucionais da área (CASYOPÉE, 2019).

O Casyopée tem duas janelas principais. A primeira, chamada de janela simbólica, fornece aos usuários recursos simbólicos de cálculo e representação, além de facilidades para a prova/verificação. Já a segunda consiste em uma janela de Geometria Dinâmica - GD. As duas janelas estão intimamente ligadas, ou seja, objetos de uma podem ser totalmente utilizados na outra e o software fornece ao usuário ajuda específica para passar elementos de uma janela para outra (LAGRANGE, 2010), conforme pode ser observado na Figura 4.

[18] Texto original: - Comment utiliser un logiciel de calcul formel pour enseigner les mathématiques, quelles sont les notions dont l'apprentissage en sera facilité ? Comment l'intégrer dans l'activité des élèves, comment faire un outil familier qu'ils sauront utiliser ?

Figura 4 – Interface do Casyopée




Fonte: Cassyopée (2019).

Funções reais de uma variável real são os objetos centrais do Casyopée: uma função é definida por uma fórmula que envolve uma variável de função e um domínio. Como a maioria dos outros sistemas simbólicos relacionados a funções e gráficos numéricos define funções para todo o conjunto de números reais, sem levar em consideração a existência de fórmulas, essa definição é uma característica peculiar desse software.

Bloco 01 – Questionamentos gerais

A construção do software Casyopée, observando a atividade docente, considerou que uma preocupação comum dos educadores matemáticos é que, após os estudos secundários, os alunos geralmente têm uma concepção e um domínio muito ruins da Álgebra. Portanto, eles não têm meios de acessar a riqueza da Matemática que poderia ajudá-los a entender o papel dessa área na sociedade atual (LAGRANGE, 2005).

O projeto Casyopée tem como motivação a utilização da tecnologia para fornecer aos alunos um meio de acessar representações algébricas existentes, construídas por matemáticos nos séculos anteriores. O cálculo simbólico (chamado de Computer Algebra Systems – CAS – Sistemas Algébricos Computacionais), agora, está disponível para o computador ou na

calculadora de todos (LAGRANGE; GELIS, 2008). O projeto Casyopée parte das potencialidades do CAS descritas abaixo:

1) Indo além da mera experimentação numérica e acessando a notação algébrica. Geometria Dinâmica – GD e ,também, planilhas, certamente oferecem aos alunos meios para modelar situações e experimentar, mas o CAS oferece meios expressivos muito mais próximos da notação matemática comum e muito mais poderosos. A álgebra computacional pode ajudar os alunos a ir além da mera experimentação numérica.

2) Focar no objetivo das transformações e não na manipulação. No papel/lápis, são necessárias manipulações algébricas e habilidades transformacionais para obter um determinado formulário, possivelmente ocultando o interesse do resultado, em comparação com o formulário inicial. Os recursos básicos do CAS (expandir, fator…) ajudam os alunos a escolher uma transformação relevante para uma determinada tarefa.

3) Conectando as atividades algébricas. O uso do CAS pelos alunos em uma atividade algébrica experimental, também, ajudaria a articular melhor as diferentes atividades algébricas.

Contudo, mesmo diante das potencialidades citadas, Lagrange e Gélis (2008) compreendem que "o CAS padrão dá pouco apoio ao trabalho dos alunos na solução de um problema e o uso na sala de aula é difícil de dominar pelos professores" (p. 576, *tradução nossa*[19]). Portanto, uma opção central no projeto Casyopée foi desenvolver um ambiente de software incorporando um núcleo simbólico, em vez de usar um CAS padrão. O objetivo foi criar um ambiente de computador aberto em que os alunos possam dominar e, facilmente, vincular à matemática do papel/lápis. Igualmente, visa dar um estatuto claro aos objetos algébricos, ajudando os alunos a se manterem afastados do comportamento errático, frequentemente, observado no uso do CAS padrão e a se concentrarem nos objetos relevantes na solução de problemas.

Bloco 02 – Equipe de desenvolvedores e métodos

A equipe Casyopée, segundo Lagrange (2010), reúne professores e pesquisadores para enfrentar o desafio de ensinar sobre funções no Ensino Médio, de acordo com os currículos

[19] Texto original: In spite of the above potentialities, standard CAS give little support to the students' work when solving a problem, and classroom use is then difficult to master for teachers.

recentes. O grupo está preocupado com o fato de que, embora a tecnologia possa oferecer recursos manipulativos multi-representacionais e simbólicos muito eficazes para resolver problemas e aprender sobre funções, não há nenhuma ferramenta realmente adaptada para o uso dos alunos.

Os software de Geometria Dinâmica oferecem meios para construir figuras operacionais e explorar co-variações e dependências nessas figuras, mas a exploração é limitada a valores numéricos. Os alunos não são incentivados nem ajudados a usar a notação algébrica e a trabalhar em modelos algébricos de dependências geométricas. Os CAS existem para facilitar a manipulação simbólica, mas são projetados para usuários mais avançados e é difícil para os alunos do Ensino Médio reconhecer funções e outros objetos, conforme introduzidos pelo currículo (LAGRANGE, 2010).

A questão abordada pela equipe do Casyopée trata da possibilidade de desenvolver um ambiente de software totalmente consistente com os objetivos do currículo e que poderia ajudar os alunos a experimentar livremente, escolhendo sua própria maneira de resolver e provar. Uma compreensão sobre a utilização de ferramentas tecnológicas é que fazer cálculo formal por cálculo formal é de pouco interesse didático, assim, o grupo trabalha para garantir que as atividades deem sentido ao trabalho matemático. As ferramentas de álgebra computacional devem permitir que os alunos continuem seus esforços e não sejam bloqueados por dificuldades computacionais.

Bloco 03 – Objetivos de ensino e aprendizagem

Quanto aos recursos disponíveis, o Casyopée fornece meios para criar conjuntos de números reais ordenados, possivelmente incluindo parâmetros, a fim de definir domínios. Esses parâmetros podem ser tratados formalmente e numericamente por meio de animação. As restrições podem ser definidas nos parâmetros para se adaptar a todas as situações. Por exemplo, se o parâmetro se destina a modelar uma medida, pode ser definido como positivo. As funções podem depender dos parâmetros. Expressões (isto é, fórmulas que não envolvem uma variável de função, mas possivelmente envolvem parâmetros) também podem ser definidas e tratadas. Assim, Casyopée trata de maneira consistente os objetos algébricos geralmente incluídos nos currículos do ensino médio sobre funções (LAGRANGE; GÉLIS, 2008).

Uma ampla gama de recursos de construção está disponível na janela GD para criar uma figura incluindo pontos livres. Curvas de funções podem ser desenhadas usando a definição

algébrica de funções (domínio e fórmula). O software oferece construções, como a interseção de uma linha e uma curva e a facilidade para exportar funções ou expressões geométricas que não são fornecidas pelos sistemas de GD existentes com base em cálculos numéricos. As medidas podem ser definidas como "cálculos geométricos", possivelmente, incluindo objetos simbólicos (parâmetros, funções, expressões, ...) criados na janela simbólica. O Casyopée pode calcular um domínio e uma fórmula para expressões ou funções "geométricas" relacionadas a medidas, fornecendo a capacidade de expressar dependências algébricas geométricas.

Os alunos podem usar Casyopée como ajuda tecnológica para resolver uma série de problemas típicos do ensino médio, incluindo sinais e variações de funções, problemas de otimização por métodos ou derivações algébricas, interseções de gráficos, encontrar funções com relação a restrições, provar zeros por caminho do Teorema do Valor Intermediário, etc. Em comparação com o ambiente comum de papel/lápis, nenhuma dificuldade de manipulação impede os alunos de abordar todos os tipos de funções e famílias de funções e, portanto, mesmo que o tipo de problemas permaneça o mesmo, os próprios problemas podem ser mais ambiciosos e os alunos exploram mais livremente. A organização do Casyopée é projetada para ajudar os alunos a evitar comportamentos erráticos, concentrando-se em objetos relevantes para problema(s) que ele deseja resolver, dar sentido à exploração e desenvolver métodos (LAGRANGE, 2005).

Quanto à cognição, a principal motivação do Casyopée é a de utilizar a tecnologia com o objetivo de fornecer aos estudantes meios de acessar as representações algébricas existentes, configurando assim a visão dos desenvolvedores sobre como o estudante aprende. Isso posto, o software é fundamentado em algumas premissas (LAGRANGE; GÉLIS, 2008): a) Ir além da mera experimentação numérica e acessando a notação algébrica – não enfatizar apenas um tipo de representação de um objeto matemático; b) Concentra-se no propósito das transformações, e não na manipulação – os estudantes são auxiliados, com o uso do software, a escolher uma transformação relevante para uma determinada tarefa; c) Conecta as atividades algébricas – explorando livremente as transformações algébricas em busca de uma que corrobore sua observação gráfica.

Quanto a algumas dificuldades do ensino e da aprendizagem de funções, Lagrange (2005) considera que a criação de técnicas significativas e compreensão transformacional (diversos tipos de representações) implica práticas de exploração em vários registros (gráficos, numéricos, algébricos...), entrelaçados com o raciocínio e a escrita algébricos. Assim, um dos focos da elaboração do Casyopée era que os alunos pudessem ir além da simples leitura de

propriedades em um gráfico ou uma tabela, o que implicou situações especiais sem evidência numérica ou gráfica, mas também um suporte para abordagens estratégicas de propriedades. Dessa forma, o software foi construído com o intuito de fornecer gráficos e tabelas numéricas ajudando o aluno a usar o raciocínio algébrico e a acompanhar as propriedades conjecturadas ou comprovadas.

Atualmente, o grupo também está desenvolvendo um módulo "Justificativas" integrado ao software. Este módulo está sendo concebido como um auxílio à demonstração e as caixas de diálogo podem ajudar os alunos a construir uma série de justificativas com as propriedades ou teoremas disponíveis no software (CASYOPÉE, 2019).

Bloco 04 – Questões técnicas/metodológicas

É importante ressaltar que as pesquisas atuais para evolução do software Casyopée ocorrem no contexto do projeto ReMath[20], inclinando-se, explicitamente, a trabalhos teóricos coerentes com as pesquisas discutidas nesse contexto. Todavia, a gênese do software é de antes do projeto referido. Sua primeira versão foi criada no grupo Casyopée, porém foi, consideravelmente, enriquecida ao longo das investigações no âmbito do ReMath. Assim, apresentamos aqui os referenciais adotados para a concepção das versões iniciais e das atuais do referido software.

Em suas primeiras versões, a equipe de desenvolvedores teve como objetivo compreender as atividades experimentais dos alunos relacionadas à funções, induzidas por combinações do currículo, práticas e ferramentas, conforme explicitado anteriormente. Para isso, Lagrange (2005) utilizou de estruturas teóricas para analisar essa atividade. Primeiro, a natureza das atividades e sua relação com a evolução do conhecimento matemático devem ser analisadas do ponto de vista didático e epistemológico. Então, analisando sua evolução, especialmente em relação às abordagens experimentais, é necessária outra estrutura para trazer alguma compreensão das mudanças nos currículos e práticas docentes. Por último, a introdução de ferramentas tecnológicas implica em mais duas estruturas: uma responsável pelos vínculos entre os saberes dos alunos sobre a ferramenta e suas compreensões nesta atividade e outra considerando o design de ferramentas e sua influência na relação de ensino e aprendizagem.

Tem-se, dessa forma, quatro questões vinculadas: 1. análise didática e epistemológica;

[20] Remath é um projeto europeu que associa seis laboratórios na Itália, Grã-Bretanha, Grécia e França. O objetivo do projeto é entender melhor como a questão das representações é conceituada em diferentes estruturas e contextos e como elas influenciam o "design" da pesquisa e a interpretação das observações (LAGRANGE et al, 2011).

2. mudanças nos currículos e práticas; 3. relacionamento entre ferramentas e Matemática; 4. design de software. Lagrange (2005) se refere a essa abordagem como "multidimensional", afirmando que muitos estudos ou relatórios de inovação sobre tecnologia no ensino de Matemática não são relevantes quando consideram apenas uma estrutura. Resumimos a seguir como o desenvolvimento do Casyopée fora realizado seguindo a abordagem proposta.

*Análise didática e epistemológica* – Lagrange (2005) considerou, observando os estudos de Kieran (2001), que o conhecimento inicial sobre funções é uma das duas principais abordagens usadas na maioria das aulas de Álgebra para fornecer significados à atividade algébrica, sendo a outra aritmética generalizada. Essas duas abordagens fornecem uma linha transversal única para três categorias de atividade algébrica: geracional, transformacional e global.

*Mudanças nos currículos e práticas; relacionamento entre ferramentas e matemática* – Segundo Lagrange (2005), "a abordagem praxeológica (Chevallard, 1985, 1994, 1999) visa dar conta das condições em que os objetos matemáticos existem e vivem em instituições ou, mais precisamente, como eles são 'conhecidos e entendidos' como entidades decorrentes de práticas" (p. 149, *tradução nossa*[21]). Lagrange (2005) pondera que a palavra "instituição" deve ser entendida em um sentido muito amplo, pois qualquer prática social ou cultural ocorre dentro de uma instituição – considerando então instituições de pesquisa científica dedicadas à produção de saberes e instituições didáticas dedicadas à aprendizagem.

Outro referencial teórico considerado foi a abordagem instrumental (GUIN e TROUCHE, 1998; TROUCHE, 2000 apud LAGRANGE, 2005) – "deriva da conceituação sobre tecnologias na vida social que distinguem um artefato tecnológico e o instrumento que um ser humano é capaz de construir a partir desse artefato" (p. 151, *tradução nossa*[22]). Enquanto o artefato se refere à ferramenta objetiva, o instrumento se refere a uma construção mental da ferramenta pelo usuário. O instrumento não é fornecido com o artefato, é construído em uma gênese instrumental complexa e molda a atividade e o pensamento matemáticos (LAGRANGE, 2005).

De acordo com Lagrange (2005), a dimensão instrumental é importante para a conceituação matemática. Como o ensino de matemática, geralmente, funciona em um

[21] Texto original: The praxeological approach (Chevallard, 1985, 1994, 1999) aims to give account of the conditions in which mathematical objects exist and live in institutions or more precisely how they are 'known and understood' as entities arising from practices.
[22] Texto original: It derives from conceptualisation about technologies in social life distinguishing a technological artefact and the instrument that a human being is able to build from this artefact.

ambiente sem diversidade de tecnologias digitais, a introdução desses recursos muda radicalmente esse ambiente. Assim, o autor orienta que o ensino deve considerar a transformação das ferramentas tecnológicas pelos alunos em instrumentos matemáticos e os processos associados de instrumentação - integrando o instrumento nos processos de pensamento e instrumentalização do usuário - descobrindo as funcionalidades da ferramenta, inventando usos práticos (LAGRANGE, 2005).

Por último, sobre a abordagem multidimensional proposta por Lagrange (2005), *o design de software* – o autor admite que há evidências encorajadoras sobre o impacto de vários recursos específicos de software na exploração e, também, evidências desencorajadoras sobre o trabalho com software educacional que nem sempre atua como a ideia do desenvolvimento que foi projetado para ser. A problemática é a visibilidade das intenções educacionais no design de software. Para o autor, o papel dos designers é o de perceber e articular suas decisões talvez inconscientes e transformá-las em considerações conscientes de design, ajudando os professores a se concentrarem nas propriedades e mensagens mais refinadas das ferramentas que usam em suas salas de aula.

Yerushalmy (2001, apud LAGRANGE, 2005) destaca três questões nas quais o design de software deve tornar visíveis as intenções educacionais. O design da ferramenta pode, primeiro, dar ao aluno um controle sobre a exploração, ajudando-o a desenvolver métodos. Também, poderia apoiar a organização do currículo sendo consistente, usando a mesma linguagem de objetos e ações que formam a grade ao longo da qual o currículo é mapeado.

Por fim, ela lamenta a confusão sobre o papel das ferramentas de solução, como calculadoras de quatro operações e CAS: como os resultados fornecidos por essas ferramentas geralmente são 'atalhos' para o que os alunos devem aprender, eles não são fáceis de conectar à aprendizagem explícita. Em vista disso, para tornar visível a exploração, o software deve entregar uma mensagem clara sobre o papel (educacional) da tecnologia. Essas três questões ajudam na construção de métodos dos alunos, suporte à organização do currículo e mensagem clara sobre as metas de aprendizado (LAGRANGE, 2005).

Contudo, ao inserir o Casyopée no projeto ReMath, foi iniciada uma nova fase: houve o aperfeiçoamento do software. De acordo com Lagrange, Artigue, Cazes, Gélis, e Vandebrouck, (2011), com o registro do software no projeto, houve uma evolução fortemente guiada pelos referenciais teóricos a partir de então considerados.

> Estruturas, como a teoria das situações didáticas (Brousseau 1997), a abordagem antropológica do didático (Chevallard 1992) ou a abordagem instrumental (Rabardel 1995), participaram da orientação geral do ambiente antes mesmo do projeto ReMath. O desenvolvimento dentro da estrutura do projeto implicou considerações mais particulares de representações e interações com representações. Para isso, contamos com outros três quadros teóricos (LAGRANGE et al, 2011, p. 74-75, *tradução nossa*[23]).

Os três quadros teóricos considerados atualmente no contexto da evolução do Casyopée, podem ser assim resumidos (adaptado de Lagrange et al, 2011):

A ***concepção da noção de função***, vista como co-variação entre medidas ou quantidades. Radford (2005) propõe situações nas quais o conceito de função se baseia na experiência de dependências dentro de um sistema físico e é construído a partir das variações mútuas de seus objetos. Arzarello et al. (2004) aborda a noção de função no contexto de movimentos de elementos materiais medidos por meio de sensores conectados a uma calculadora. A geometria dinâmica (Falcade, 2007), também, oferece a possibilidade de experimentar a co-variação de quantidades. Comin (1999) tenta mostrar a importância, no nível epistemológico, da introdução de funções a partir da dependência entre quantidades (no sentido de que qualquer variação de uma leva à variação do outro), abordagem explicitamente mencionada nos programas do ensino médio e seus comentários.

***Registros de Representações Semióticas*** (DUVAL, 1995). Esses registros garantem as seguintes atividades: (1) a formação de representações; (2) seu processamento dentro do registro; (3) converter um registro em outro registro representativo. Duval afirma que a capacidade de conversão entre registros desempenha um papel indispensável na conceituação.

***Tipos de atividades*** (KIERAN, 2007) propôs uma classificação das atividades algébricas em três categorias: geracional, transformacional e global/meta-nível. O primeiro tipo de atividade diz respeito à formação dos objetos básicos de expressões e equações. O segundo diz respeito à produção de novos objetos algébricos a partir de regras como o agrupamento de termos semelhantes, expansão e fatoração. O último tipo de atividade é caracterizado pelo fato de as

[23]Texto original: Des cadres, comme la théorie des situations didactiques (Brousseau 1997), l'approche anthropologique du didactique (Chevallard 1992) ou l'approche instrumentale (Rabardel 1995), ont participé à l'orientation générale de l'environnement avant même le projet ReMath. L'évolution dans le cadre du projet impliquait la prise en compte plus particulière des représentations et interactions avec les représentations. Nous nous sommes appuyés pour cela sur trois autres cadres théoriques.

funções aparecerem como ferramentas de solução. Essa estrutura já havia sido proposta para classificar as atividades algébricas dos alunos na versão do Casyopée existente antes do ReMath.

Além do robusto referencial teórico adotado para as melhorias do Casyopée, uma característica a ser considerada no ciclo de vida do software é a experimentação contínua com usuários e os retornos dessas experimentações para criar novos recursos e funcionalidades. Em uma das utilizações, houve a percepção da urgência de criar um novo recurso, segundo Lagrange (2005),

> Neste experimento, os alunos tenderam a perceber os gestos correspondentes como mais restritivos do que a prova comum de papel / lápis. Na versão de Casyopée desenvolvida neste momento, os alunos tinham que procurar em um menu o item relevante e preencher duas caixas de diálogo sucessivas, enquanto a razão do máximo estava clara em uma forma fatorada de uma função. Não obstante, pensamos que esses gestos são importantes para que a resolução simbólica ajude à conceituação. Desenvolvendo ainda mais o ambiente, criamos desvios para que Casyopée faça parte da prova quando solicitada pelo aluno e permitida pelo professor (LAGRANGE, 2005, p. 176-177, *tradução nossa*[24]).

Outra situação ilustra a preocupação com os retornos oriundos da experiência do usuário. Em um dos problemas propostos na experimentação, foi percebido que era necessário articular elementos da Álgebra com a Geometria (problema do retângulo de área máxima – Lagrange, 2005), assim, afim de proporcionar uma transição da percepção geométrica para a modelagem algébrica, a equipe começou a considerar a possibilidade de diversificar a maneira como os usuários podem introduzir uma função: com uma fórmula algébrica, mas também a partir de definições geométricas.

## 6.3 FUNCTION PROBE

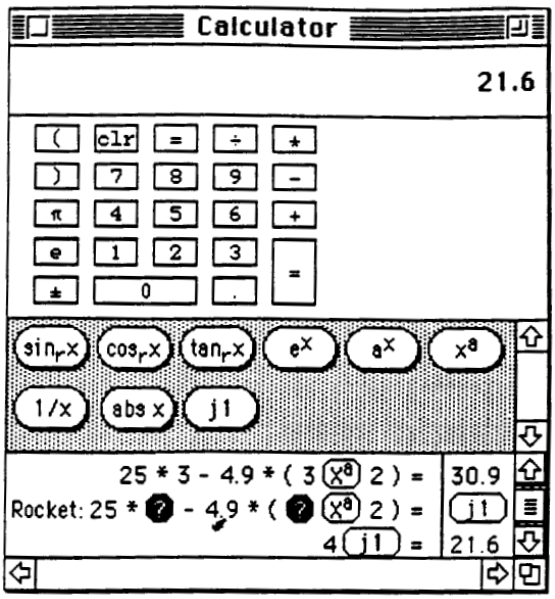
O Function Probe é uma ferramenta multirepresentacional para aprender funções. Ao chamá-la de ferramenta, Confrey e Smith (1992) indicam que o software se destina a ser usado ativamente pelos estudantes em seus esforços de resolução de problemas. Com isso, o Function

[24] Texto original: In this experiment students tended to perceive the corresponding gestures as more constraining than ordinary paper/pencil proof. In the version of Casyope´e devel oped at this time, students had to look into a menu for the relevant item and fill two successive dialog boxes whereas reason for the maximum was clear on a factorised form of a function. We nevertheless thought that these gestures are important in order that the symbolic resolution helps conceptualisation. Further developing the environment, we created bypasses making Casyopée do a part of the proof when requested by the student and allowed by the teacher.

Probe não é um tutorial nem está vinculado a materiais curriculares específicos, mas sim projetado para ser compatível com ações e representações que os alunos criam e usam em uma variedade de situações problemáticas que eles podem encontrar enquanto estudam Matemática (CONFREY; SMITH, 1992).

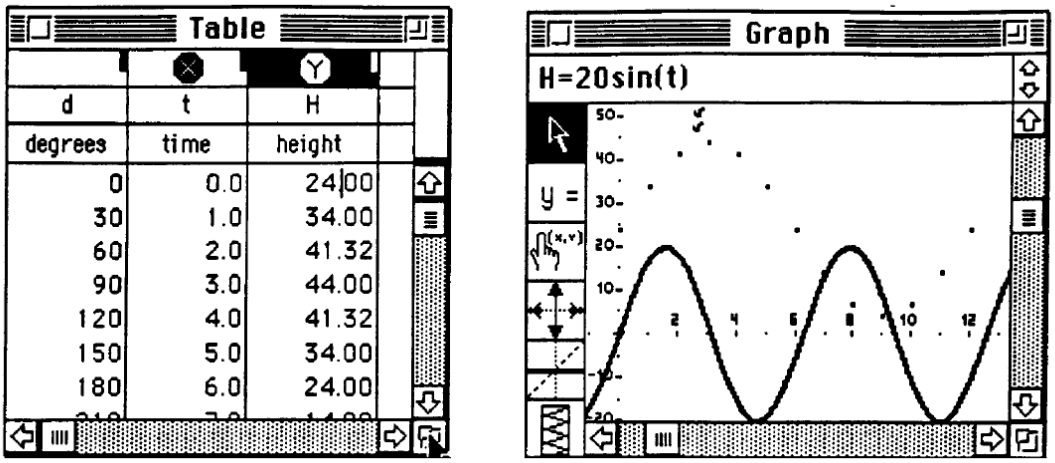
A interface do software é composta por três janelas integradas entre si: a janela gráfica, a janela de tabelas e a janela da calculadora. Cada uma dessas permite ao usuário realizar comandos dependentes assim como independentes das demais, conforme pode ser observado nas figuras 5 e 6.

Figura 5 – Janela da calculadora




Fonte: Confrey e Smith (1992).

Figura 6 – Janela gráfica e tabular




Fonte: Confrey e Smith (1992).

A janela gráfica possibilita a inserção de vários gráficos, mantendo um histórico das funções inseridas e esboça funções e pontos. Translações, reflexões e alongamentos podem ser executados pelas ações do mouse diretamente nos gráficos e seu impacto na equação é exibido, se desejado. A janela da tabela atua como uma planilha personalizada para relações funcionais. Marcações informais e formais podem ser inseridas; as colunas podem ser preenchidas, classificadas e vinculadas; e o programa calcula e exibe as primeiras diferenças e proporções e acumula valores das colunas (CONFREY, 1992).

Referente a janela da calculadora, Confrey (1992) observa a semelhança com uma calculadora científica padrão e pondera que embora existam semelhanças, essa janela exibe a notação de pressionamento de tecla, que pode ser examinada para ver resultados totais e parciais e permite capturar relações funcionais criando botões individuais. Todas as janelas podem ser usadas para definir funções que podem ser exibidas nas demais janelas. Além disso, os pontos podem ser enviados da tabela para o gráfico e vice-versa.

Bloco 01 – Questionamentos gerais

O contexto de investigação e concepção deste software tem cerca de 15 anos. Originalmente, foi desenvolvido para apoiar um curso de pré-cálculo na Cornell University e também aulas de Matemática na Apple Classroom of Tomorrow – ACOT[25], em Columbus-

[25] Texto original: é um projeto de pesquisa que explora o aprendizado quando crianças e professores têm acesso imediato a tecnologias interativas (APPLE, 2020) - http://www.appleclassrooms.com/apple-classrooms-of-tomorrow/.

Ohio. Este software foi projetado para apoiar o pensamento, a exploração e compreensão de famílias de funções, incluindo lineares, quadráticas, exponenciais, polinomiais, racionais e trigonométricas. Foi desenvolvido para permitir que os alunos possam explorar a aparência e o comportamento contrastantes e complementares dessas funções usando diferentes representações. O software foi projetado para responder às próprias inclinações dos alunos para agir em ambientes de solução de problemas aplicados e permitir que eles explorem os resultados de suas ações (CONFREY; MALONEY, 2008).

O Function Probe foi construído através de várias interações de design e pesquisa que serviram para esclarecer como os alunos pensavam sobre funções, como poderiam aprender funções de maneiras aprimoradas e, portanto, para fornecer informações de design dos recursos do software. Ao discutir essas interações entre o design do software e a pesquisa, Confrey e Maloney (2008) afirmam que o próprio conhecimento do conteúdo dos participantes da pesquisa cresceu como pesquisadores e formadores de professores, tanto em termos de pedagógicos quanto em termos de domínio generativo.

Ambos os tipos de conhecimento são elementos necessários para orientar e auxiliar o trabalho docente. A distinção entre eles é que o conhecimento pedagógico do conteúdo se refere ao conteúdo necessário para saber como ensinar um tópico, sua pedagogia, didática, entre outros, enquanto o domínio generativo se refere a como o próprio conteúdo precisa ser reestruturado à luz de novas abordagens da tecnologia, da aprendizagem e aplicações da matemática – daí o termo "generativo".

Por exemplo, a compreensão do conteúdo pedagógico é necessária para orientar os professores na escolha de exemplos, antecipando estratégias e conceitos errôneos dos alunos e gerando explicações convincentes e compreensíveis. O domínio generativo se refere à forma como o conhecimento do conteúdo de uma pessoa é transformado, substancialmente, no contexto da solução de problemas e de múltiplas representações. Essa transformação é necessária para que os professores se tornem flexíveis e adaptáveis, reconheçam abordagens inovadoras dos alunos, gerem e identifiquem novas aplicações e possibilidades representativas e vinculem tópicos matemáticos de novas maneiras (CONFREY; MALONE, 1992).

Bloco 02 – Equipe de desenvolvedores e métodos

O trabalho de design coincidiu com um campo de pesquisa muito ativo para produzir oportunidades para experimentar novas abordagens e avaliar os resultados. Um grupo de

talentosos estudantes de pós-graduação realizou uma variedade de estudos que tornaram possíveis essas investigações. O Function Probe foi construído através de várias rodadas de design e pesquisa, que serviram para esclarecer como os alunos pensavam sobre funções, como podem aprender sobre funções de maneiras aprimoradas e, portanto, para informar as rodadas subsequentes de design dos recursos do software (CONFREY; MALONEY, 2008).

Bloco 03 – Objetivos de ensino e aprendizagem

Esse software possui uma série de problemas que apoiam a compreensão dos alunos sobre diferentes famílias de funções que são extraídos e expandidos a partir de problemas no aprendizado de funções e de resolução de problemas. O software foi utilizado em muitas faculdades e instituições de ensino nos Estados Unidos da América e em países ao redor do mundo.

A construção do Function Probe é o resultado de quase cinco anos de projeto, teste, redesenho e novos testes. Esse processo surgiu de três compromissos básicos em relação ao ensino de Matemática no ensino médio, de acordo com os autores:

> 1) a crença de que o currículo deve ser orientado por problemas contextuais; 2) a crença de que aprendemos a ser melhores professores ouvindo e procurando os métodos dos alunos enquanto trabalham nos problemas; e 3) a crença de que aprender a trabalhar e coordenar múltiplas representações é uma parte essencial da construção de concepções matemáticas fortes e viáveis (CONFREY; SMITH, 1992, p. 60, *tradução nossa*[26]).

Quanto ao desenvolvimento do software os três compromissos citados podem ser detalhados do seguinte modo (CONFREY; SMITH, 1992):

1) ***Um currículo baseado em contexto*** – Na experiência dos autores em ministrar cursos de pré-cálculo no ensino médio e superior, foi percebido que ao criar problemas contextuais para os alunos era possível envolvê-los com experiências e atitudes diferentes de atividades matemáticas significativas e que, também, pareciam capturar aspectos importantes dos conceitos matemáticos básicos. Os problemas

[26] Texto original: 1) the belief that curriculum should be driven by contextual problems; 2) the belief that we learn to be better teachers by listening to and looking for students' methods as they work on problems; and 3) the belief that learning to work with and coordinate multiple representations is an essential part of building strong and viable mathematical conceptions.

eram projetados para apresentar aos alunos funções dentro de um contexto. Os problemas contextuais provaram ser bem-sucedidos em incentivar os alunos a relacionar a matemática com sua experiência cotidiana e a aprender a reconhecer quando uma função específica pode ser útil na modelagem de uma situação.

2) ***Ouvindo os métodos dos alunos*** – Uma parte importante da abordagem no ensino de Matemática é a concepção de que os alunos compreendem melhor os conceitos quando esses fazem sentido dentro dos contextos de sua própria experiência. Assim, o ensino da Matemática envolve, necessariamente, aprender efetivamente para ajudar os alunos a aproveitar suas experiências, em vez de lhes impor ideias que eles não têm como relacionar com seus próprios conceitos. O Function Probe foi desenvolvido após um estudo cuidadoso das maneiras pelas quais os alunos abordam os problemas. Durante o processo de design, foi verificada a possibilidade de recriar uma variedade de métodos que foram percebidos observando os alunos na solução de problemas e incorporando ao programa a flexibilidade necessária para permitir essas ações e métodos de representação.

3) ***Coordenação de múltiplas representações*** – Para muitos estudantes, resolver um problema matemático é sinônimo de encontrar uma equação e resolvê-la. Reduzir a riqueza da Matemática a um processo de obtenção de equação é uma injustiça tanto para o assunto quanto para os alunos. Aprender a entender como as ideias matemáticas podem ser representadas em vários meios e chegar a entender as relações entre essas formas de representação não apenas cria a emoção de aprender matemática, mas é imprescindível para que os alunos construam ideias matemáticas que serão viáveis fora da escola. Embora o Function Probe seja considerado como uma ferramenta multirepresentacional, na medida em que incorpora pressionamentos de teclas, tabelas, gráficos e equações da calculadora como formas de representação, os autores acreditam que é apenas um primeiro passo para criar ambientes de computador que permitam e incentivem os alunos a criar livremente maneiras de representar relações matemáticas.

Segundo Confrey e Smith (1992), por trás desses compromissos está a convicção de que a Matemática é um empreendimento humano. “Cada indivíduo constrói suas ideias matemáticas executando ações que resolvem situações problemáticas em sua vida e refletindo sobre os

processos pelos quais esses problemas são resolvidos" (p. 60, *tradução nossa*[27]). Combinar o processo pessoal de reflexão com o processo social de comunicar essas ideias e conceitos a outras pessoas é uma parte considerável para se tornar um membro ativo e produtivo da sociedade.

Quanto aos recursos disponíveis para auxiliar o ensino e a aprendizagem, o Function Probe possui três janelas: uma janela de gráficos, uma calculadora e uma janela de criação de tabelas, conforme citado anteriormente. Cada janela é projetada para incentivar o aluno a operar ativamente de maneiras compatíveis com as respectivas representações. Além disso, os alunos podem transmitir funções e dados entre as representações, permitindo que eles desenvolvam flexibilidade e discernimento na compreensão dos conceitos. Para conveniência do aluno e do professor, é mantido o histórico em cada janela das ações que o aluno realizou (CONFREY; SMITH, 1992).

Organizamos uma síntese descritiva dos recursos de cada janela segundo Confrey e Smith (1992):

*A janela da tabela* – é semelhante a uma planilha com a importante exceção de que os relacionamentos funcionais são entre colunas e não células individuais. O software permite a entrada de valores de dados individualmente ou preenchendo iterativamente uma coluna por adição, subtração, multiplicação ou divisão. Dessa forma, muitas relações funcionais podem ser representadas preenchendo colunas, em vez de exigir equações. As colunas podem ser nomeadas com rótulos formais e informais. Os rótulos formais devem ser variáveis únicas que podem ser usadas para construir relações funcionais. Etiquetas informais podem ser estendidas conforme necessário. Os alunos podem classificar colunas e examinar as diferenças e a proporção de valores sucessivos.

*A janela gráfica* – pontos discretos ou equações contínuas podem ser inseridos nessa janela. Nos dois casos, a relação funcional é mantida, permitindo que os conjuntos de pontos/equações sejam traduzidos, esticados e/ou refletidos como funções. Um registro é projetado para acompanhar a magnitude e a direção das transformações e essas informações são armazenadas no histórico. Um ícone de ponteiro pode ser usado para ver as coordenadas dos pontos ou para inserir novos pontos. Vários gráficos podem ser representados e selecionados de forma independente. As funções também podem ser inseridas algebricamente ou importadas

[27] Texto original: Each individual constructs her mathematical ideas by taking actions which resolve problematic situations in her life and reflecting on the processes by which those problems are resolved.

de outras janelas.

*A janela da calculadora* – além de ter a funcionalidade de uma calculadora científica básica, essa janela foi projetada para incentivar a visualização de funções como generalizações de procedimentos. Os procedimentos, representados por pressionamentos de tecla da calculadora, podem ser transformados em um "botão", selecionando a sequência desejada no histórico de pressionamentos de tecla. Os botões podem ser acionados na calculadora pressionando uma única tecla ou "importados" da janela gráfica ou da tabela.

Bloco 04 – Questões técnicas/metodológicas

Dois dos princípios de design foram cruciais para o ciclo de vida do software. Primeiro, realizou-se um estudo cuidadoso do uso, pelos estudantes, dos protótipos de software e tarefas e revisaram-se os projetos para responder às necessidades documentadas dos alunos – uma "abordagem de design centrado no aluno". Segundo, foram construídas representações que apoiavam de forma independente a busca de ações matemáticas consistentes com a funcionalidade da representação. Isso levou a equipe a desenvolver ferramentas únicas e gerando possibilidades matemáticas interessantes dentro de uma epistemologia de múltiplas representações (CONFREY; MALONEY, 2008).

Com isso, percebe-se que o Function Probe é baseado em um design centrado no aluno, o que significa que há uma observação cautelosa nas ações que os alunos executam ao resolver problemas e essas observações são incorporadas ao design do software. Porém, o aluno não é o único meio de obter quais as funcionalidades e recursos que o programa deve conter (CONFREY, 1992). Os princípios fundamentais de design são:

- ***Cada representação deve ter sua própria integridade*** – Procurou-se evitar dependências desnecessárias entre representações. Exemplificando: deve-se ser capaz de realizar transformações nos gráficos agindo diretamente sobre eles, em vez de ser necessário fazer as alterações de maneira alfabética, como na maioria dos outros software funcionais, da época. Além disso, procurou-se usar designs que permanecessem relativamente fiéis à ação. Assim, as transformações do gráfico são realizadas através de uma ação do mouse projetada para recordar a sensação física de realmente esticar ou transladar o gráfico.

- ***Distinguimos entre decisões curriculares e de design*** – O software não está vinculado a uma forma ou sequência específica de um currículo. Se uma ação é legítima em uma representação - por exemplo, refletindo uma função quadrática em torno de seu vértice, em vez de em torno do eixo, permitimos essa ação e assumimos que ela deveria ser abordada como elemento do currículo.
- ***É essencial fazer distinções cuidadosas entre projetar uma ferramenta de aprendizado e projetar uma ferramenta de um especialista*** – Embora questões curriculares específicas não tenham sido incluídas no design, certas questões pedagógicas desempenharam parte integrante de seu design. Em particular, é preciso, sempre, traçar uma linha entre os recursos que aprimorarão os esforços construtivos de um aluno em explorar e resolver problemas e aqueles que podem levar, restringir ou negar a oportunidade de tal exploração. Por exemplo, é importante ser deliberado sobre a escolha de automação. Uma opção é tornar a automação progressiva. No projeto do software, não foram automatizados os processos que os alunos devem realizar para entender os conceitos que desejava-se ensinar.
- ***Cada representação deve incorporar recursos que tornem disponíveis os tipos de ações que os alunos tem maior probabilidade de usar –*** Quando os alunos demonstraram, em experimentos, que desejavam agir de uma certa maneira, como a criação de funções preenchendo duas colunas, foram criados métodos para permitir isso.
- ***O significado que os alunos constroem para o conceito de função variará entre as representações –*** A capacidade de construir esses significados, de forma independente, deve ser protegida. Por exemplo, existem incompatibilidades entre a notação de pressionamento de tecla e a notação algébrica; o sinal de igual pode ser usado várias vezes na notação de pressionamento de tecla e tem o significado de "avaliar", em vez de "igual". A equipe permitiu que eles coexistissem.
- ***A passagem entre representações de funções é um meio importante para incentivar o entrelaçamento de ideias de funções –*** O software permite definir uma função em qualquer representação. Os alunos podem passar por diversas representações através do procedimento "Enviar" ou "Ligar".

- ***É preciso equilibrar possíveis limitações da notação tradicional com o risco de introduzir novas formas*** **–** Usamos uma chave binária intitulada $a^x$ com o inverso do $\log_a x$. Nem a chave, nem a sua inversa, eram limitadas em suas bases. Como resultado, a calculadora não se limitou a registros naturais ou comuns.
- ***O retorno mais direto em várias representações é através da convergência ou disparidade*** **–** Atualmente, o feedback dos alunos é dado através da convergência, ou falta dela, nas várias representações.
- ***Um histórico do uso de cada representação é mantido e está disponível para os alunos*** **–** Os alunos podem se beneficiar aprendendo a refletir sobre suas ações; assim, o software foi projetado para registrar o histórico que está sempre disponível para os alunos. Dados resumidos sobre o uso das janelas, sua ordem e duração, também, podem ser registrados para fins de pesquisa.

Dessa maneira, conclui-se a análise da engenharia do Function Probe, bem como as dos outros software discutidos até aqui. Apresentaremos, na sessão a seguir, algumas das características dessas engenharias e como a realização dessa abordagem histórica apresentou elementos para o aperfeiçoamento da EDI.

## 6.4 SÍNTESE DAS ENGENHARIAS UTILIZADAS

Em resumo, as engenharias que foram idealizadas e postas em prática para a criação dos software Casyopée, Function Probe e Modellus, tem características semelhantes e, igualmente, algumas particularidades. Um dos pontos de aproximação, por exemplo, é que todos os software foram criados no âmbito acadêmico em grupos de pesquisas com professores, pesquisadores e estudantes dos diversos níveis colaborando com os projetos. Sintetizamos, nos parágrafos que seguem, algumas das características desses projetos.

Nos projetos analisados, os pesquisadores responsáveis notavam, através de suas vivências como professores da Educação Básica e Superior, dificuldades quanto ao ensino e a aprendizagem de conhecimentos da Matemática e, também, da Física. No Casyopée, por exemplo, existiu, inicialmente, a tentativa de contribuir com o ensino e a aprendizagem de Álgebra e funções matemáticas ao se observar as possibilidades de utilização de tecnologias tanto para o Professor quanto para o estudante. Já no projeto Modellus, saberes da Física e da Matemática eram o foco, no momento em que o líder do projeto, analisando sua experiência docente, percebeu a dificuldade dos estudantes na compreensão de conceitos abstratos dessas

ciências bem como a dificuldade dos professores no ensino dos objetos de ambas as áreas.

O Function Probe, por sua vez, fora elaborado, também, com objetivo de auxiliar as relações didáticas no âmbito de conhecimentos matemáticos. Entretanto, não nasceu de dificuldades observadas e sim com a finalidade de auxiliar estudantes na realização das atividades de aprendizagem de funções. Acrescenta-se aqui que as ***equipes de desenvolvedores*** foram formadas por membros que estavam inseridos nos grupos de pesquisa e outros convidados externos de áreas necessárias para o desenvolvimento dos produtos, como designers, programadores, entre outros.

Assim, observamos que as motivações iniciais de ambos os projetos estavam concentradas na ***problemática*** do ensino e da aprendizagem com tecnologias digitais. Ambos os pesquisadores consideraram as possibilidades de utilização e os auxílios que podem ser oferecidos. Enfim, ponderaram-se todos os subsídios que os recursos tecnológicos podem prover para auxiliar os usuários em suas atividades de ensinar e aprender.

Definidas as problemáticas, foram criadas ***hipóteses*** de utilização dos recursos, porém considerando os referenciais teóricos que são utilizados dentro dos grupos de pesquisa que os software estavam sendo desenvolvidos. Foram vistos os paradigmas educativos vigentes (e escolhidos) pelos autores para dar início a idealização de situações hipotéticas de como o software poderia auxiliar na problemática delimitada. Com a delimitação de situações hipotéticas, surgiram, simultaneamente, os ***referenciais teóricos*** base que fundamentaram essas situações. Pudemos perceber, com a análise realizada, a mudança dos referenciais ao longo do tempo e ao longo da utilização dos programas, bem como quando ocorreu a mudança de grupo de pesquisa, no caso do Casyopée, que ao ser inserido em outro projeto, foi submetido a uma atualização de seus referenciais teóricos para melhor atender as prerrogativas das pesquisas do grupo em questão.

É válido ressaltar, ainda, que esses referenciais não são escolhidos ao acaso. Cada equipe realizou um levantamento teórico quanto as contribuições didáticas, cognitivas, epistemológicas, tecnológicas, e de outras naturezas, para dar aporte aos produtos que estavam sendo idealizados. Dessa forma, definiam-se os ***objetivos*** dos produtos e como esses auxiliariam nas diversas problemáticas levantadas. Questões curriculares, concepções dos estudantes, visão de ensino dos professores, documentos oficiais, novos recursos digitais, entre outros aportes, eram consultados para verificar o modo que os produtos poderiam ser úteis para os seus diversos usuários.

À vista disso, com os levantamentos realizados, a ***especificação*** dos projetos era iniciada, decidia-se qual tipologia do recurso e quais ***situações de utilização*** seriam construídas observando as possíveis reações dos usuários e como esses se comportariam diante das atividades colocadas – surgindo assim o protótipo do software. A única exceção quanto a utilização de protótipos foi o Modellus, visto que o mesmo se tratava de uma evolução de outro projeto. A ***prototipação*** considerava as situações hipotéticas levantadas, bem como as previsões das interações dos usuários e, em ambos os projetos, o protótipo era colocado em utilização tanto com as equipes, quanto com os possíveis usuários, fazendo com que a utilização, também, fosse uma fonte de dados para evoluir e aperfeiçoar os produtos.

De forma concomitante à prototipação, ocorria a ***experimentação*** dos software. Os desenvolvedores testavam as hipóteses, as funcionalidades e verificavam se os objetivos elencados eram atendidos conforme o que fora proposto nas situações de utilização. Ressaltamos a inserção dos projetos em grupos de pesquisa e todo o aparato necessário para produções acadêmicas de qualidade. Ambos os projetos realizaram experimentações com estudantes da Educação Básica e/ou Superior, em algumas situações nos ambientes próprios desses estudantes e, em outras, convidando-os para as faculdades/universidades.

Ainda na experimentação, as equipes se organizaram para obter dados quanto a utilização, o desempenho e o atendimento dos objetivos por parte do software. Era iniciada, assim, a ***análise a posteriori.*** Apesar de receber o nome de "posteriori", ocorre ao passo que a experimentação está sendo realizada. Essa análise, nos projetos investigados, teve por finalidade verificar como as interações com os usuários ocorreram, compreender limites e possibilidades com a utilização dos software; entender como os referenciais teóricos eram considerados nas experimentações e, em geral, confrontar as hipóteses levantadas inicialmente, culminando assim com o processo de ***validação*** dos produtos.

As validações foram momentos em que os pesquisadores apresentavam as contribuições de seus projetos para a comunidade acadêmica. Utilizando-se das experimentações e do confronto das hipóteses com o que aconteceu em uso, os subsídios, apontamentos e as incoerências dos software eram apresentadas.

Concluímos, assim, que as metodologias de desenvolvimento as quais os software Casyopée, Function Probe e Modellus foram submetidos podem ser classificadas como engenharias Didático-Informáticas. Isso, visto que consideraram nas criações desses produtos os cuidados de desenvolver situações de ensino e aprendizagem analisando referenciais teóricos, considerando situações hipotéticas e prevendo as interações com os usuários –

elementos da Engenharia Didática, e, também, foram considerados os métodos de levantamento de requisitos, as potencialidades tecnológicas, características de layout e design – elementos da Engenharia de Software.

Apresentamos esse desfecho na tentativa de padronizar (porém, sem considerar como único meio) o desenvolvimento da tipologia de software micromundo, colocando a Engenharia Didático-Informática como uma metodologia robusta para criação de recursos tecnológicos digitais que auxiliem de fato nas relações de ensinar e de aprender, considerando fundamentações teóricas ricas sobre os conhecimentos e aportes tecnológicos substanciais.

# 7 A UTILIZAÇÃO DA EDI EM PROJETOS DE SOFTWARE

Apresentamos neste capítulo os resultados da investigação realizada nos projetos de software que utilizaram a Engenharia Didático-Informática e considerações sobre as utilizações: Function Studium, Conics Studium 3D e Magnitude Studium.

A análise foi realizada a partir dos resultados das pesquisas expostos nas teses de doutoramento e dissertação de mestrado dos pesquisadores, bem como na produção acadêmica realizada e publicada sobre a utilização, design e avaliação desses recursos.

## 7.1 FUNCTION STUDIUM

Com o objetivo de elicitar requisitos, prototipar e validar um software para abordar o conceito de taxa de variação de funções, Silva (2016), em sua pesquisa de Mestrado, utilizou como aporte teórico-metodológico os princípios da Engenharia Didático-Informática para nortear a construção do software Function Studium.

A problemática central do estudo de Silva (2016) foi a verificação das contribuições de tecnologias digitais para a aprendizagem do conceito de taxa de variação onde se considere a perspectiva covariacional, sendo essa de suma importância para a compreensão dos saberes relacionados ao campo de funções. Além disso, havia a preocupação em integrar características de estudos teóricos sobre o ensino e a aprendizagem da taxa de variação às potencialidades tecnológicas oferecidas nos tempos atuais.

Com isso, os objetivos da investigação de Silva (2016) foram:

> [...] a prototipação e validação de um software para a abordagem da taxa de variação de funções, orientado pelo quadro teórico-metodológico da Engenharia Didático-Informática. O termo prototipação, referente à Engenharia de Software (SOMMERVILLE, 2003) foi tomado aqui no sentido do processo de estabelecimento dos requisitos e modelos prévios à versão final do software, já a validação corresponde a avaliação que busca responder se o software atendeu aos objetivos pré-definidos (2016, p. 16).

Silva (2016) justifica a escolha em utilizar a Engenharia Didático-Informática percebendo a importância dessa metodologia, observando que a EDI era uma metodologia em construção e sua utilização contribuiu para a evolução dessa. Segundo o autor,

> A Engenharia Didático-Informática (EDI) (BELLEMAIN et al, 2015) caracteriza-se como uma metodologia de desenvolvimento de software educativo, que busca aliar os aspectos do ensino e da aprendizagem aos aspectos tecnológicos computacionais, na concepção e desenvolvimento de software educacionais de Matemática (SILVA, 2016, p. 18).

Com isso, Silva et al (2017) divulgam os resultados da utilização do modelo de processo proposto na EDI. Apresentamos na Tabela 14 um resumo com a utilização de algumas etapas da EDI para a criação do software Function Studium.

Tabela 14 – Resumo do Function Studium

| **DELIMITAÇÃO DO CAMPO** |
|---|
| Teve por objetivo selecionar os saberes que o software iria abordar, ou seja, procurou-se responder os seguintes questionamentos: quais conhecimentos matemáticos serão abordados com o software? Quais são os objetos relacionados que também devem ser trabalhados? Quais profissionais podem auxiliar nesse desenvolvimento? |
| **FASE TEÓRICA** |
| Realizou-se uma pesquisa inicial direcionada a conhecer os encaminhamentos didáticos, epistemológicos, cognitivos e tecnológicos do conhecimento delimitado. Foi feito um apanhado teórico sobre o campo para dar início ao processo de levantamento de requisitos.<br><br>Ao ser concluído o levantamento teórico, os primeiros requisitos puderam ser descritos. Ainda na fase teórica, iniciou-se a prototipação, na qual, durante o desenvolvimento do protótipo, foram delimitadas as situações de uso, os problemas que poderiam surgir com a utilização do software e as hipóteses de respostas dos usuários, por meio da análise a priori. |
| **FASE EXPERIMENTAL** |
| Foi composta de momentos específicos para os testes e da análise do software: interface, comandos, botões, etc, juntamente com a verificação do atendimento aos objetivos de ensino e aprendizagem. A experimentação do software foi realizada com estudantes da Licenciatura em Matemática da UFPE, que utilizaram o *Function Studium* para resolver atividades sobre a taxa de variação de funções. |
| **ANÁLISE A POSTERIORI E VALIDAÇÃO** |
| A análise a posteriori foi realizada considerando os dados da experimentação. Foram analisadas a gravação da tela no momento do uso do software, a gravação em vídeo da interação dos usuários-sujeitos, as folhas do teste escrito aplicado e as observações do pesquisador.<br>A validação se deu por meio do estudo dos benefícios e restrições do uso do software e a relação com as etapas do modelo de processo. Foi verificado se os referenciais teóricos escolhidos, bem como as funcionalidades tecnológicas, auxiliaram na aprendizagem dos saberes delimitados. Também foram estudadas as contribuições da EDI na criação do software e quais foram as limitações dessa metodologia. |

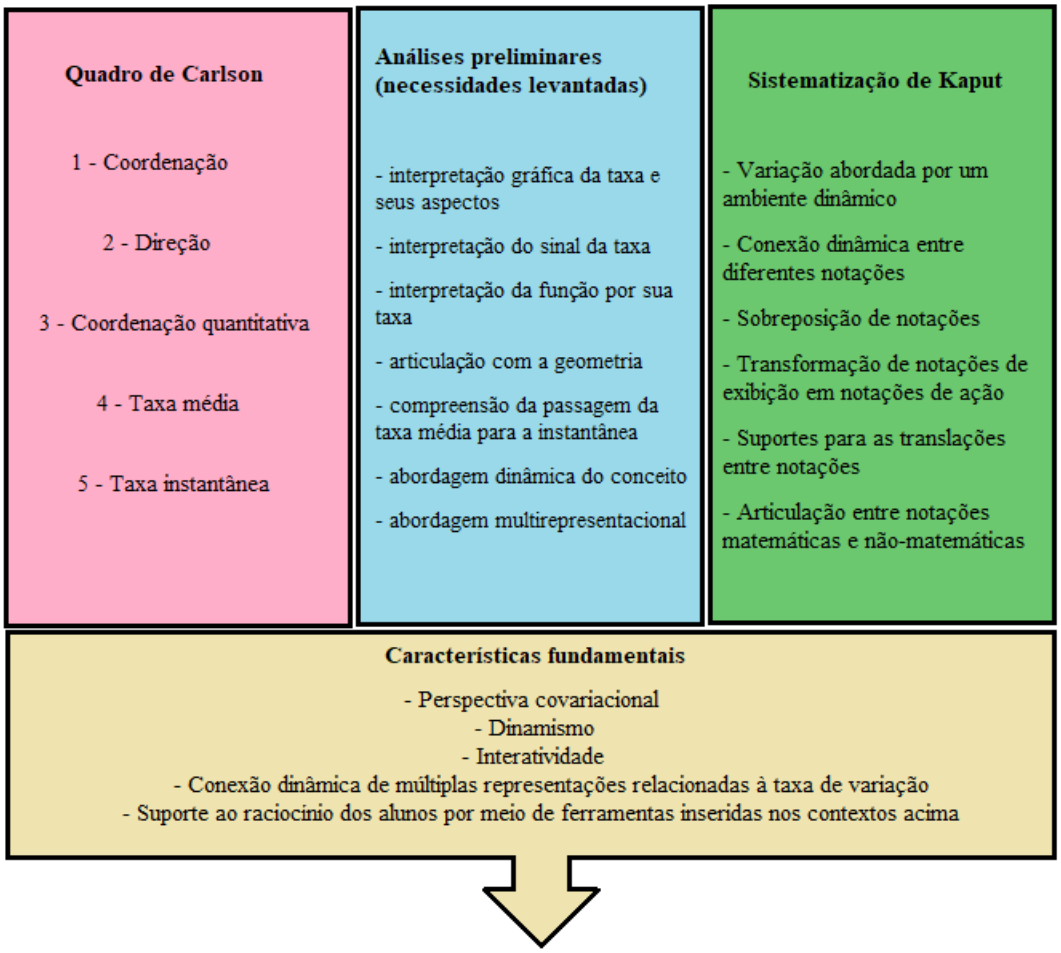
Adaptado de Silva et al (2017).

Observamos, com a análise do desenvolvimento do Function Studium, algumas características importantes da utilização da EDI no estudo de Silva (2016). Um aspecto observado fora na experimentação do programa. Percebemos que o emprego da EDI apresentou resultados significativos:

> De forma geral, os benefícios do uso do software Function Studium apontados na análise dos dados do experimento referem-se a recursos, funcionalidades e notações elencadas na análise de requisitos, com base nas análises preliminares. É preciso destacar, dentro do Modelo de Processo, a sequência definida (análise preliminar - análise de requisitos - desenvolvimento) como importante para a conexão entre os objetivos pretendidos pelo software e os recursos nele implementados.
> Um dos benefícios do Function Studium mais destacados no experimento foi relacionado às contribuições da conexão de notações de forma simultânea (B1, B5 a B13), um requisito elicitado para o software com base nas análises preliminares. (SILVA, 2016, p. 148).

De acordo com Silva (2016), houve benefícios do uso de recursos idealizados e implementados a partir dos resultados da análise preliminar nas dimensões epistemológica e cognitiva, o que reforça a importância da utilização da EDI. As análises teóricas realizadas pelo autor fundamentaram os princípios para a construção do Function Studium. A Figura 7 apresenta os referenciais utilizados quanto ao que deve ser levado em consideração no ensino e na aprendizagem da área de estudo do software.

Figura 7 – Articulação para estabelecer os princípios fundamentais do protótipo



Fonte: Silva (2016).

Em relação a fase de prototipação, o autor destaca a interação de pesquisadores com desenvolvedores, também solicitada pela EDI. A percepção da relevância da formação de uma equipe transdisciplinar ainda é um ideal a ser alcançado, mas que fora percebida a importância no desenvolvimento de produtos de software. Apresentamos, na Figura 8, uma das interações entre o pesquisador e o engenheiro programador.

Figura 8 – Interação para implementar melhorias

**Pesquisador do estudo - usuário:** Ao traçar um gráfico, o protótipo permite o deslize de um ponto do gráfico diretamente na curva. É interessante que essa variação também seja permitida ao selecionar-se o ponto do eixo x correspondente ao ponto da curva, além de conectar a variação em x à variação em y.

**Engenheiro - Programador:** OK, veja se o que fiz resolve.

**Pesquisador do estudo - usuário:** Ok, agora só falta a figura do ponto para indicar melhor onde posicionar o cursor no eixo.

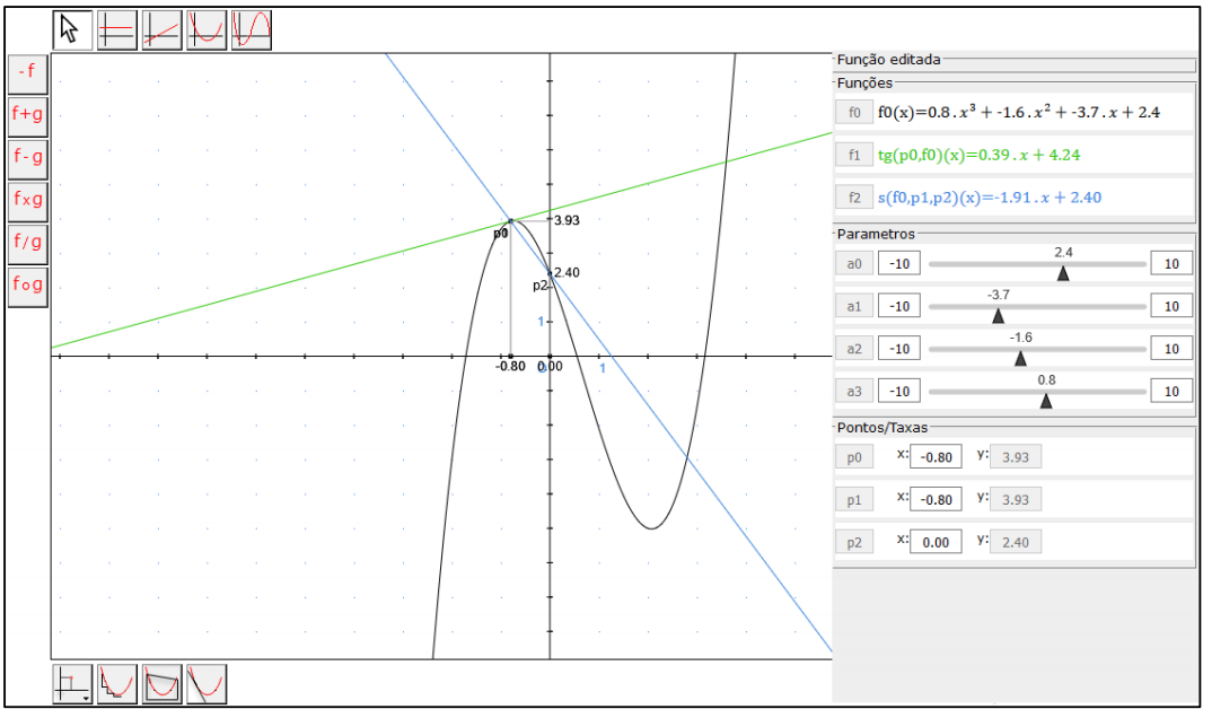
**Engenheiro-Programador:** OK, feito.

Fonte: Silva (2016).

A interação, exibida na Figura 8, é um dos diversos diálogos em que o pesquisador comunica que há a necessidade de alteração de uma característica do protótipo em fase de construção. A possibilidade de uma comunicação rápida entre programador e desenvolvedor/pesquisador é uma das características que preconizamos na EDI advinda dos métodos ágeis.

Após o lançamento do Function Studium, seu primeiro layout é apresentado na Figura 9, e, a partir das interações e discussões oriundas dos estudos de Silva (2016) e Silva et al (2017), verificamos algumas exigências de modificações no modelo de processo de software da EDI. Com isso, organizamos, na Tabela 15, a seguir, algumas sugestões de implementações.

Figura 9 – Primeiro layout do Function Studium




Fonte: Silva (2016).

Tabela 15 – Algumas implementações

| ETAPA DA EDI | SUGESTÃO |
|---|---|
| Fases teórica e experimental | Explicitar melhor cada fase da EDI; identificar os elementos de cada fase. |
| Formação da equipe de desenvolvedores | Definir o que é uma equipe pluridisciplinar. |
| Processo de desenvolvimento | Deixar mais clara a operacionalização do processo; especificar como “construir o software”. |

Fonte: o autor.

É importante mencionar que tais sugestões surgiram no momento em que o modelo de processo de software educativo fundamentado na Engenharia Didático-Informática estava em estudo e em processo de criação, porém, todas essas sugestões foram consideradas nesta investigação com o objetivo de aperfeiçoar e fazer evoluir a EDI.

## 7.2 CONICS STUDIUM 3D

A motivação da elaboração do software Conics Studium 3D se deu a partir do foco na investigação dos conhecimentos das curvas cônicas, considerando as dificuldades do ensino, observando-se a forma fragmentada e o foco nas representações algébricas dessa área. Siqueira (2019) observou a importância de integrar as abordagens sintética e analítica articulando várias representações desses objetos matemáticos. De acordo com o autor,

> Foi apontada a necessidade de privilegiar uma proposta integrativa das abordagens sintética e analítica, em oposição à fragmentação, com articulação entre as várias representações, relacionando as variáveis visuais às unidades simbólicas correspondentes, por meio das transformações por conversão e tratamento, fazendo uso de um recurso computacional (SIQUEIRA, 2019).

Em específico, Siqueira (2019) investigou, referente às curvas cônicas, quais são "os princípios teórico-metodológicos que podem servir à concepção-desenvolvimento-validação de um recurso computacional capaz de favorecer a manipulação e articulação dinâmica de sistemas de representação" (p. 33).

Os objetivos do estudo caracterizam uma pesquisa com a perspectiva de concepção e construção de software educativo, destacando no objetivo geral as fases de construção de software: "conceber, desenvolver e validar um artefato que possibilite a integração de recursos computacionais na exploração dos tratamentos e das conversões dos registros de representação semiótica das curvas Cônicas" (SIQUEIRA, 2019, p. 33).

O autor justifica a utilização da Engenharia Didático-Informática observando as limitações da Engenharia de Software para a construção de software educativos, segundo ele,

> Ao estudar a concepção e desenvolvimento de software educativos Tibúrcio (2016) identificou a necessidade de um arcabouço teórico-metodológico que venha favorecer a concepção de software que contemplassem aspectos do ensino e da aprendizagem da Matemática e do desenvolvimento de tecnologias computacionais, quais sejam, didáticos, cognitivos, epistemológicos, informáticos, dentre outros. Além disso, observou as limitações da engenharia de software, que não contempla as especificidades que os software educativos necessitam e da engenharia didática, que embora não tenha sido criada para auxiliar no desenvolvimento de software educativos, tem, em suas fases de criação e o desenvolvimento de sequências didáticas, características comuns com as dos software existentes. Aliado a isso, analisou as características aplicáveis para a construção de software educativos a partir

> da engenharia de requisitos. Após estas constatações, Tibúrcio (2016) integrou os conhecimentos de tais engenharias, a de software, com a padronização do desenvolvimento de software e métodos de obtenção de requisitos; e a didática, com os elementos de investigação teórica e experimental sobre o ensino e a aprendizagem, criando a Engenharia Didático-Informática (SIQUEIRA, 2019, p. 35).

Com isso, a utilização da EDI por Siqueira (2019) foi realizada considerando todas as fases propostas no modelo de processo, o que nos proporcionou reflexões sobre pontos a serem melhorados na metodologia aqui discutida. Quanto às dimensões apresentadas na EDI, o autor resume, em uma tabela, o que cada uma significa,

Tabela 16 – Dimensões da EDI por Siqueira (2019)

| | |
|---|---|
| *Dimensão epistemológica* | analisa as características específicas do conhecimento e sua relação com aquilo que pode favorecer ou dificultar a aprendizagem. |
| *Dimensão cognitiva* | aborda a(s) teoria(s) cognitivas que estão envolvidas com a aprendizagem de um domínio específico, considerando como o estudante aprende o conhecimento, por exemplo, como mobiliza as múltiplas representações, como articula as representações, ou como realiza a manipulação e faz a conexão entre elas. |
| *Dimensão didática* | cuida de investigar o estado atual do ensino do domínio, bem como as principais contribuições e dificuldades geradas e as consequências desse ensino. |
| *Dimensão informática* | Trata de como os recursos tecnológicos computacionais podem contribuir para o ensino e a aprendizagem do domínio, além das características fundamentais que o ambiente deve conter para atender às especificidades emanadas dos estudos das outras dimensões. |

Fonte: Siqueira (2019, p. 36).

É interessante destacar o aprofundamento teórico realizado por Siqueira (2019) baseado nas dimensões da EDI. O autor constituiu um apanhado específico das curvas cônicas que foi de grande valia para a construção do protótipo do software. Apresentamos, na Tabela 17, os indicativos desse levantamento para a concepção do software Conics Studium 3D.

Tabela 17 – Dimensões da EDI no estudo de Siqueira (2019)

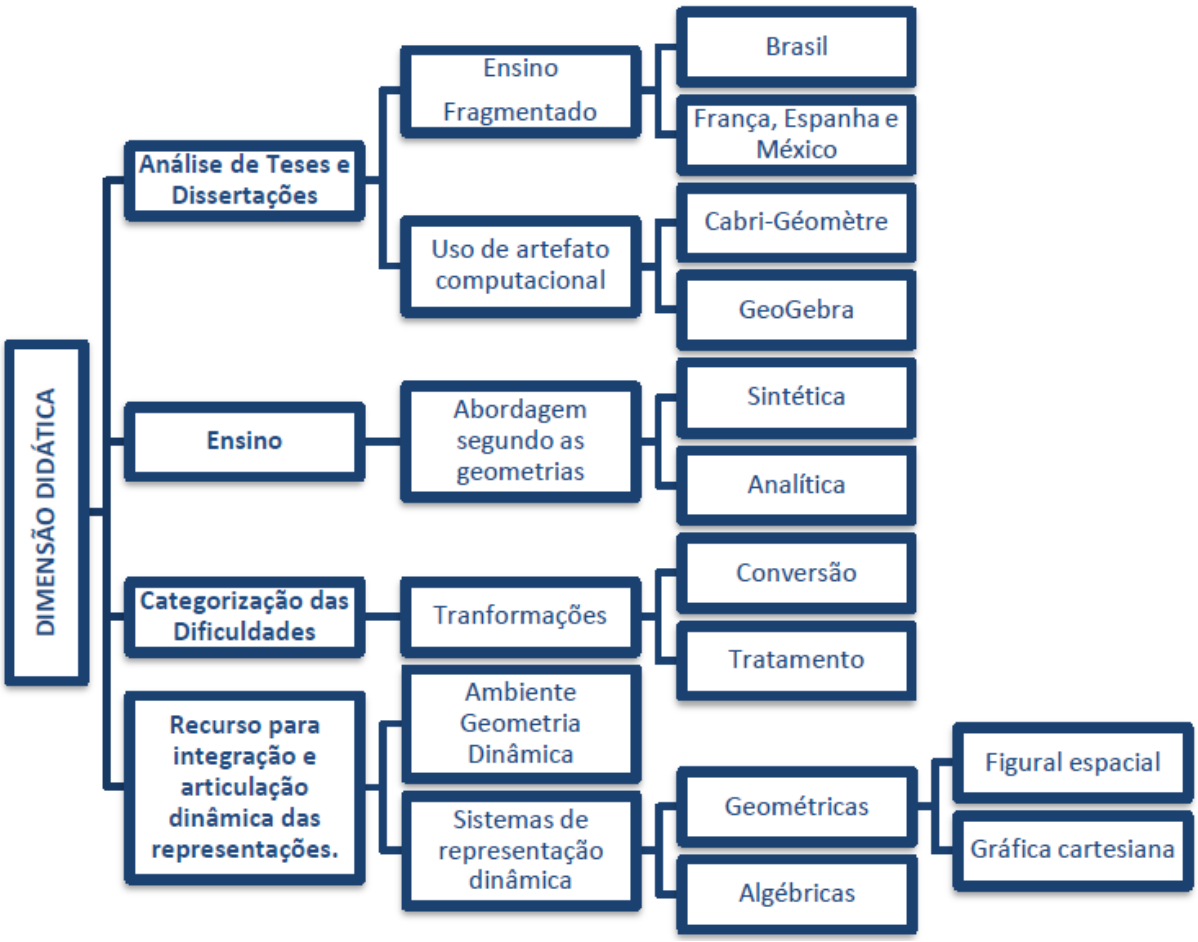
| **DIMENSÃO EPISTEMOLÓGICA** |
| --- |
| O panorama histórico e, principalmente, epistemológico do nascimento e desenvolvimento da Teoria das Cônicas como um objeto matemático, possibilitou identificar vários elementos que parecem importantes destacar quando buscamos construir um encadeamento que justifique a articulação dos seus registros de representação, levando em consideração a geometria numa perspectiva sintética, analítica e projetiva. |
| **DIMENSÃO COGNITIVA** |
| Destacamos que a dimensão cognitiva tem o propósito de tentar compreender a natureza e onde se encontram as dificuldades, que inúmeros alunos e muitas vezes professores, têm na compreensão de conceitos matemáticos. Nesse sentido, a Teoria dos Registros de Representações Semióticas – TRRS de Raymond Duval, esforça-se em entender o funcionamento cognitivo dos estudantes em situações de ensino, possibilitando, desse modo, compreender as dificuldades e os problemas da aprendizagem. |
| **DIMENSÃO DIDÁTICA** |
| Estudos têm mostrado que, apesar das cônicas se constituírem um tema matemático relevante, sendo objeto de investigação ao longo de vários séculos, seu espaço vem sendo reduzido na Educação Básica, embora continue presente em vários cursos superiores. Apontam, ainda, como causas dessa grande dificuldade em ensinar as cônicas: o contexto geométrico, o próprio objeto e a formação de professores. O seu ensino ocorre de maneira fragmentada e superficial, sendo priorizada sua abordagem analítica com enfoque algébrico, o que tem levado pesquisas a sugerirem a articulação dinâmica dos seus registros de representação semiótica, aliada à utilização de recursos computacionais, como uma maneira de superar tais problemas. |
| **DIMENSÃO INFORMÁTICA** |
| Destacamos que a criação, por meio do computador, de verdadeiros registros de representação semiótica - no sentido de Duval (1995) e destacada por Balacheff (1999) - tem por objetivo facilitar, através da manipulação dessas representações, a compreensão dos conceitos representados. Parece ser possível conceber e desenvolver, através dos princípios da Engenharia de Software Educativos com tecnologia computacional, ambientes que venham contemplar as especificidades dos conteúdos associadas à possibilidade de representação e exploração dos objetos matemáticos, como por exemplo, na criação de novos registros “dinâmicos” de representação, onde as articulações entre variáveis visuais, visuais figurais e unidades simbólicas correspondentes, podem aparecer nas variações conjuntas e contínuas, tanto dos elementos algébricos como geométricos. |

Fonte: Adaptado de Siqueira (2019).

O autor percebe que há uma convergência das dimensões Didática, Epistemológica e Cognitiva a fim de alimentar a Dimensão Informática, no sentido que fora observado em Tiburcio (2016), quando se coloca que essa última dimensão é, também, alimentada pelo levantamento teórico realizado nas dimensões anteriores.

Para explicitar o levantamento realizado por Siqueira (2019), apresentamos aqui, como exemplo, o esquema elaborado pelo autor (Figura 10) com a sistematização do que foi pesquisado na Dimensão Didática.

Figura 10 – Sistematização da Dimensão Didática




Fonte: Siqueira (2019).

Em resumo, visando responder perguntas simples: “O que ensinar?”; “Como ensinar?” e “E como superar as dificuldades?”; O autor realiza um estudo teórico e consegue chegar às conclusões apresentadas na Figura 10. Além da dimensão Didática, detalhada com a figura referida, Siqueira (2019) expõe um diagrama com o que deve ser considerado em cada uma das dimensões, conforme pode ser observado na Figura 11.

Figura 11 – Síntese das Dimensões

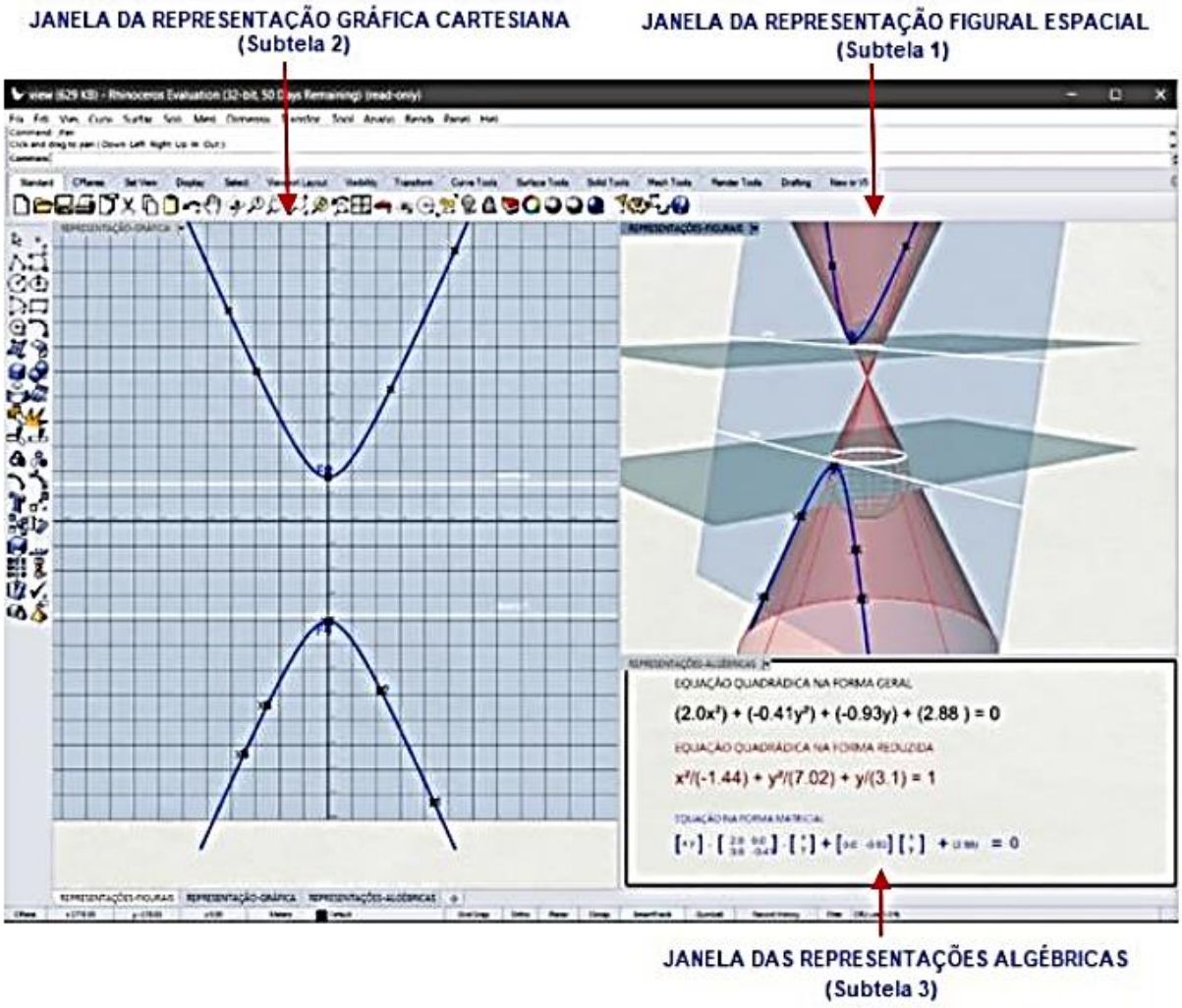

Fonte: Siqueira (2019).

Após a análise teórica concluída, Siqueira (2019) segue as etapas da EDI descrevendo os requisitos para o software que fora desenvolvido. Uma particularidade que é pertinente ressaltar é que o autor considerou a formação de uma equipe pluridisciplinar para a construção do software e teve a colaboração de profissionais de diversas áreas para auxiliar no processo de criação do Conics Studium 3D.

Os resultados da pesquisa de Siqueira (2019) trazem à tona as potencialidades da utilização da EDI, na Figura 12 exibimos o primeiro layout do software, de acordo com o autor,

> O modelo da Engenharia Didático-Informática estruturou nossa pesquisa, contribuindo com a compreensão das características específicas do objeto cônicas (dimensão epistemológica), da maneira como os estudantes mobilizam suas representações (dimensão cognitiva) e das dificuldades geradas no seu ensino (dimensão didática) e, por fim, obtivemos um conjunto de requisitos que colaboraram com a concepção, desenvolvimento e validação do recurso computacional (dimensão informática). Ressaltamos a importância das transformações das questões teóricas, emanadas das dimensões epistemológica, cognitiva, didática e informática, em requisitos para o desenvolvimento do protótipo (SIQUEIRA, 2019, p. 294).

Figura 12 – Layout inicial do Conics Studium 3D




Fonte: Siqueira (2019).

Ressaltamos, a respeito do uso da EDI, uma sugestão quanto a implementar no modelo do processo a Teoria dos Registros de Representação Semióticas proposta por Raymond Duval. Segundo Siqueira (2019), essa teoria poderia ser integrada ao processo de desenvolvimento de software a fim de considerar as múltiplas representações dos objetos matemáticos (de acordo com o referencial em questão). Contudo, a escolha da fundamentação teórica, seja ela cognitiva, didática, epistemológica ou de qualquer outra natureza, surge das análises realizadas de acordo com o campo de saberes a ser investigado para a construção do software.

Acreditamos que a escolha dos referenciais teóricos, em qualquer uma das dimensões, deve ser efetuada após as análises dos pesquisadores e não postas antes da compreensão do problema que se pretende resolver com a utilização das tecnologias digitais a serem desenvolvidas.

## 7.3 MAGNITUDE STUDIUM

No contexto de desenvolvimento de software educativo para o ensino e a aprendizagem de conhecimentos matemáticos, Silva (2019) projetou o Magnitude Studium para dar suporte às atividades docentes no ensino de área e perímetro. O autor utilizou a EDI observando as potencialidades da integração entre a Engenharia Didática e a Engenharia de Software propostas nessa metodologia.

A investigação de Silva (2019) se insere na problemática de concepção, construção e utilização de recursos tecnológicos com fins educativos. A questão de pesquisa central do estudo reflete a preocupação com a criação do software bem como sua utilização para fins didáticos, a saber

> Diante desse contexto, pretendemos responder à seguinte questão de pesquisa: o estudo da concepção, desenvolvimento e validação por meio da operacionalização da Engenharia de Software Educativos (ESE)-EDI permite conceber um software educativo de qualidade para o ensino e aprendizagem da matemática? Partimos da hipótese de que os recursos e funcionalidades implementados no micromundo poderão dar suportes ao ensino de área e perímetro de figuras planas (SILVA, 2019, p. 25).

Os conhecimentos que foram delimitados como foco do estudo de Silva (2019) estão inseridos na Geometria e nas Grandezas e Medidas. De acordo com o autor, após realizar uma análise de software disponíveis (Apprenti Géomètre 2, Cabri Géomètre II, Cinderella, Déclic, Euklides, Geogebra, Igeom, Régua e Compasso, Tracenpoche) fora observada a necessidade de construir um software que fosse voltado para as grandezas geométricas, em especial para a grandeza área: “afinal, embora nos software supracitados possam ser construídas diferentes tarefas de área e perímetro, eles não foram desenvolvidos especificamente para trabalhar esses conceitos” (p. 24).

Com isso, os objetivos da pesquisa de Silva (2019) dizem respeito à concepção de um software que considere as potencialidades tecnológicas e, também, o que as teorias de ensino e aprendizagem indicam sobre os saberes do âmbito.

> Traçamos como objetivo geral: Conceber, desenvolver e validar um micromundo como elemento de suporte ao professor para o ensino de área e perímetro, utilizando-

> se do modelo de desenvolvimento de Software Educativo proposto na EDI. Para alcançarmos esse objetivo, elencamos três específicos:
> Realizar um levantamento dos aspectos epistemológicos, didáticos, cognitivos e informáticos a respeito do conceito de área e perímetro;
> Verificar os requisitos necessários ao software, considerando a dimensão computacional de aspectos epistemológicos, didáticos e pedagógicos para engenharia;
> Prototipar, desenvolver e validar a versão inicial do micromundo utilizando-se do modelo de desenvolvimento de SE estabelecido na EDI (SILVA, 2019).

A utilização da EDI, por Silva (2019), seguiu as fases delimitadas no modelo de processo de software da metodologia. Na *delimitação do campo* fora definido o conjunto de conhecimentos relacionados a área e perímetro, por conseguinte esses foram situados nas dimensões estruturadas: cognitiva, epistemológica, didática e informática que fizeram surgir os requisitos para a fase de *análise de requisitos*. Além disso, o levantamento nas dimensões da EDI contribuiu para perceber a organização estrutural do modelo de processo, observado também, nos outros projetos que utilizaram essa metodologia, conforme pode ser observado na Tabela 18.

Tabela 18 – Dimensões da EDI no estudo de Silva (2019)

| **DIMENSÃO EPISTEMOLÓGICA** |
|---|
| Por meio dessas teorias, podemos observar que a noção de área pode ser definida usando objetos geométricos como polígonos, na teoria presente nos Elementos de Euclides e em Hilbert, ou quadrados, na teoria elaborada por Lebesgue. (ANWANDTER-CUELLAR, 2012). Essas abordagens teóricas se referem a duas maneiras de conceber a noção de área. As primeiras definem áreas sem o uso das medidas e a segunda se baseia em uma "unidade de área" no processo de medição. |
| **DIMENSÃO COGNITIVA** |
| Trazemos à tona, ainda, a Teoria dos Campos Conceituais de Vergnaud (1996), que nos permite entender a construção de um conceito, seguido de um conjunto de situações que dão sentido ao conceito de área. |
| **DIMENSÃO DIDÁTICA** |
| Apoiamos esse estudo nas pesquisas realizadas por Régine Douady e Marie Jeanne Perrin-Glorian (1989). Entendemos que as malhas quadriculadas e triangulares, o tangram e os poliminós produzidos e utilizados em papel, borracha, plástico, madeira (ambientes não digitais) se mostraram pertinentes para o ensino de área e perímetro de figuras planas. Pretendemos implementar tais recursos em um micromundo e verificar suas possíveis potencialidades para o estudo desses conceitos. |
| **DIMENSÃO INFORMÁTICA** |
| Realizamos análises em vários micromundos, tutoriais e simulações, que tratam do conceito de área e/ou perímetro. Para isso, elencamos três pontos centrais de acordo com o foco da nossa pesquisa: recursos para o trabalho com área (ferramentas e menus específicos); ferramentas de planejamento (configuração de menus, macro construções e histórico das construções) e aspectos técnicos. Nessa fase, também discutimos de forma mais enfática, questões cognitivas e didáticas acerca do conceito de perímetro, com base nos estudos de Barbosa (2002), Brito (2003) e Brito e Bellemain (2004). |

Fonte: Adaptado de Silva (2019).

Em síntese, os requisitos levantados na análise teórica foram destacados em duas partes: a primeira referente às dimensões epistemológica, cognitiva e didática, conforme a Tabela 19, e a segunda, referente aos requisitos da dimensão informática, conforme a Tabela 20.

Tabela 19 – Requisitos finais Magnitude Studium – Parte 1

| **Dimensão Epistemológica** | **Ferramentas e menus específicos para trabalhar as abordagens de área** |
|---|---|
| Abordagem de área nos Elementos de Euclides (2009) e em David Hilbert (1994).<br><br>Abordagem de área em Lebesgue (1975). | ✓ Corte e colagem<br>✓ Mover;<br>✓ Rotação;<br>✓ Reflexão;<br>✓ Malha (quadrada, isométrica e hexagonal, pontilhadas ou não).<br>✓ Opção de magnetismo nas malhas;<br>✓ Ladrilhos predefinidos (Tangram e Poliminós);<br>✓ Ferramentas de: medir área, perímetro e comprimento;<br>✓ Construção de objetos geométricos (ponto, ponto sobre objeto, ponto de intersecção, ponto médio, reta, reta paralela, reta perpendicular, segmento, semirreta, circunferência e arco, mediatriz e bissetriz, compasso);<br>✓ Opção de deixar o polígono rígido;<br>✓ Polígono regular; |
| **Dimensão Cognitiva e Didática** | |
| Abordagem de área como uma grandeza. Douady e Perrin-Glorian (1989).<br><br>Teoria dos Campos Conceituais (VERGNAUD, 1996): construção de situações que dão sentido ao conceito de área. | |

Fonte: Silva (2019, p. 185).

Tabela 20 – Requisitos finais Magnitude Studium - Parte 2

| **Dimensão Informática** | |
|---|---|
| **Ferramentas de suporte ao planejamento do professor (comuns aos softwares de GD explicitados na dimensão informática)** | • Configuração de menus;<br>• Macro Construção;<br>• Histórico das construções;<br>• Opção para a escolha das unidades de medidas ( múltiplos e submúltiplos do metro para o perímetro e o comprimento, e os múltiplos e submúltiplos do $m^2$ para as áreas). |
| **Ferramentas comuns aos softwares de geometria** | ✓ Duplicar;<br>✓ Colorir com níveis de gradação à figura;<br>✓ Copiar e Colar;<br>✓ Esconder objeto;<br>✓ Apagar objetos;<br>✓ Desfazer (ctrl z);<br>✓ Nomear objetos. |
| **Aspectos de acessibilidade** | • Compatibilidade com plataforma web;<br>• Interface atrativa;<br>• Facilidade no manuseio;<br>• Ferramentas autoexplicativas;<br>• Disponibilidade de tutorial e menu ajuda;<br>• Navegabilidade pelo teclado. |

Fonte: Silva (2019, p. 186).

Na fase de *análise à priori e prototipação*, Silva (2019) construiu os layouts iniciais com os recursos que deveriam ser implementados no Magnitude Studium. Assim, a primeira

versão foi apresentada e seu funcionamento foi analisado. É importante destacar a descrição detalhada dos elementos necessários, bem como as situações de utilização que o autor explicitou em sua pesquisa: funcionalidades, menus, ferramentas, enfim, todas as possibilidades de utilização do Magnitude Studium foram elencadas.

Em sequência, a *experimentação* se deu com docentes especialistas na área de conhecimentos que o software aborda e, também, com profissionais com experiências e habilidades na área de informática educativa. A Tabela 21 apresenta os participantes do experimento do Magnitude Studium e algumas informações sobre eles.

Tabela 21 – Perfil dos sujeitos da pesquisa

| | **INFORMAÇÕES** | | | |
|---|---|---|---|---|
| **SUJEITOS DA PESQUISA** | **Formação Inicial** | **Maior Titulação** | **Etapa da Educação que Atua** | **Tempo de Atuação na Área de Educação** |
| **A** | Licenciatura em Expressão Gráfica | Mestrando em Educação Matemática e Tecnológica. | Educação Básica 6º ao 9º e no Ensino Médio | 2 anos |
| **B** | Licenciatura em Matemática | Doutor em Educação Matemática e Tecnológica | Educação Básica (6º ao 9º e no Ensino Médio) | 35 anos |
| **C** | Licenciatura em Expressão Gráfica | Mestre em Educação Matemática e Tecnológica (Doutoranda em Educação Matemática e tecnológica) | Formação de Professores da Educação Básica e 7º ano do Ensino Fundamental. | 1 ano |
| **D** | Licenciatura em Matemática | Mestre em Educação Matemática e Tecnológica (Doutorando em Educação Matemática e tecnológica) | Educação Básica (6º ao 9º e no Ensino Médio) | 5 anos |
| **E** | Licenciatura em Matemática | Doutor em Educação Matemática e Tecnológica | Educação Básica (Ensino Médio) | 15 anos |
| **F** | Bacharel em Design | Mestre em Design | Ensino Superior | 5 anos |

Fonte: Silva (2019).

Destacamos, aqui, o cuidado na escolha dos profissionais para a análise do protótipo. Essa situação que deve ser considerada em outros projetos de software utilizando a EDI: o experimento contou com a participação de integrantes de grupos de pesquisa vinculados ao programa de pós-graduação que o pesquisador estava inserido. A justificativa dessa escolha se deu pelo fato de os sujeitos terem potencial para contribuir no processo de experimentação e

validação com olhares direcionados aos aspectos teóricos conceituais referentes às grandezas área e perímetro e tecnológicos, no que diz respeito à interface, funcionalidades e interatividade das diversas ferramentas do software em desenvolvimento (SILVA, 2019).

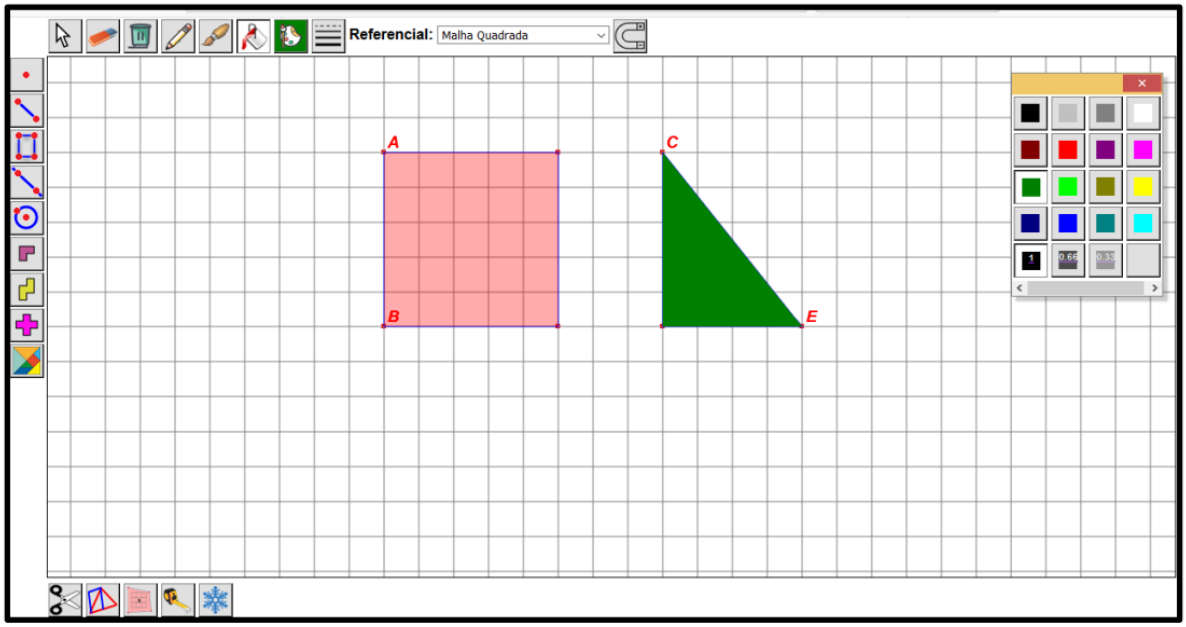
Após essa fase, o pesquisador iniciou a *análise à posteriori e validação* verificando que o protótipo do software atendeu parcialmente ao objetivo especificado na sua concepção, levando o autor a considerar os retornos dados na fase da experimentação a fim de aprimorar o software para que esse contemplasse os objetivos que foram definidos em sua concepção. O autor resume essa fase observando os seguintes aspectos:

> Mesmo apresentando algumas limitações, pudemos perceber a potencialidade do MS para o desenvolvimento das diferentes tarefas construídas pelas duplas. De forma geral, temos que o uso do tangram e dos poliminós mostraram-se pertinentes para a elaboração das tarefas de área e perímetro, como também a possibilidade de arrastar as figuras sobre a tela; a aplicação de rotação e reflexão nos poliminós; a construção de polígonos; o uso da ferramenta *measurement*[28] e a malha quadriculada, mesmo com as limitações apresentadas durante todo o experimento. A sugestão de destaque diz respeito à solicitação da dupla 1 de ampliar o trabalho com as figuras não poligonais e a composição de uma figura poligonal com outra não poligonal (SILVA, 2019, p. 306).

Como resultado da criação do Magnitude Studium, o primeiro layout é apresentado na Figura 13 com duas construções geométricas elaboradas com o software. O autor indica que, mesmo havendo a indispensabilidade de aperfeiçoamento de algumas características do software, foi possível perceber o potencial desse quando comparado a outros programas de Geometria Dinâmica. "[...] haja vista que nenhum dos software de geometria dinâmica desenvolvidos em diferentes épocas e apresentados em nossas análises preliminares propõe-se a esse objetivo, ainda que, por meio deles, possamos construir diferentes tarefas [...]" (SILVA, 2019, p. 318).

[28] A Ferramenta *Measurement* é uma funcionalidade do software Magnitude Studium que possibilita determinar a área de polígonos.

Figura 13 – Layout inicial do Magnitude Studium




Fonte: Silva (2019).

Uma das dificuldades elencadas por Silva (2019) se refere ao processo de validação de software proposto na EDI. Segundo o autor, não está claro como realizar esse processo conforme o que é explicitado no modelo de processo. Essa sugestão fora levada em consideração e foi implementada no novo modelo que foi aperfeiçoado nesse estudo.

## 7.4 A PLURALIDADE DE PARTICULARIDADES NA UTILIZAÇÃO DA ENGENHARIA DIDÁTICO-INFORMÁTICA

Ao serem observadas as utilizações da Engenharia Didático-Informática, percebemos questões em comum ao mesmo tempo que diversas particularidades de cada desenvolvimento foram apreciadas. Nesta sessão discutimos elementos da EDI apresentando pontos de grande relevância em cada um dos estudos que utilizou essa metodologia para desenvolver software educativo.

### 7.4.1 O ensino da Matemática com o uso de tecnologias digitais

Nos estudos aqui discutidos, percebemos que o levantamento teórico realizado na dimensão didática apresentou que existem problemas referentes ao ensino dos conhecimentos abordados pelos pesquisadores. Mesmo se tratando de áreas distintas da Matemática, os problemas expostos revelam que o ensino atual é deficitário no que tange à aprendizagem dos

estudantes.

Essa conjuntura nos fez observar a importância que os conceitos de ***situação, tarefa e atividade*** ocupam em cada uma das pesquisas discutidas. Enquanto no bingo dos racionais as situações de utilização (situações didáticas) já estavam definidas pela Engenharia Didática realizada, nos demais software (Function, Conics e Magnitude Studium) foi necessário realizar todo um levantamento teórico para chegar nas situações de utilização dos produtos. Contudo, é importante esclarecer que cada pesquisador possui suas hipóteses de como os software serão utilizados e como eles podem auxiliar no ensino e na aprendizagem dos saberes matemáticos, visto que se deve considerar as pesquisas por eles já realizadas, bem como suas motivações particulares.

Na Tabela 22 apresentamos como essas pesquisas abordam o conceito de situação com a finalidade de perceber as semelhanças e diferenças conceituais, mas com o objetivo comum de utilizar as tecnologias para o progresso da aprendizagem matemática:

Tabela 22 – Conceito de situação nos desenvolvimentos

| SOFTWARE | CONCEITOS-CHAVE |
|---|---|
| **Magnitude Studium** | "Consideramos primeiramente um campo conceitual como um conjunto de problemas e situações cujo tratamento requer conceitos, procedimentos e representações de tipos diferentes, mas intimamente relacionados **(VERGNAUD, 1983)**". Silva, 2019, p. 25. |
| **Conics Studium 3D** | "um dos aspectos mais relevantes à originalidade da atividade matemática diz respeito à mobilização simultânea, de no mínimo, dois registros de representação, ou da possibilidade de trocar a todo o momento de registro de representação **(DUVAL, 2005)**". Siqueira, 2019, p. 131. |
| **Function Studium** | "definem o raciocínio covariacional como: "... atividades cognitivas envolvidas na coordenação de duas quantidades variáveis enquanto se observam as formas como elas mudam uma em relação à outra" **(CARLSON et al, 2002)**". Silva, 2016, p. 37. |

Fonte: o autor

No Magnitude Studium, as situações de uso do software foram construídas fundamentadas na *Teoria dos Campos Conceituais* – TCC de Gérard Vergnaud. Silva (2019) considerou, em sua revisão de literatura, estudos em que a TCC era utilizada para analisar e estruturar situações que dão sentido a área como grandeza. De acordo com o autor,

> Baltar (1996) realizou um breve levantamento dos tipos de problemas sobre área, ao longo da história da matemática, nos resultados das pesquisas sobre a aprendizagem desse conceito e nas avaliações do desempenho de alunos do ensino fundamental no contexto francês. Esse levantamento permitiu à autora evidenciar uma variedade de situações concernente a esse conceito (SILVA, 2019, p. 56).

Fica evidente o conceito de situações no sentido adotado pelo autor com a finalidade de considerar atividades que auxiliam a aprendizagem de competências e habilidades pelos estudantes.

Também observamos que no Function Studium, mas utilizando-se de outro referencial teórico, existiu a preocupação em delimitar as situações que propiciam a compreensão dos conhecimentos abordados pelo software. O *Raciocínio Covariacional*, investigado por Marilyn Carlson, é abordado por Silva (2016) como de grande importância para que o estudante compreenda a variação de duas variáveis quando essas possuem relação. Em específico para a realidade educacional brasileira, o autor apresenta que os documentos oficiais do país orientam os professores a utilizarem uma abordagem covariacional para o ensino de funções.

O Conics Studium 3D se fundamenta na *Teoria dos Registros de Representações Semióticas* proposta por Raymond Duval, com isso, de acordo com Siqueira (2019), deve-se propiciar *atividades* que auxiliem a compreensão dos saberes a serem trabalhados pelos estudantes,

> do ponto de vista cognitivo, a atividade requerida pela matemática se difere daquelas requeridas em outras áreas do conhecimento, e apresenta duas características que evidenciam tal diferença: a importância das representações semióticas e a variedade de representações semióticas (SIQUEIRA, 2019, p. 126).

É imprescindível explicitar a distinção entre "situação de ensino", claramente do ponto de vista didático, no sentido da organização de uma aula, de uma sequência de aulas, de atividades a serem realizadas com estudantes, com a finalidade de que haja aprendizado de algum saber, ou seja, a ação que promove o ensino, envolvendo professor e aprendiz. Enquanto que "situação" denota a criação de circunstâncias nas quais cada ideia pode emergir para fundamentar e auxiliar a elaboração da situação de ensino.

Como exemplo, para desenvolver o Function Studium, Silva (2016) evoca as "atividades cognitivas" envolvidas na coordenação de duas quantidades que o auxiliam a criar as situações

de ensino do software, bem como Siqueira (2019) considerando que a "atividade matemática" diz respeito à mobilização simultânea, de no mínimo, dois registros de representação, no Conics Studium 3D; também em Silva (2019) considerando as "situações" que dão sentido a área como grandeza para desenvolver os usos do Magnitude Studium.

O que queremos trazer à tona com a ênfase nas definições de ***situações*** e ***atividades*** propostas no âmbito dos software que foram desenvolvidos é que o modo de utilização do software, seja ele pelo professor, estudante, ou qualquer outro usuário, é definido pelo referencial teórico que fundamenta a pesquisa de acordo com os saberes envolvidos. O levantamento realizado nas análises preliminares, norteando-se pelas dimensões didática, epistemológica, cognitiva e informática, dará subsídios aos engenheiros-pesquisadores para realizar as escolhas teóricas que melhor se adequem aos conhecimentos a serem abordados pelo software.

Em outras situações, por exemplo, pode ser considerado o conceito de *tarefa*, proposto por Yves Chevallard, no contexto da Praxeologia discutida pelo autor. Como outro exemplo, pode-se, também, utilizar o conceito de *modo de exploração*, advindo da Teoria da Orquestração Instrumental, construída por Luc Trouche. O que constatamos é que as escolhas teóricas dependerão das análises realizadas, bem como as predileções e preferências dos pesquisadores quanto aos saberes em questão. Essa conclusão consta na versão aperfeiçoada da Engenharia Didático-Informática.

### 7.4.2 O processo de validação de software educativo: como analisar o subjetivo?

Ao pensar no processo de validação de software, a primeira ideia que surge é a de verificação do funcionamento: menus, telas, construções, gráficos, entre outras características. A esse processo, rotineiramente, atribui-se o objetivo de verificar se o produto desenvolvido atende aos propósitos idealizados.

De fato, no campo de conhecimentos da Engenharia e Ciência da Computação, define-se essa etapa como responsável por comprovar "documentalmente que o sistema cumpre com as funções das quais foi designado, em conformidade com as especificações dos requisitos do usuário e com a garantia de segurança e rastreabilidade de informações" (MELO, 2018).

Quando utilizamos as atuais tecnologias digitais de informação e comunicação, como smartphones, computadores, tablets, etc, percebemos que os software (aplicativos) são atualizados com uma frequência bem intensa. Novas versões, novas funcionalidades, novos

menus, correção de erros, enfim, são diversas possibilidades a cada atualização. A interação do usuário com os sistemas e aplicativos, mesmo que em situações simples de utilização, proporciona feedbacks importantes para as equipes de desenvolvedores que estão trabalhando para a evolução dos produtos.

No entanto, apreciando as minúcias dos software educativos, consideramos em nossos estudos que o procedimento de validação desse tipo tão peculiar de software deve verificar se o conjunto "levantamento teórico mais experimentação" está coerente e contribui para auxiliar nas atividades de ensinar e aprender conhecimentos matemáticos. Ou seja: validar um software educativo é compreender se os requisitos que foram levantados na análise teórica, bem como se na experimentação houve a efetiva contribuição para o ensino e o aprendizado dos saberes envolvidos (TIBURCIO, 2016).

A discussão que aqui queremos levantar surgiu da observação de como ocorre a atualização de produtos tecnológicos com fins educativos. Ao perceber os múltiplos usuários dessas tecnologias: professores, pesquisadores, equipes de coordenação escolar, estudantes, etc., indagamo-nos: os software educativos são atualizados com a velocidade que os demais tipos de software são? A primeira resposta é não, e a justificativa se dá pelo fato de que as sugestões e diversas necessidades observadas pelos usuários demoram bastante para chegar às equipes de desenvolvedores. A importância das interações dos usuários pode ser destacada em Zaina (2014):

> A tecnologia evoluiu, porém o que de fato mudou foram os usuários. Hoje os usuários buscam não somente um lançamento de um computador e/ou novos software. Querem que a experiência humano-computador se torne mais agradável e tragam benefícios efetivos a ele. Este novo requisito do usuário tem pressionado as ações das empresas desenvolvedoras de software, fomentando discussões sobre a Experiência do Usuário (UX - User eXperience). UX deve ser encarado como um conceito que engloba todos os aspectos da interação do usuário com a empresa, seus serviços e seus produtos. A área de UX tem como foco atender as expectativas, qualidade da experiência, totalidade das percepções, eficiência, eficácia e satisfação emocional das pessoas. Não são apenas os objetos envolvidos no momento da interação do usuário com um software que importam, mas também os sentimentos que esse momento provocará no usuário durante e após a interação (ZAINA, 2014, p. 1).

Queremos salientar que todos os usuários, ao utilizar qualquer software ou sistema que seja, tem sugestões de melhorias, críticas, elogios, etc, a serem reportados aos desenvolvedores. Uma das vertentes do surgimento dessa área indica que foi a partir da exigência dos usuários

“quanto à qualidade, facilidade de uso, segurança e confiabilidade de dados, e com relação ao baixo custo fez com que surgisse, a partir de 1970, a Engenharia de Software” (AMARAL; GUEDES, 2008). Contudo, nem sempre o processo de retorno do usuário aos desenvolvedores é simples de ser realizado e muitas vezes a solicitação das equipes de desenvolvimento, o conhecido “reporte o erro ao desenvolvedor”, é ignorado.

A EDI tem em sua proposta a articulação de teorias sobre ensino e aprendizagem de conhecimentos matemáticos com o que se propõe na Engenharia de Software, destarte refletimos que para validar um produto criado nessa perspectiva, faz-se necessário contemplar as escolhas teóricas dos desenvolvedores e, igualmente, se as funcionalidades corroboram para que efetivamente se alcance o que fora descrito nos requisitos do produto.

As Dimensões Epistemológica, Cognitiva, Didática e Informática, descritas na EDI auxiliam o desenvolvedor no mapeamento dos requisitos, o que faz sentido considerar, no processo de validação, a eficácia do software em atendimento a esses requisitos. A Fase Experimental, também proposta na EDI, revela as contribuições e limitações do produto, o que, também, deve ser considerado no processo de validação.

Observemos, inicialmente, os pesquisadores debruçados no estudo de um determinado software. Em linhas gerais, analisam-se as possibilidades de utilização do software para determinado campo de saberes e verificam-se quais são as limitações, falhas e problemas gerais encontrados com esse uso. É importante que o feedback dessa análise seja encaminhado para os desenvolvedores a fim de que atualizações sejam realizadas com diversos olhares de vários pesquisadores, assim o processo de atualização seria enriquecido com os comentários desses usuários.

Ao considerarmos o uso de software educativos por professores, é simples observar que em suas atividades em sala de aula e, também, de planejamento, esses profissionais conseguem verificar novas possibilidades e formas de utilização dos produtos os adequando corriqueiramente para as realidades de seus estudantes. Porém, com tantas atribuições e muitas vezes distantes da produção de conhecimento científico da academia, os feedbacks por parte dos professores, em alguns casos de projetos de software, dificilmente chegam aos desenvolvedores, o que se configura com uma perda lastimável visto que o professor é usuário do software. Mas, ao mesmo tempo, o professor produz conhecimento com ele quando elabora sequências didáticas de utilização, configurando assim mais um dos entraves nas atualizações de software e sistemas no campo educacional.

Ao analisar a utilização de software educativos por estudantes, podemos dividir esse grupo em três categorias quanto a proximidade com as pesquisas na academia: estudantes da educação básica, estudantes do ensino superior e estudantes de pós-graduação. Na educação básica temos diversos produtos que são úteis para a aprendizagem: jogos, tutoriais, aplicativos, ambientes virtuais, etc., sabemos que os estudantes deste nível utilizam com frequência tecnologias digitais e suas experiências trazem, também, julgamentos que podem ser aproveitados para as atualizações dos produtos. Contudo, o incentivo à pesquisa, análise e investigação ainda é tímida no Brasil, o que faz com que esse feedback desses estudantes também não chegue com tanta facilidade aos desenvolvedores.

Quanto aos estudantes do ensino superior, o acesso à pesquisa e à produção de conhecimento faz com que observações e análises sejam realizadas com uma frequência maior e encaminhadas para as equipes de desenvolvimento. Além disso, dentro de determinados cursos, produzem-se software educativos e as lacunas encontradas em produtos já existentes muitas vezes são discutidas nos grupos de pesquisa. Ainda assim, o feedback para os desenvolvedores poderia ser mais eficaz.

Em nível de pós-graduação, tanto professores quanto pesquisadores são produtores de tecnologias, sejam elas digitais ou não. Nesse nível, existe a tendência da inovação, em que se busca desenvolver novas ferramentas que auxiliem o ensino e a aprendizagem. Referente aos feedbacks dos usuários, as experimentações realizadas nos grupos de pesquisas, com a participação de sujeitos externos, são momentos de grande valia para compreender como os usuários se comportam perante a utilização do software, surgindo assim possibilidades de melhorias, implementações, lacunas e outras situações comuns ao uso.

Além dos retornos dos usuários, cabe aos pesquisadores organizarem suas experimentações com a finalidade de colher o maior número de informações possíveis para que seja possível aprimorar o software em desenvolvimento a partir das experimentações dos mesmos. Como exemplo, podemos analisar as validações dos software Function e Magnitude Studium e do Concis Studium 3D para perceber como os desenvolvedores compreenderam o que foi proposto na EDI, bem com as experimentações que ocorreram no âmbito acadêmico.

Silva (2016) percebeu, com a utilização da EDI, que a validação é o processo no qual se analisa “o que ocorreu na experimentação e é verificado se o software atende às necessidades do ensino e da aprendizagem dos conceitos” (p. 28). Com isso, o autor iniciou essa etapa descrevendo detalhadamente o experimento que foi realizado com os sujeitos de sua pesquisa. “O experimento foi planejado para a aplicação das atividades com uma dupla de estudantes da

Licenciatura em Matemática, a fim de testar como o software contribuiria ou não para a abordagem da taxa de variação das funções afim e quadrática com eles" (p. 109). Em resumo, temos,

> A experimentação com os estudantes teve como objetivo testar o software Function Studium como ferramenta de exploração da taxa de variação das funções abordadas, servindo para apontar as implicações do seu uso em termos de contribuição ou limitação à aprendizagem do conceito da taxa.
> A aplicação das atividades foi realizada em uma sala com uma dupla de estudantes da Licenciatura em Matemática, selecionados pelos critérios de terem cursado ou estarem cursando alguma disciplina de Cálculo Básico e terem alguma experiência anterior de uso da tecnologia computacional no estudo de funções, a fim de terem um olhar mais crítico e detalhado sobre os recursos do software (SILVA, 2016, p. 109).

O autor descreveu todas as atividades e os seus objetivos, categorizando da seguinte forma: *Tempo esperado* (para a resolução da atividade proposta), *Aprendizados esperados*, *Benefícios com o uso do software* e *Dificuldades/limitações com o uso do software*. Como pode ser observado na Figura 14.

Figura 14 – Atividade proposta por Silva (2016)

1.2) Escolha outras funções afim e para cada caso teste o item (c) da questão anterior. O que você percebeu no item anterior continua válido para esses casos? O que isso sugere em relação à variação na função afim?

**Tempo previsto**: 10 min

**Aprendizados esperados**: Inferir que na função afim, variações iguais em x levam a variações iguais em f(x).

**Benefícios com o uso do software**: Variação dinâmica e conexão de ações entre o gráfico, a janela de pontos e a janela de parâmetros para testar várias funções de forma prática.

**Dificuldades/limitações com o uso do software**: O software não permite memorizar cada valor testado ou disponibilizar uma sequência com os valores testados para auxiliar o estudante na análise. Outra limitação é não permitir a variação automática de x ou de $\Delta x$, por exemplo, o que permitiria reduzir as ações do estudante na tela e manter a atenção no essencial.

Fonte: Silva (2006).

No trecho apresentado na Figura 14, observamos como fora realizada uma parte da validação do Function Studium. Em detalhes, referente às dificuldades e limitações, nota-se que o autor indicou que o software não conseguiu realizar determinado comando executado pelo usuário, bem como não houve uma variação automática entre algumas variáveis. Ora, essa descrição é crucial para que o desenvolvedor considere o que foi observado pelo autor e, por sua vez, consiga aprimorar o software em construção, legitimando assim o procedimento de validação: o software atende à proposta idealizada, mas necessita de implementações (procedimento natural em qualquer projeto de criação, configurando um ciclo).

Destacamos, também, em Silva (2016), a análise realizada concernente ao referencial teórico adotado. Em uma sessão intitulada "Discussão dos resultados em relação ao Quadro de Níveis de Raciocínio Covariacional", são descritas quais foram as contribuições e limitações da utilização do software a fim de atender o que se propõe no referencial teórico que fundamentou seu estudo. De acordo com ele, "as atividades buscaram explorar a taxa de variação dentro da perspectiva covariacional, de forma a avaliar dentro dessa perspectiva a utilização dos recursos implementados, por meio da experiência dos estudantes com eles" (p. 142). Para Silva (2016), o software se fez validado tanto com os resultados da experimentação, quanto ao atendimento aos pressupostos teóricos levantados em sua análise a priori,

> Dessa forma, o Function Studium ofereceu suporte ao raciocínio covariacional dos estudantes ao possibilitar que eles testassem essa propriedade no caso da função afim, variando o x dinamicamente e percebendo a invariância na variação de y na janela de pontos. No caso da função quadrática, os estudantes perceberam que ao variar valores de x no gráfico, com $\Delta x = 1$, os valores correspondentes de y não tinham a propriedade de $\Delta y$ ser também constante, conforme iam percebendo sua variação simultânea na janela de pontos, o que mostrou que a conexão simultânea do dinamismo do gráfico com a janela de pontos foi fundamental nesses casos.[...]
> O suporte dado pela tecnologia implementada no Function Studium foi por meio da conexão simultânea do modelo algébrico com a janela de pontos. Com essa conexão, foi possível variar os coeficientes dinamicamente e um por vez, por meio dos controles deslizantes, e visualizar o valor da taxa média que a janela de pontos exibia, possibilitando ao estudante perceber a influência de cada coeficiente separadamente, uma covariação entre a e $\Delta y/\Delta x$ e outra entre b e $\Delta y/\Delta x$ (SILVA, 2016, p. 143).

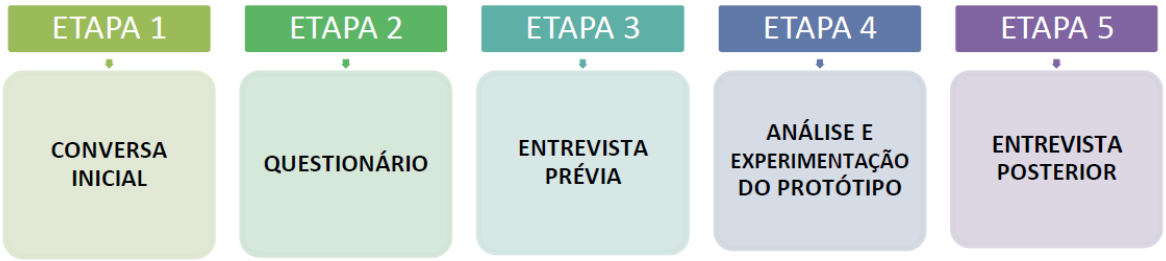
Também, é importante destacar alguns problemas observadas pelo autor, que são naturais em processos de desenvolvimento de software, visto que algumas implementações podem ser difíceis de serem atendidas por limitações tecnológicas, incompreensões do que foi solicitado, complexidade dos requisitos levantados, entre outros. Segundo Silva (2016),

> Por outro lado, algumas características do Function Studium limitaram o raciocínio covariacional dos estudantes quando eles tentaram coordenar a variação em notações distintas, o que exige um esforço cognitivo e, consequentemente, um suporte tecnológico. Foram exemplos desses fatores limitadores, a disposição das variáveis na janela de pontos/taxas, a desconexão entre as ferramentas taxa de variação e reta tangente e a ausência de alguns suportes ao raciocínio dos estudantes que poderiam ter auxiliado na coordenação da variação em notações distintas, como a memória, uma lista de valores das variáveis associadas ou até mesmo um gráfico auxiliar (SILVA, 2016, p. 156).

Com características semelhantes ao processo de validação do software Function Studium, Siqueira (2019) apresentou os elementos dessa etapa no Conics Studium 3D considerando como essencial a fase experimental do processo de software. O autor compreendeu que o processo de validação é responsável por verificar a eficácia do protótipo, se atende aos requisitos levantados inicialmente. Em específico: se pode contribuir eficazmente para o ensino das curvas cônicas (SIQUEIRA, 2019).

A princípio, o autor descreveu a etapa de experimentação do protótipo seguindo uma sequência como pode ser observado na figura a seguir:

Figura 15 – Organização das etapas para recolher dados experimentais na pesquisa




Fonte: Siqueira (2019).

Na Etapa 1, foi apresentada a pesquisa e seus objetivos para os voluntários/sujeitos; A Etapa 2 consistiu nas respostas de um formulário que teve por objetivo obter informações relacionadas ao perfil profissional de cada docente. Na Etapa 3, foram coletadas informações sobre o exercício profissional dos voluntários, tais como: funcionamento de suas aulas, os recursos utilizados para o ensino, a disposição da classe, os conhecimentos a serem trabalhados, as dificuldades de ensino e os recursos para superar essas dificuldades.

Na Etapa 4, foi realizada a experimentação do protótipo: análise das funcionalidades,

verificação das características e possibilidades de utilização, verificação da eficiência dos recursos disponibilizados visando enfrentar as dificuldades relacionadas ao ensino das curvas cônicas com respeito à articulação das suas diferentes representações. Por último, na Etapa 5, foi realizada outra entrevista sobre o protótipo a fim de compreender como os sujeitos avaliaram a experiência de utilização (SIQUEIRA, 2019).

A experimentação realizada com o protótipo Conics Studium 3D tem um caráter diferente do comum referente aos usuários. Em geral, ao se propor a experimentação de um software educativo, apresenta-se uma sequência de atividades a serem desenvolvidas pelos usuários. Entretanto, Siqueira (2019) experimentou seu protótipo com usuários que são professores e esses receberam a tarefa de analisar as possibilidades do software; e não resolver atividades com esses, no sentido de que os professores iriam pensar em situações de ensino, em atividades e em formas de utilização do software. De acordo com o autor,

> Por ter como foco um suporte computacional ao docente para auxiliar no ensino das curvas cônicas, achou-se importante que, durante a coleta de dados experimentais para a validação do protótipo, fosse possível analisar o seu funcionamento, ao mesmo tempo em que elicitaríamos requisitos necessários ao aperfeiçoamento do protótipo (SIQUEIRA, 2019, p. 246).

Com isso, a experimentação apresentou as percepções dos profissionais nos seguintes aspectos: *sobre o protótipo* - descrição de aspectos positivos e negativos; *sobre o uso do protótipo* - apresentação de possibilidades e limitações a respeito do efetivo uso do protótipo; *mapa esquemático* - produção de um mapa, estilo diagrama, contendo a representação mental do sistema de recursos utilizados pelos usuários. Como exemplo, apresentamos na Figura 16 seis funcionalidades que foram propostas para o Conics Studium 3D. As que estão assinaladas são as que a usuária conseguiu perceber que funcionam.

Figura 16 – Funcionalidades propostas para o protótipo

| | |
|---|---|
| | Articular simultaneamente as representações algébricas, gráfica e figural espacial, com registro de entrada podendo se dar por qualquer representação. |
| X | Explicitar e articular pelo menos duas representações algébricas. |
| | Evidenciar a excentricidade através de uma expressão algébrica. |
| | Oferecer opções de esconder e mostrar qualquer um dos objetos, geométricos ou algébricos. |
| X | Exibir na interface as sucessivas variações algébricas (entre todas as expressões possíveis), gráfica e figural espacial. |
| X | Oferecer interações; simulações; visualizações, explorações e manipulação direta dos objetos algébricos e geométricos. |

Fonte: Siqueira (2019).

Ao realizar a análise do protótipo do Conics Studium 3D, Siqueira (2019), também, observou se o produto auxilia no que fora levantado em seu referencial teórico quanto às dificuldades que os estudantes possuem quanto ao saber abordado. De acordo com ele, os estudantes possuem, segundo a Teoria dos Registros de Representação Semióticas, dificuldades em articular diferentes representações e, igualmente, em articular a representação quando em um mesmo registro. Assim, o autor exibe as funcionalidades do software que auxiliam os estudantes a superar essas dificuldades, como pode ser observado na Tabela 23.

Tabela 23 – Tipos de dificuldades x funcionalidades do protótipo (o que possibilita)

| DIFICULDADES | FUNCIONALIDADES PRESENTES NO PROTÓTIPO |
|---|---|
| EM ARTICULAR AS DIFERENTES REPRESENTAÇÕES | Articular simultaneamente as representações algébricas, gráfica e figural espacial, com registro de entrada por esta última. |
| | Exibir na interface as sucessivas variações algébricas (entre todas as expressões possíveis), gráfica e figural espacial. |
| | Oferecer interações; simulações; visualizações, explorações e manipulação dos objetos algébricos e geométricos. |
| EM ARTICULAR UM MESMO REGISTRO | Explicitar e articular pelo menos duas representações algébricas. |

Fonte: Siqueira (2019).

Finalizou-se o processo de validação no momento em que Siqueira (2019) elencou as impressões dos usuários sobre o protótipo e, também, sobre o uso desse, expondo características positivas e negativas listadas pelos usuários. Como exemplo, podemos perceber como o autor analisou as opiniões dos usuários, a fim de obter melhorias no software,

> Com relação às potencialidades (aspectos positivos) do protótipo do Conics, os docentes foram unânimes em indicar a possibilidade de visualização das várias representações das curvas cônicas. Além disso, foi salientada a manipulação das curvas através de alguns "parâmetros", tais como: altura do cone, inclinação do plano secção, por intermédio de um painel, e controle e sincronização simultânea entre as várias representações, permitindo que as alterações nas diferentes formas de representações sejam visualizadas. Quanto às limitações, um docente ressaltou a ausência da opção de poder ocultar alguma janela, especificamente a de notação matricial. Essa observação se justifica no contexto do ensino da Educação Básica, de maneira que o professor possa escolher quais janelas ficam ativas (SIQUEIRA, 2019, p. 282).

O processo de validação do Magnitude Studium também seguiu as diretrizes apontadas na EDI, de acordo com Silva (2019), utilizando a definição de Tiburcio et al (2015) e as observações sobre análise a posteriori e validação oriundas do estudo de Tiburcio (2016). O autor considerou que essa etapa vai além de validações técnicas e de uso em sala de aula, expondo que validar não é apenas verificar o efetivo funcionamento do software e, também, não basta apenas verificar as contribuições no uso em situações de ensino e aprendizagem. Para o autor, devem existir duas validações: uma teórica, feita pelos especialistas das áreas de conhecimento, e uma "semi-teórica", em condição de laboratório (situações didáticas específicas, professores e alunos escolhidos)" (SILVA, 2019, p. 36).

A experimentação do Magnitude Studium ocorreu, de modo semelhante ao Conics Studium, com professores e pesquisadores da área que o software foi elaborado. Silva (2019) justifica essa escolha pelo fato de os sujeitos contribuírem nesse processo considerando suas habilidades, competências e experiências na área. Em resumo, as etapas da experimentação do autor foram: *Caracterização dos Sujeitos da Pesquisa* - foram apresentadas informações sobre formação, tempo de atuação e público atendido pelos profissionais; *Descrição do Experimento* - momento em que o funcionamento do experimento foi apresentado, formulários, questões e encontros; *Análise e Discussão dos Dados* - as informações do experimento foram discutidas analisando limitações e potencialidades do protótipo.

Na experimentação, Silva (2019) estava interessado em obter dos seus sujeitos a avaliação do protótipo do software bem como sugestões de implementos a partir das experiências e formações deles, o que pode ser observado no retorno de uma das duplas como mostra a Figura 17.

Figura 17 – Retorno de uma das duplas

**Dupla 1:**

- ✓ O ícone precisa ser representativo da sua função;
- ✓ Padronizar a nomeação dos ícones para o português ou o inglês, ou seja, padronizar a língua oficial do MS. Fazer a correção dessas nomeações (Por exemplo, poliminós está sem o acento, assim como circunferência; a trena está nomeada em inglês, *measurement*, que na realidade significa medição; trena em inglês seria *measuring tape*; também no quadro das medidas, a palavra ponto aparece em inglês). Mesmo se o inglês for mantido como a língua do MS será necessário rever a nomenclatura de todos os ícones. Por exemplo, *line* expressa uma linha qualquer entre dois pontos, seja um segmento ou uma curva, mas para se referir a uma linha reta o correto seria utilizar *straight line*;
- ✓ Colocar instruções para cada ícone, como realizado em outros softwares como o Geogebra e Cabri.

Fonte: Silva (2019).

É importante observar que Silva (2019) considera as observações realizadas pelos sujeitos da experimentação do Magnitude Studium propondo implementações a serem realizadas no software, como pode ser observado no trecho a seguir:

> A partir das respostas dadas pelas duplas a este questionário, pudemos observar com mais ênfase a necessidade da implementação de ferramentas no MS que ampliem o trabalho com o perímetro de figuras planas, além das que já havíamos inserido (fita métrica e a decomposição dos lados da figura). Outro fato que nos chama a atenção em termos de usabilidade é a implementação de algumas funcionalidades, como zoom in e zoom out, voltar passos da construção, deletar a construção tudo de uma única vez. Todos esses elementos deverão ser levados em conta para a evolução do software (SILVA, 2016, p. 310).

Também na validação do Magnitude Studium, existe, como no Conics Studium 3D, a preocupação com o atendimento do software aos requisitos teóricos levantados nas análises preliminares. Silva (2019) apresenta aos participantes do experimento um texto explicativo sobre a Teoria dos Campos Conceituais e, após a leitura, solicita que respondam o seguinte questionamento: “Em sua opinião o Magnitude Studium oferece condições ao professor para elaborar tarefas sobre área relacionadas aos tipos de situações supracitados? Justifique sua resposta”. A seguir, o texto utilizado para essa etapa da experimentação é apresentado na Figura

18.

Figura 18 – Texto apresentado aos sujeitos da experimentação do Magnitude Studium

**Teoria dos Campos Conceituais (VERGNAUD, 1996)**

O conceito de área é constituído por uma tríade C= (S, I e R), na qual **"S"** é o conjunto de situações que dão sentido ao conceito (a referência), **I** é o conjunto de invariantes sobre os quais se assenta a operacionalidade dos esquemas (conceitos em ação e teoremas em ação) (o significado) e **R** é o conjunto das formas pertencentes e não pertencentes à linguagem que permitem representar simbolicamente o conceito, as suas propriedades, as situações e os procedimentos de tratamento (o significante). Baseada nessa teoria, Baltar (1996) desenvolveu em sua tese de doutorado um conjunto de situações que dão sentido ao conceito de área e que fora ampliado nos estudos de Ferreira (2010), a saber:

**Situação de comparação** trata-se em primeiro plano de decidir se duas figuras pertencem ou não a uma mesma classe de equivalência (ou seja, se têm mesma área) ou de estabelecer uma ordem entre as áreas de duas ou mais figuras.
**Situações de medida**, o resultado esperado é um número seguido de uma unidade de área e, nesse caso, o que está em foco é a passagem da grandeza ao número mediante a escolha de uma unidade.
**Situações de produção** são aquelas nas quais se solicita que sejam desenhadas ou descritas figuras que respeitam determinadas condições: por exemplo, traçar no papel quadriculado uma figura de área 10 $cm^2$; dada uma figura F, desenhar uma figura G, diferente de F, com mesma área; ou uma figura J com perímetro igual ao de F etc.
**Situações de mudança de unidade**, uma área é inicialmente representada utilizando certa unidade U e solicita-se que seja expressa a mesma área, com o uso de uma unidade diferente de U, como por exemplo, expressar em hectares a área de uma região de três quilômetros quadrados. (SILVA, BELLEMAIN, 2017, p. 6).

Fonte: Silva (2019).

Com a utilização do texto, o objetivo do autor foi o de fazer com que os sujeitos tivessem compreensão do referencial teórico de suporte para a construção do Magnitude Studium, visto que se considera que na experimentação os participantes deveriam observar todos os elementos que fossem possíveis do software em desenvolvimento. De acordo com Silva (2019),

> Nas respostas dadas pelas duplas, identificamos as mais diferentes maneiras oferecidas pelo MS para se construir as situações que dão sentido à área como grandeza. Todavia, deixam clara a fragilidade do software no que diz respeito às escalas das malhas, a não possibilidade de mudança da unidade convencional, como também criar uma figura tendo sua área previamente estabelecida. As duplas solicitam ainda que possa ser ampliado o trabalho com as grandezas área e perímetro relacionado às figuras não poligonais. Sugerem, por fim, que essas observações sejam

> levadas em conta visando ao aprimoramento do MS para o ensino de área e perímetro (SILVA, 2019, p. 277).

Um dos acontecimentos interessantes que merecem destaque no processo de validação de Silva (2019) é o reconhecimento, por parte do autor, das lacunas que o software possui e das melhorias que devem ser implementadas. Isso que nos faz refletir sobre a importância dessa etapa no desenvolvimento de outros software educativos, a saber

> Na fase de experimentação, discutimos como o protótipo foi testado por especialistas em grandezas e medidas e em informática e educação matemática. Os resultados dessa fase culminaram na Análise à Posteriori e Validação, as quais indicam que o protótipo atendeu parcialmente ao objetivo especificado na sua concepção.
> As análises dos especialistas destacaram a fragilidade do software com relação à construção conceitual do perímetro, visto que há uma restrição das figuras a serem construídas (figuras não poligonais e a composição de figuras poligonais com não poligonais) (SILVA, 2019, p. 313).

Com isso, percebemos a importância desse processo. Silva (2019), utilizando-se da validação proposta na EDI, conseguiu levantar elementos a serem considerados para implementação do Magnitude Studium. Essa forma de validação, presente nos projetos de software aqui discutidos, proporciona aos desenvolvedores compreender não somente se o software funciona (seus menus, imagens, comandos, etc), mas se tem potencial suficiente para atender o que se solicita nos estudos teóricos dentro das diversas áreas de conhecimentos.

As estratégias utilizadas por cada desenvolvedor não foram previstas no processo da Engenharia Didático-Informática, o que nos fez refletir na possibilidade de lançar uma nova versão considerando uma apresentação com mais elementos para a etapa de validação. Isso pois observamos que cada pesquisador fez de uso de suas experiências, competências e habilidades para validar o software: uma consequência da EDI não ter apresentado, de forma clara, como proceder nessa etapa do processo. Resumimos na Tabela 24 as etapas de como cada desenvolvedor validou os software desenvolvidos.

Tabela 24 – Procedimentos de validação

| ETAPA | DESCRIÇÃO |
|---|---|
| Análise dos dados da experimentação | *Apresentação do experimento e das atividades; Caracterização dos sujeitos; Análise das interações com os sujeitos.* |
| Análise do uso do software | *Descrição dos benefícios e limitações do produto em teste; Análise dos requisitos técnicos.* |
| Confronto com os pressupostos teóricos | *Verificação do atendimento aos requisitos teóricos;* |
| Análise geral do software | *Conclusão do processo de análise do software; Descrição dos detalhes da experimentação: possibilidades e entraves.* |

Fonte: o autor.

O processo de validação, não só de software, mas de todo o conhecimento produzido na Engenharia de Software Educativos, foi alvo de investigações do grupo de pesquisas que o autor desse texto participa. Com o objetivo de compreender a natureza da validação das produções acadêmicas e científicas da ESE, Bellemain et al (2017) realizou uma análise bibliográfica em que "foram analisadas produções na área de desenvolvimento de artefatos tecnológicos com o objetivo comum de contribuir para o ensino e a aprendizagem de conhecimentos matemáticos" (p. 1). Foram estudadas dissertações cuja temática principal era a concepção e construção de software educativo para auxiliar nas relações de ensino e aprendizagem da Matemática.

Desse estudo, percebemos um determinado "padrão" ou "regularidade" para a validação das produções. O Diagrama 13 apresenta as características comuns nas investigações que foram analisadas.

Diagrama 13 – Padronização dos estudos


Fonte: Bellemain et al (2017).

De acordo com a investigação discutida, os estudos se iniciam com a constatação de um determinado problema no ensino ou na aprendizagem de algum conhecimento e, ao serem verificados esses problemas, surgem hipóteses para a superação desses. As hipóteses, por sua vez, levam em consideração os recursos tecnológicos atuais, seja o que eles podem oferecer, bem como as limitações, justificando a construção de novos produtos. Com a análise teórica realizada, dá-se início à fase de desenvolvimento. Os referenciais teóricos, bem como a revisão de literatura, são articulados para fundamentar a criação das novas tecnologias (BELLEMAIN et al, 2017).

Ainda sobre o Diagrama 13, as duas últimas fases se referem a fase experimental e a análise dos resultados, respectivamente. Nesse momento, o produto é posto à prova para verificar se suas funcionalidades estão de acordo com os objetivos no momento da criação do produto. Por último, avalia-se não só o produto, mas toda a investigação que fora realizada. Mesmo assim, apresentaremos de forma objetiva, na perspectiva de aperfeiçoamento da Engenharia Didático-Informática, todos os procedimentos, a delimitação de etapas, procedimentos e como efetivamente deve ocorrer a validação.

# 8 MODELO DE PROCESSO DE SOFTWARE DA ENGENHARIA DIDÁTICO-INFORMÁTICA

Apresentamos nesse capítulo, a metodologia da Engenharia Didático-Informática para criação de software educativo. Dentro dos princípios teóricos e metodológicos discutidos até aqui, temos o modelo de processo de software que apresenta as etapas e procedimentos para a atividade de criar um recurso tecnológico que auxilie as relações de ensino e aprendizagem.

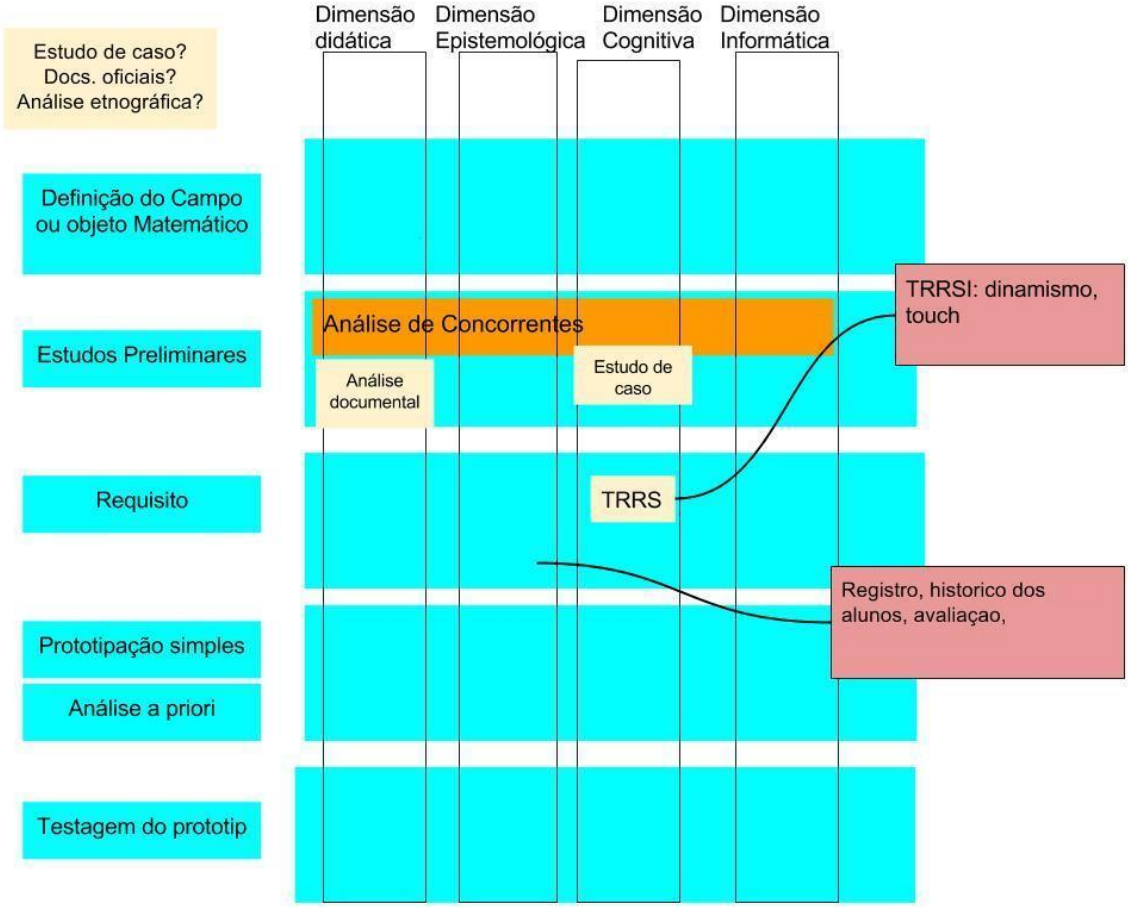
## 8.1 EVOLUÇÃO DO MODELO DE PROCESSO

O modelo de processo de software da Engenharia Didático-Informática sofreu diversas reestruturações desde seu primeiro esboço ainda enquanto era formulado. As primeiras estruturas concebidas até as mais recentes, apresentada nos estudos de Tiburcio (2014, 2016) e Tiburcio e Bellemain (2016, 2018), passaram por análises e atualizações para melhor atender as demandas do desenvolvimento de software educativo. A seguir apresentamos as versões iniciais e a atual, criada no âmbito da investigação aqui discutida.

A primeira versão carecia de efeitos visuais sofisticados, construída em uma discussão fundamentada nas hipóteses dos pesquisadores envolvidos no processo de criação, tal produção foi significativa para a criação da versão seguinte do modelo de software, onde, percebeu-se que,

> Os procedimentos observados na figura (definição do campo ou objeto matemático, Estudos preliminares, Requisito, Prototipação simples, Análise a priori e testagem do protótipo) relacionam-se com as dimensões assinaladas na horizontal (Didática, Epistemológica, Cognitiva e Informática) na medida em que questionamentos e observações de tais procedimentos procurassem responder as referentes dimensões. A partir da versão inicial, reformulamos visualmente o modelo do processo para que a compreensão das etapas ficasse mais bem estruturadas e visíveis (TIBURCIO, 2016, p 57).

Figura 19 – Modelo inicial de Processo de Software – EDI




Fonte: Tiburcio (2016).

Nesse modelo, havia a premissa de considerar a Teoria dos Registros de Representação Semióticas – TRRS de Raymond Duval, no entanto, ao refletir sobre as possibilidades de construção de software, optamos por deixar que as escolhas dos referenciais teóricos ficasse a critério dos desenvolvedores, conforme já mencionado, quando esses realizassem as análises teóricas pautados nas dimensões didática, cognitiva, epistemológica e informática. Isso visto que a percepção, as experiências e hipóteses dos desenvolvedores possuem grande influência nos referenciais a serem utilizados. Uma das prerrogativas da Engenharia Didático-Informática, assim como da Engenharia Didática, é que o pesquisador defina um *quadro teórico* apoiado nos conhecimentos do domínio em estudo, como já citado nesse trabalho.

Outro item reformulado foi a “análise de concorrentes”, que tinha por objetivo verificar quais as possibilidades que outros programas apresentavam referente ao que se pretendia com o software a ser desenvolvido. Justificamos a alteração do termo “concorrentes” visto que não consideramos que exista uma competição entre os produtos, embora ponderamos que esse tipo

de análise seja relevante. Consideramos, assim, uma *análise externa*, a fim de verificar quais benefícios, características e possibilidades que outros software podem oferecer e que contribuam para o andamento do projeto em curso em uma perspectiva de inovação – a prerrogativa é que o software em desenvolvimento tenha diferenciais quanto ao que já se está posto. A análise externa acontece após o levantamento dos requisitos pois esses servem de parâmetro de análise dos produtos existentes. Nesse sentido, essa etapa serve como uma primeira avaliação do software que se pretende desenvolver, averiguando o que esse trará de diferencial quanto ao que já existe.

Na figura a seguir apresentamos o modelo de processo que foi concluído no estudo de Tiburcio (2016) e utilizado para a criação dos software que foram discutidos nessa pesquisa (Function Studium, Conics Studium 3D e Magnitude Studium).

Figura 20 – Processo de desenvolvimento de software educativo


Fonte: Tiburcio (2016).

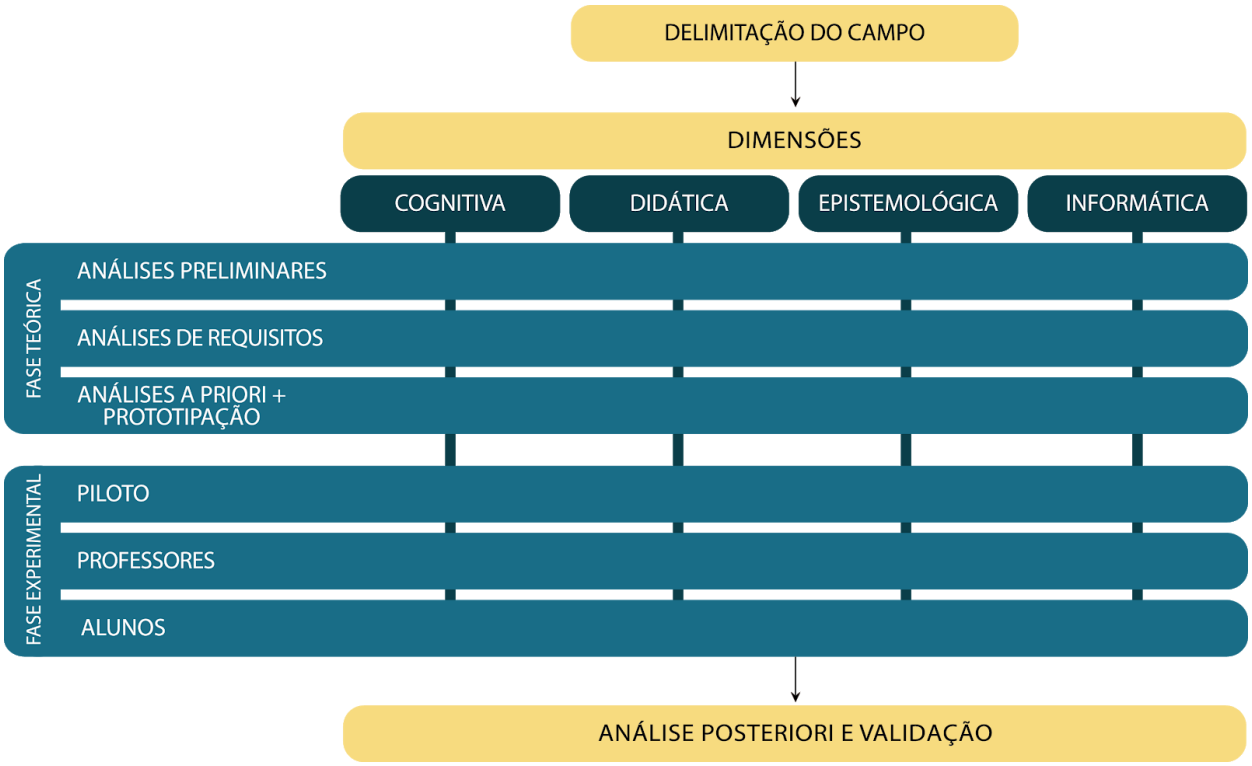
Além do aperfeiçoamento visual, os procedimentos foram divididos em duas fases: a Teórica e a Experimental. Esse modelo apresentava as etapas norteadoras da elaboração, porém

cada etapa ilustrada fazia o processo ser rígido, como se houvesse uma sequência obrigatória a ser seguida. Essa percepção de sequenciamento foi verificada ao longo das análises dos projetos de software discutidos nesse estudo, assim a reconfiguração do modelo se deu com os resultados dessa pesquisa.

Percebemos que o desenvolvimento de software necessita de etapas lógicas e sequenciais, porém consideramos no modelo de processo atual uma sequência cíclica, onde é possível articular todas as etapas considerando as interseções, conexões e relações entre as fases de construção. A rigidez de um processo de software impossibilita retornos e reformulações de requisitos, objetivos, características e outras especificações do software.

A última versão do modelo, antes da mais recente lançada nessa pesquisa, passou por reformulações visuais e a “escolha do tipo de software” foi retirada. Essa exclusão se deu visto que na especificação do software e no momento que se consideram os requisitos, a tipologia do produto será definida verificadas as características que os desenvolvedores perceberam que o projeto deve conter. Ao considerar os requisitos, esse apanhado trará subsídios para determinar qual tipo de software será desenvolvido. Embora a EDI, inicialmente, tenha como produto principal o desenvolvimento de ambientes de simulação/micromundos, outras tipologias podem ser consideradas, em caráter experimental, para serem desenvolvidas utilizando essa metodologia.

Figura 21 – Último modelo de processo da EDI

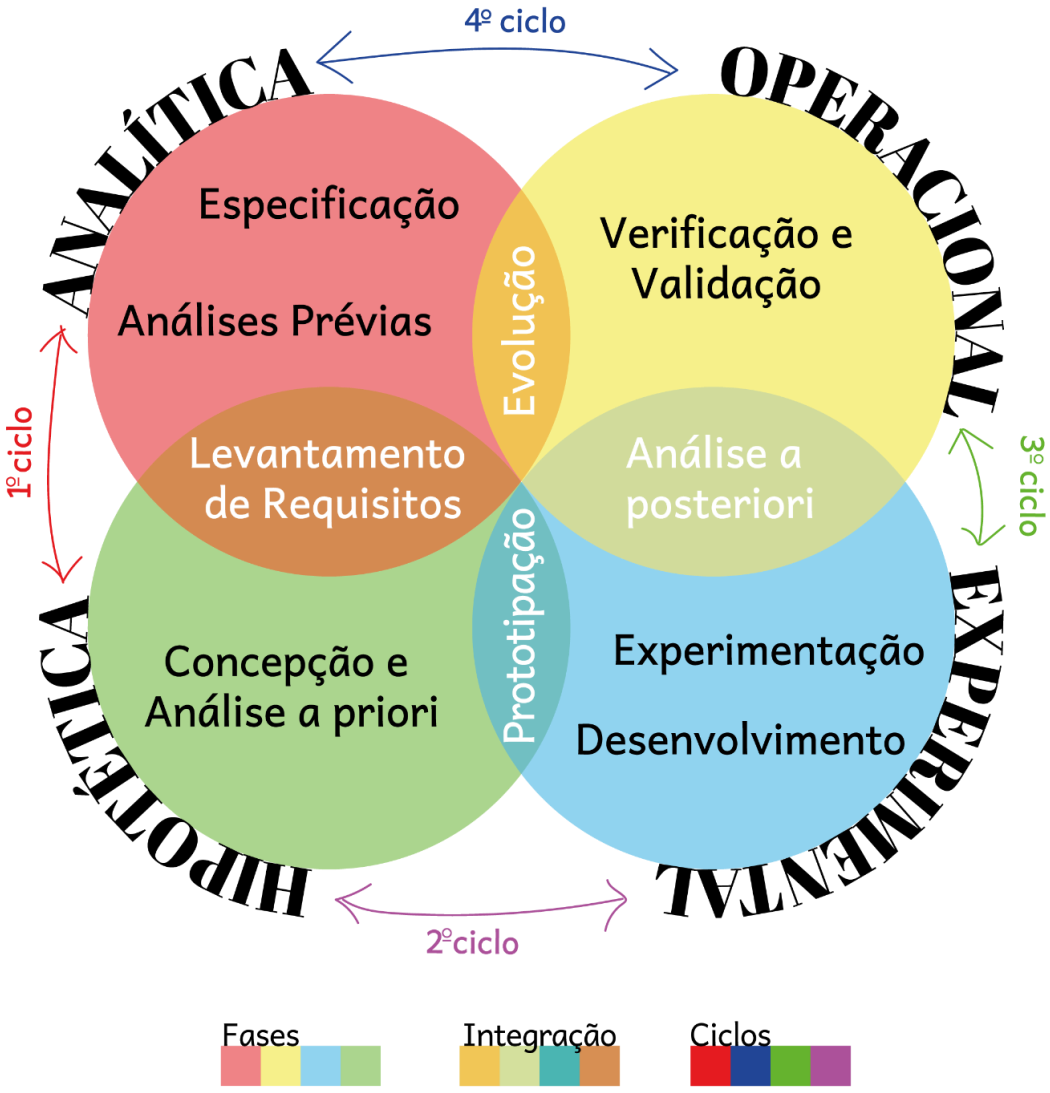



Fonte: Tiburcio (2018).

Quanto ao layout, realizamos uma modernização no sentido dos formatos e cores. Entretanto, além do layout, consideramos nessa pesquisa a estrutura sequencial que apresenta uma ideia de ser rígida e inflexível. Assim, atualizamos as formas e como exibir os procedimentos para que seja possível compreender as articulações e interseções entre todas as etapas do processo. Na sessão a seguir, designamos a nova versão do modelo de processo da Engenharia Didático-Informática e todas as suas fases, etapas e procedimentos.

## 8.2 ATUAL MODELO DE PROCESSO DA ENGEHARIA DIDÁTICO-INFORMÁTICA

Figura 22 – Modelo de Processo de Software – Engenharia Didático-Informática




Fonte: o autor.

Esse modelo é composto por quatro fases e quatro ciclos. Nas sessões seguintes, descreveremos cada uma delas bem como as etapas para desenvolver software educativo baseado na Engenharia Didático-Informática. As fases são: *analítica*, *hipotética*, *experimental* e *operacional*.

Os ciclos são formados pela integração das fases. Ao observamos modelos de processos de software rotineiramente utilizados nas engenharias de produtos (educativos e não educativos) percebemos que havia certa rigidez entre as etapas e procedimentos e retornos às

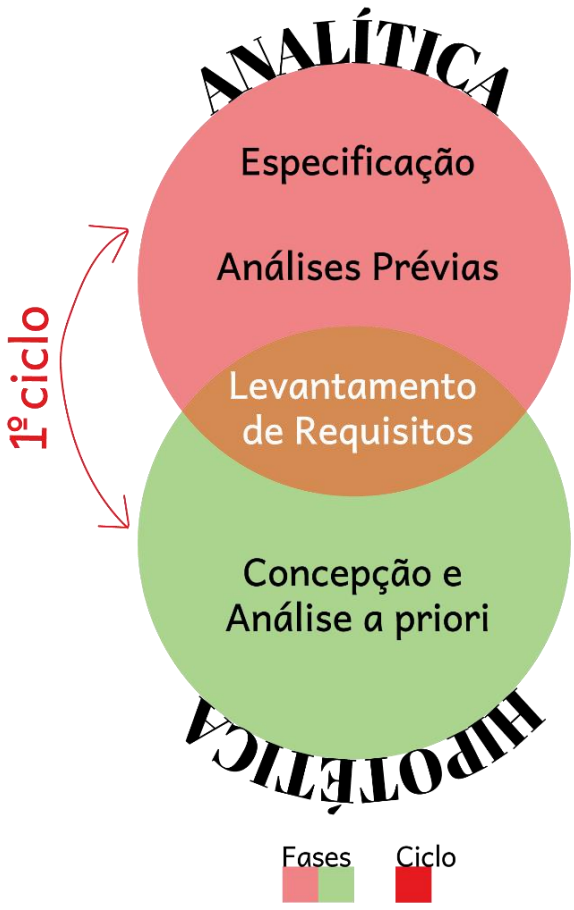
etapas anteriores eram desconsiderados. Com isso, cada ciclo da EDI reflete um avanço no processo, porém, de forma simultânea, as fases são consideradas e revisitadas. Os ciclos são: *analítico-hipotético*; *hipotético-experimental*; *experimental-operacional*; *operacional-analítico*.

## 8.3 DESENVOLVENDO SOFTWARE EDUCATIVO COM A ENGENHARIA DIDÁTICO-INFORMÁTICA

Apresentamos nesta sessão como interpretar o modelo de processo da Engenharia Didático-Informática, suas fases, etapas e encaminhamentos. Para fins de compreensão, chamaremos de "líderes do projeto" os responsáveis e interessados por desenvolver software educativo considerando essa metodologia.

### 8.3.1 Ciclo analítico-hipotético

Figura 23 – Primeiro ciclo



Fonte: o autor.

Neste ciclo, é realizada a ***especificação*** inicial – delimitam-se os problemas existentes que o software a ser desenvolvido auxiliará na solução. Tais problemas podem ser de diversas naturezas: dificuldades de aprendizagem; ensino pautado em situações que não contribuem para a compreensão do conhecimento; percepção da contribuição tecnológica para auxiliar na aprendizagem e/ou ensino, entre outros. É importante considerar nessa etapa quais serão os potenciais usuários do produto a ser construído, visto que o programa deve ser criado com discernimento a quem se destina.

Define-se, também, nessa etapa. quais serão os saberes abordados e, de forma hipotética, já deve ser considerado como as tecnologias podem auxiliar na compreensão desses conhecimentos.

Os questionamentos a seguir norteiam a fase de especificação:

*Quais são os problemas percebidos que o software poderá se apresentar como solução?*
*Quais conhecimentos se pretende abordar na utilização do software?*
*Considerando as relações entre os saberes delimitados, quais conceitos e definições devem estar presentes?*
*Qual será o diferencial da utilização desse software comparado a um ambiente papel e lápis?*

Definidas algumas delimitações, faz-se necessário que os líderes do projeto pensem na ***composição da equipe***. A EDI considera composição de equipes transdisciplinares fazendo com que os saberes das áreas envolvidas sejam integrados numa perspectiva de união para que novos conhecimentos sejam criados. Além disso, os grupos de desenvolvimento devem ser compostos por possíveis usuários, sejam eles pesquisadores, professores, estudantes, entre outros, com o propósito de enriquecer o desenvolvimento com os retornos, percepções e experiências desses integrantes.

É proveitoso pensar nas contribuições que cada integrante da equipe poderá trazer. Consideramos que a participação de alguns profissionais é indispensável: um pesquisador/professor que possua compreensão sobre a área de conhecimentos que o software abordará; um profissional da área da Ciência da Computação (engenheiro de software/programador/outros) que compreenda as características e requisitos do software a ser

desenvolvido e realize a transposição para os meios digitais; e os usuários que já foram citados.

Profissionais como designers, ilustradores, psicólogos e outros que podem contribuir para a criação de software, também podem ser integrados ao projeto de desenvolvimento. Consideramos eles importantes, entretanto, visto o cenário de recursos e de viabilidade da participação, percebemos que não é algo simples envolver tantos profissionais. Acrescentamos, ainda, que a ausência de algum desses profissionais não inviabiliza que os conhecimentos dessas áreas sejam considerados no desenvolvimento do software. Levantamentos bibliográficos podem auxiliar na compreensão de características pertinentes dessas áreas, sendo assim implementadas mesmo sem a efetiva participação dos referidos profissionais.

Finalizada a composição da equipe, as ***análises prévias*** são iniciadas. Nessa etapa é realizado um levantamento analítico com o intuito de compreender os encaminhamentos das dimensões Didática, Epistemológica, Cognitiva e Informática do conhecimento delimitado a ser trabalhado no software. Essas análises devem contemplar os resultados de pesquisas, estudos e investigações sobre os saberes que serão abordados no software. Os questionamentos, exibidos na Tabela 25, orientam como realizar esta etapa.

Tabela 25 – Direcionamento para as análises prévias

| *DIMENSÕES* | *QUESTIONAMENTOS* |
|---|---|
| *Cognitiva* | *Existem indicações na literatura de como o estudante aprende?*<br>*Quais dificuldades de aprendizado são identificadas?*<br>*Quais etapas são elencadas para a construção do conhecimento?* |
| *Didática* | *Qual é o estado atual do ensino do conhecimento?*<br>*Quais são as consequências desse ensino?*<br>*Quais são as dificuldades em ensinar esse conhecimento?* |
| *Epistemológica* | *Quais intervenções são realizadas para adaptar o saber matemático ao saber a ser ensinado?*<br>*Quais são os aspectos do conhecimento que podem dificultar e/ou facilitar a aprendizagem?* |
| *Informática* | *Quais são as contribuições tecnológicas que o software deve conter para auxiliar na compreensão e ensino dos conhecimentos?*<br>*Em que aspectos as tecnologias digitais influenciam no currículo e nas mudanças das práticas docente e discente?* |

Fonte: o autor.

Esses questionamentos direcionam a equipe a compreender os requisitos iniciais. Destacamos que uma das premissas da EDI é considerar a efetiva participação dos usuários no processo de concepção do produto. Assim, tais perguntas, elencadas nas dimensões, também devem ser respondidas pelos usuários, sejam eles pesquisadores, professores e/ou estudantes, e suas respostas consideradas no desenvolvimento do software.

Concluída a primeira fase da análise, a equipe terá diante de si uma descrição de características que o software deve conter para auxiliar nos problemas observados nas dimensões citadas. Essa descrição é um primeiro documento para iniciar o estudo de requisitos do software, que deve estar com o máximo de detalhes possível e com uma linguagem clara para que não haja falhas na comunicação e as características sejam compreendidas por todos que compõe a equipe.

Decidimos por criar uma conexão entre a fase analítica e a hipotética, visto que existe a necessidade de criar hipóteses a partir do que fora levantado nas análises prévias, mas, simultaneamente rever as especificações, caso necessário modificá-las, para posteriormente iniciar o protótipo do software idealizando situações de uso que, hipoteticamente, podem auxiliar nas relações de ensino e aprendizagem do que fora delimitado.

Com isso, o ***levantamento de requisitos*** é realizado observando o que foi verificado na fase analítica. A engenharia de requisitos deve se guiar pelo rol de contribuições percebidas nas dimensões Didática, Epistemológica, Cognitiva e Informática; a fim de explicitar o maior número de subsídios sobre os conhecimentos que o software a ser produzido abordará. Sabe-se que os requisitos podem ser modificados considerando as hipóteses e a experimentação, por esse motivo o documento de requisitos deve ser de fácil compreensão de modo que esteja suscetível a mudanças.

Fazem parte da atividade de levantamento: obter, analisar, especificar e validar os requisitos. Além disso, considerando um processo cíclico, a Engenharia Didático-Informática orienta que o gerenciamento dos requisitos deve ser realizado durante todo o ciclo de vida do software. Resumimos, a seguir, as etapas de como obter os requisitos, considerando as dimensões da EDI e questionamentos norteadores em cada uma delas.

1. Realiza-se uma estimativa para que se verifique as necessidades dos usuários que foram identificadas e como essas podem ser satisfeitas utilizando as atuais tecnologias de hardware e software; *Como o ensino e a aprendizagem podem ser favorecidos? Como*

*a compreensão dos saberes é auxiliada com o uso do software? Quais recursos e situações o software propõe para ajudar o usuário a compreender os conhecimentos?*

2. É iniciado, nessa etapa, o processo de análise externa. Consideram-se aqui as possibilidades de outros software que versam sobre o mesmo conhecimento do produto que se pretende desenvolver a fim de propor situações e funcionalidades que vão além do que já está disponível; *Quais funcionalidades existem em produtos da área? Quais são os possíveis diferenciais do software que se pretende desenvolver? O que o software trará de novo referente ao que já existe?*
3. Neste momento são colocadas, em linguagem clara e objetiva, as informações coletadas durante as etapas anteriores criando, assim, o documento de requisitos. Neste documento devem ser exibidas as características que o software precisa conter para alcançar os objetivos especificados. Orienta-se que o documento de requisitos seja delimitado considerando as dimensões Didática, Cognitiva, Epistemológica e Informática, além de outras que podem surgir ao longo da análise teórica. Indicamos que se construa uma tabela, como a seguir, para elencar os requisitos considerando as dimensões citadas e também elementos de outras naturezas.

Tabela 26 – Sugestão de documentação de requisitos

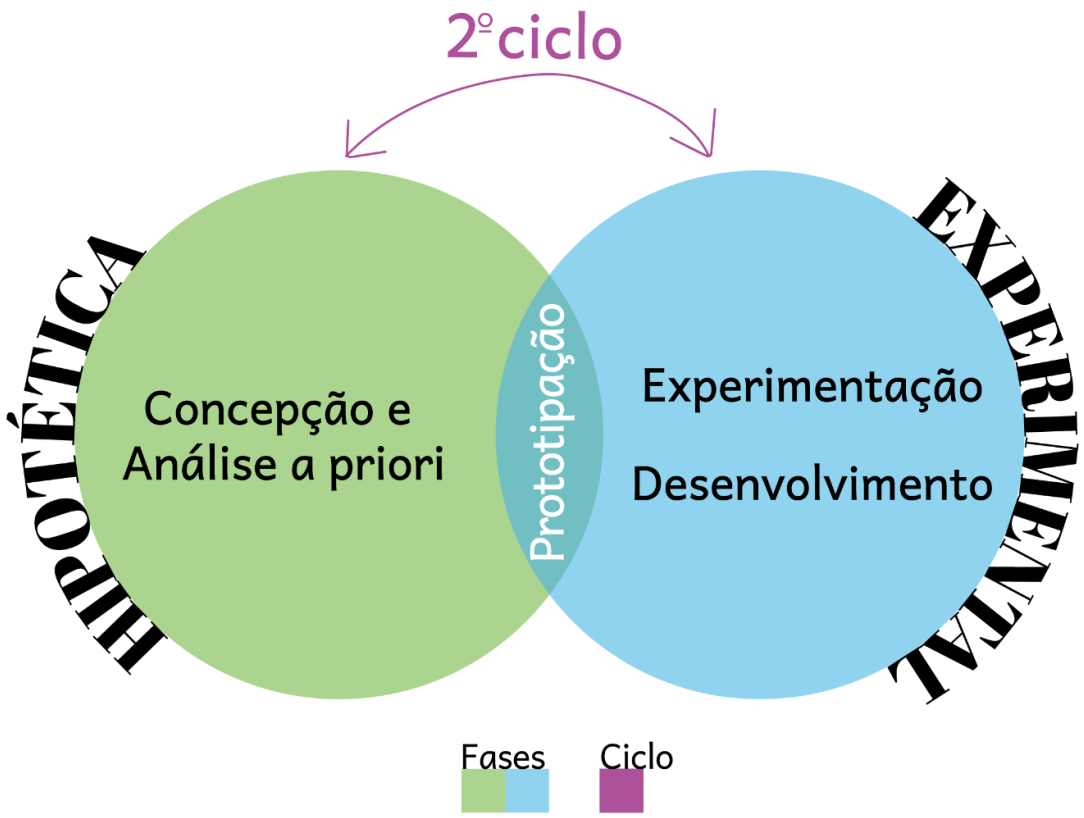
| DIDÁTICOS | COGNITIVOS | EPISTEMOLÓGICOS | INFORMÁTICOS | OUTRAS NATUREZAS |
|---|---|---|---|---|
| | | | | |

Fonte: o autor.

4. Neste momento, a equipe irá verificar os requisitos quanto à pertinência, consistência e integralidade. Neste processo podem ser descobertos erros quanto aos requisitos levantados. Assim, os requisitos podem ser modificados, a fim de corrigir os problemas encontrados.

### 8.3.2 Ciclo hipotético-experimental

Figura 24 – Segundo ciclo




Fonte: o autor.

Neste ciclo são idealizadas: as situações de uso, as hipóteses de interações dos usuários com o sistema, os problemas que podem surgir com a utilização do software e o desenvolvimento do protótipo para iniciar os testes na etapa seguinte.

Na ***concepção e análise a priori,*** são desenvolvidas as situações de utilização do software, os referenciais teóricos e metodológicos, elencados pela equipe, que, em hipótese, auxiliam no ensino e na aprendizagem dos elementos a serem trabalhados, são considerados e as interações com os usuários, a serem desenvolvidas, devem estar fundamentadas nesse levantamento. Essas situações devem ser criadas ao mesmo tempo em que o protótipo do software é idealizado, visto que a prototipação serve para que os profissionais de design e arquitetura de software compreendam os requisitos e funcionalidades e consigam executá-las com o que se dispõe de hardware e software.

As situações de utilização, bem como o protótipo, devem ser construídas objetivando

superar os problemas de ensino e aprendizagem (e de outras naturezas) descritos pela equipe considerando o levantamento teórico nas dimensões da EDI e, principalmente, analisando o documento de requisitos. Esse procedimento serve para fundamentar o software na base teórica e metodológica em pesquisas com resultados consolidados que foram analisadas pela equipe.

Indicamos que a prototipação seja feita "em telas" – os líderes da equipe, de posse dos requisitos, podem utilizar editores de textos ou imagens para simular como será o ambiente a ser desenvolvido, com as funcionalidades, botões, menus, etc. Assim exibindo as telas iniciais do produto e como algumas de suas funções a serem executadas.

Consideramos o ciclo hipotético-experimental visto que o protótipo do software está na interseção entre o que se considera como hipótese e o que será desenvolvido, uma vez que há a expectativa de que o produto atenda aos objetivos levantados pela equipe. Dessa forma, as situações que foram idealizadas são implementadas no software (ou não) ao passo que a experimentação acontece. Os testes iniciais, o feedback da utilização, as percepções dos membros da equipe, as compreensões e incompreensões das funcionalidades, tudo isso acontece de forma simultânea à elaboração do produto. O software deve ser criado considerando a validação e invalidação das hipóteses e requisitos, ambos testados em caráter experimental.

Com as ideias do protótipo fundamentadas, é dado início ao design e arquitetura do software, o ***desenvolvimento***. Definem-se, aqui, considerando o protótipo como base, os componentes e demais características do sistema. Esta etapa tem como resultado tanto componentes internos – como especificações de sistemas operacionais, linguagem de programação, definição de hardware, entre outras, e externos – como as interfaces, layout, objetos, relações entre objetos, funcionalidades, entre outros.

Acreditamos que a etapa do desenvolvimento ocorre de forma simultânea à ***experimentação***, visto que a utilização do protótipo nas situações idealizadas pela equipe trará subsídios para nortear características que o produto deve conter, como funcionalidades, especificações e possíveis novos requisitos. Com isso, o protótipo desenvolvido precisa ser utilizado pelos possíveis usuários e essas experiências devem ser registradas a fim de obter elementos de análise e implementação do software.

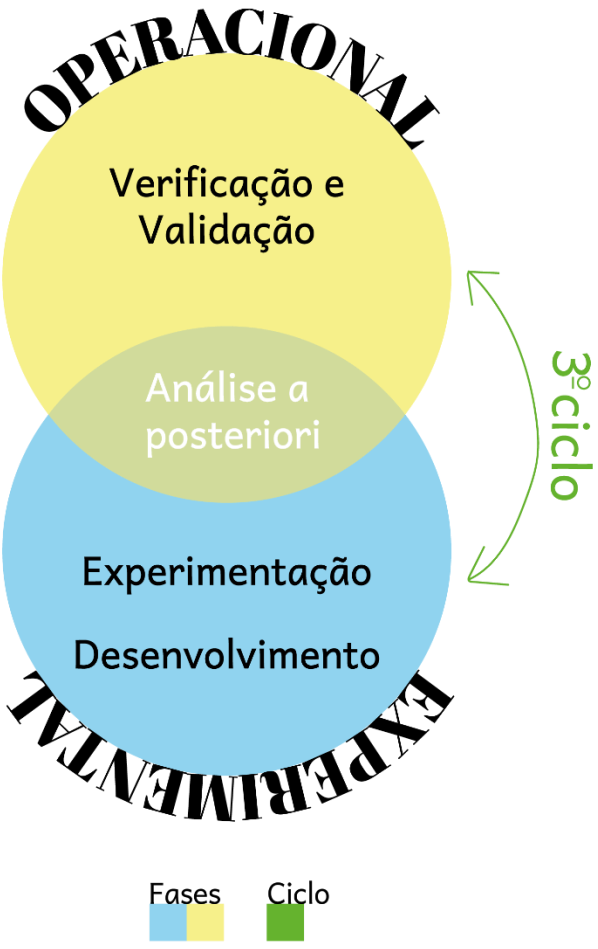
Deste modo, para iniciar a experimentação, a equipe deve criar o *manual do usuário*, ou, no mínimo, um documento (vídeo, gráfico, etc) de caráter instrutivo apresentando as funcionalidades básicas e o que será utilizado na experimentação. É necessário que as instruções sejam claras e concisas para evitar incompreensões nos usuários. Orientamos que esse

documento contenha:

1. Guia de instalação;
2. Espaço de armazenamento necessário para instalação do software (no hardware que o receberá);
3. Sistemas operacionais compatíveis (Windows, Macbook, outros);
4. Exigência mínima de hardware;
5. Inicialização do software;
6. Demonstração de como utilizar as funcionalidades.

### 8.3.3 Ciclo experimental-operacional

Figura 25 – Terceiro ciclo




Fonte: o autor.

Este ciclo contempla a ***experimentação*** que é clássica, no sentido de colocar o software em desenvolvimento em situações reais de utilização, não considerando apenas os testes realizados pela equipe ou o ambiente de utilização simulado, mas com potenciais usuários do produto. Coloca-se em funcionamento o que foi construído, caso haja exigências de alterações, modificações e diversas outras implementações, devem ser realizadas ao passo que se experimenta.

Indicamos que seja realizada uma oficina ou um curso de curta duração em que os objetivos, tanto do desenvolvimento do software quanto ao da pesquisa associada (se houver) sejam detalhados aos participantes. Na oficina ou curso, a equipe deve exibir as funcionalidades básicas do software conduzindo os usuários a compreensão do que será realizado. A seguir, indicamos alguns procedimentos para realizar a experimentação:

**Procedimentos internos**

*organização da equipe de desenvolvedores antes da experimentação*

1. Justificativa e caracterização dos sujeitos;
2. Descrição das situações, objetivos e possíveis respostas dos usuários;
3. Elaboração de instrumentos de coleta de dados (questionários, entrevistas, etc);
4. Análise e discussão dos dados;
5. Coleta de sugestões e implementações a serem realizadas.

**Procedimentos externos**

*organização do experimento com os sujeitos*

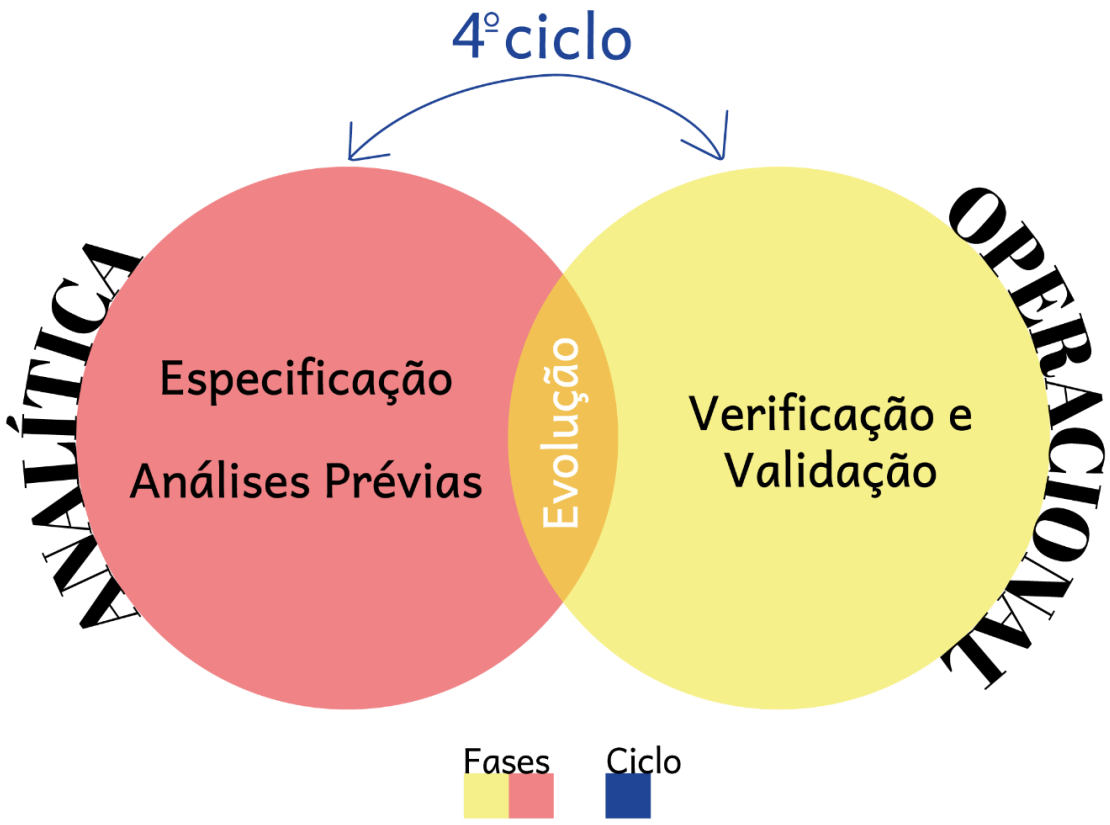
1. Apresentação do projeto e objetivos;
2. Descrição detalhada do experimento;
3. Ambientação no software;
4. Proposição das situações;
5. Oitiva dos sujeitos;
6. Análise coletiva do experimento.

Uma das finalidades da experimentação, já diretamente relacionada com a ***análise a***

***posteriori***, é a testagem da validade das situações propostas. A equipe deve observar se o software auxilia os usuários a construir os conhecimentos que são esperados. Dessa maneira, à medida em que a experimentação ocorre, informações importantes são levantadas. Por um lado, realiza-se a análise dos requisitos levantados: se eles atendem as expectativas, se auxiliam o ensino e a aprendizagem, se ajudam na compreensão dos saberes, se os recursos tecnológicos são eficazes para atender o que foi planejado; por outro lado são verificadas as falhas, sugestões de implementações por parte dos usuários e pela análise da equipe (interface, comandos, botões, menus, etc), bugs e outras inconsistências no uso, ou seja, as etapas de análise a posteriori e validação tem início ainda na experimentação, acontecem de forma simultânea.

### 8.3.4 Ciclo operacional-analítico

Figura 26 – Quarto ciclo




Fonte: o autor.

Este ciclo contempla análises conclusivas e evolutivas visto que a experimentação do software foi finalizada. A ***análise a posteriori*** consiste no confronto das hipóteses iniciais com

o que se observou na utilização do software em caráter experimental, além das implementações realizadas pela equipe. Nessa etapa é necessário que a equipe confronte o estudo teórico realizado com a experimentação para que novos requisitos e complementos sejam considerados. Têm-se, aqui, a finalidade de confrontar a análise a priori a fim de verificar o atendimento dos objetivos do software e, posteriormente, validar suas contribuições. Essa análise deve se fundamentar na observação do conjunto dos dados coletados durante a experimentação bem como todo o ciclo da EDI.

Para validar o software, a equipe deve verificar se o conjunto teórico-hipotético alcançou os objetivos esperados e se a utilização do software contribuiu para o ensino e a aprendizagem dos conhecimentos elencados. Indicamos que a validação seja realizada de duas formas: 1. Teórica – consiste em verificar se as teorias e hipóteses, com as situações e funcionalidades do software, devem ser refutadas ou aprovadas; 2. experimental – se a utilização do software apresenta contribuições efetivas para os problemas elencados na fase de sua concepção. Os questionamentos a seguir, nas tabelas 27 e 28, orientam como realizar essas análises:

Tabela 27 – Análise a posteriori e Validação Teórica – Questionamentos

| **VALIDAÇÃO TEÓRICA** |
|---|
| A utilização do software contribuiu para superar/auxiliar nos problemas elencados quanto ao ensino e a aprendizagem? |
| Foi possível contemplar todos os conhecimentos idealizados na fase de análise? |
| O diferencial entre trabalhar com um ambiente papel e lápis foi alcançado? |
| A composição da equipe de desenvolvedores auxiliou na criação do software? |
| O software apresentou diferenciais para outros que versam sobre a mesma temática? |
| Os referenciais teóricos adotados foram úteis para o desenvolvimento e criação de situações de uso do software? |
| As possibilidades de hardware e recursos digitais auxiliaram na criação do software? |
| Outros comentários |

Fonte: o autor.

Tabela 28 – Análise a posteriori e Validação Experimental – Questionamentos

| **VALIDAÇÃO EXPERIMENTAL** |
|---|
| Houve auxilio ao ensino dos conhecimentos com a utilização do software? |
| As dificuldades de compreensão dos saberes foram auxiliadas com a utilização do software? |
| As funcionalidades e recursos digitais contribuíram para as relações de ensino e aprendizagem dos conhecimentos? |
| Outros comentários. |
| Os sujeitos envolvidos no experimento contribuíram de qual modo para o desenvolvimento do software? |
| As situações propostas foram facilitadoras para compreender os conhecimentos trabalhados? |
| Quais foram as implementações que surgiram com a realização do experimento? |
| Houve incompreensões ou dificuldades de utilização do software? |
| Outros comentários. |

Fonte: o autor.

Reiteramos a atenção para a questão temporal referente a todo o processo de desenvolvimento de software: as equipes devem estar atentas e estabelecer datas, metas e prazos para concluir o desenvolvimento. Em específico, quanto a etapa de validação, as equipes devem ponderar o tempo que será dedicado a experimentação, análise dos dados e validação. Visto que, considerando as particularidades dos projetos de desenvolvimento, podem existir limites de tempo para a conclusão dos projetos e essas etapas citadas demandam um prazo relativamente longo.

Realizada a validação do software, obtém-se características a serem modificadas, alterações de diversas naturezas (funcionalidades, menus, botões, imagens, etc) são percebidas e implementadas caracterizando assim a ***evolução*** do software. Essa etapa faz o ciclo da Engenharia Didático-Informática recomeçar, visto que existe a possibilidade de uma nova especificação, no sentido de considerar outros objetivos para o software. Novos membros, também, podem entrar na equipe, observada a necessidade durante a desenvolvimento, experimentação e validação, novos requisitos podem surgir, entre outros fatores que fazem com que o processo seja iniciado novamente.

Uma das premissas da evolução é a manutenção, que tem por finalidade gerir o funcionamento do software com as sugestões oriundas dos usuários ou da equipe de desenvolvedores que continua estudando o software pensando em seu aprimoramento. Com

isso, a equipe de desenvolvedores deve considerar os quatro tipos de manutenção:

1. Corretiva: realizar os reparos identificados pelos usuários – reparos de erros, inconsistências, erros conceituais, entre outros.
2. Adaptativa: alterar características do produto para se adaptar aos ambientes (hardware, sistema, etc) em que está inserido – mudança de sistema operacional, ambiente com ou sem touchscreen, características de acessibilidade, etc.
3. Perfectiva: modificar o programa a fim de fornecer aprimoramentos aos usuários – criação de novas funcionalidades, melhoria de desempenho, alteração de layout, etc.
4. Preventiva: detectar e corrigir falhas antes que elas tomem maiores proporções – observação antecipada da equipe das funcionalidades que podem levar a erros.

É possível perceber que a manutenção e evolução estão conectadas e se configuram como uma etapa contínua, visto que a utilização do software, bem como as pesquisas e investigações de suas potencialidades, apresentarão novas possibilidades de implementação. À vista disso, os desenvolvedores podem iniciar novamente o ciclo com novas especificações, novas análises... e assim por diante.

Desse modo, concluímos a apresentação do modelo de processo de software educativo da Engenharia Didático-Informática apresentando suas fases e as orientações para desenvolver tecnologias digitais educativas que aliem elementos sobre o ensino, a aprendizagem, a epistemologia e outras diversas áreas que facilitem a compreensão de conhecimentos de naturezas diversificadas.

# 9 CONSIDERAÇÕES FINAIS

Apresentamos aqui as últimas considerações quanto aos resultados obtidos através dessa investigação. Discutimos se as hipóteses da pesquisa foram confirmadas ou refutadas e se os objetivos foram alcançados. Além disso, verificamos alguns encaminhamentos para a continuidade da pesquisa e os veiculamos na última sessão deste capítulo.

Desenvolver software educativo é uma tarefa complexa. Ao se deparar com a finalidade principal de auxiliar o ensino e a aprendizagem, o engenheiro tem diante de si questões que envolvem diversas áreas de estudo e pesquisa e que precisam ser relacionadas para alcançar um produto que contribua de forma efetiva para a atividade docente e discente. Ao se considerar a problemática da criação de recursos digitais educativos e ao verificar que muitas dessas tecnologias carecem de qualidade devido à engenharia a qual foram submetidas, além da ausência de um referencial para direcionar de forma metódica a criação de software educativo, essa pesquisa considerou investigar questões referentes a modelizações de processo de desenvolvimento, bem como apresentar uma proposta de metodologia para construção de software educativo.

Uma primeira solução para essa problemática foi, em pesquisa de Mestrado (TIBURCIO, 2016), a concepção da Engenharia Didático-Informática – uma metodologia de desenvolvimento de software educativo que considera a utilização dos procedimentos de duas engenharias: a Engenharia Didática e a Engenharia de Software, de forma a articular as análises teóricas e metodológicas de ambas. Essa metodologia foi utilizada em um estudo de caso e o resultado dessa investigação culminou na criação do software Function Studium – que apresenta uma perspectiva dinâmica da variação de funções.

A referida pesquisa de Mestrado teve como encaminhamentos futuros sugestões de aperfeiçoamento da Engenharia Didático-Informática em alguns aspectos: 1. Considerar na construção do software questões de layout; 2. Indicar a composição da equipe e como as contribuições dos profissionais devem ser organizadas; 3. Orientar, de forma concisa, como a experimentação do software deve ocorrer; 4. Detalhar cada etapa do modelo de processo de software de forma explícita. Além desses indicativos, o avanço dos estudos sobre as engenharias Didática e de Software fez com que percebêssemos a urgência de atualizar os referenciais teóricos da EDI, justificando, assim, a continuidade do estudo com a presente investigação de Doutorado.

Dessa forma, essa pesquisa teve por objetivo aperfeiçoar a Engenharia Didático-Informática, bem como o modelo de processo dessa metodologia. Assim, realizando uma abordagem histórica e analítica. Apresentamos, a seguir, considerações sobre o estudo e os seus principais resultados.

## 9.1 METODOLOGIA E CONFIRMAÇÃO DAS HIPÓTESES

Elegemos um percurso metodológico que foi dividido em duas etapas: um ***resgate histórico*** – que consistiu em investigar as engenharias que foram submetidos os software Casyopée, Function Probe e Modellus, para a concepção e construção desses; e uma ***abordagem analítica*** – com a finalidade de compreender como a Engenharia Didático-Informática foi utilizada nos projetos de concepção dos software Function Studium, Conics Studium 3D e Magnitude Studium.

Inicialmente, observando a problemática do desenvolvimento de software e a ausência de uma metodologia específica para criar esses produtos, assumimos como hipóteses: 1. compreender metodologias de construção de software educativo colabora para a criação de modelizações de processos; 2. A realização de um estudo histórico traz subsídios para o aprimoramento da Engenharia Didático-Informática; 3. A análise das utilizações da EDI, na sua primeira versão, traz elementos para aprimorar a modelização do processo de software.

Com o resgaste histórico, confirmamos a primeira hipótese da pesquisa, visto que foi possível compreender um padrão de modelização das engenharias analisadas. Em síntese, as equipes de desenvolvedores analisaram determinados cenários de ensino e aprendizagem em que observaram carências didáticas, cognitivas, epistemológicas e de outras naturezas e ponderaram como as tecnologias digitais poderiam auxiliar na superação desses problemas. Após isso, conjecturaram-se hipóteses observando possibilidades com recursos tecnológicos a fim de atender às lacunas observadas no ensino, na aprendizagem e em como características dos conhecimentos em questão trazem dificuldades de compreensão para os estudantes.

Em sequência, o desenvolvimento do software era iniciado pautado nos requisitos levantados com as análises teóricas realizadas e os programas a serem desenvolvidos eram nutridos com as características que foram observadas nos estudos teóricos sobre as relações de ensino e aprendizagem, a natureza dos saberes, as dificuldades de compreensão, entre outros aspectos que fazem o software auxiliar na compreensão de conhecimentos e nas relações de ensino.

Ainda com o resgaste histórico, ratificamos a segunda hipótese posto que os resultados desta análise indicaram que existe a atenção por parte dos desenvolvedores em considerar teorias sobre o ensino, a aprendizagem, o uso de tecnologias, as concepções, entre outras contribuições, para desenvolver recursos digitais educativos. Com esse resgate, observamos semelhanças entre as engenharias utilizadas e critérios similares de levantamento de requisitos desses software que foram úteis para o aperfeiçoamento da EDI.

Outro resultado da investigação, agora referente a abordagem analítica – onde fora investigado a aplicação da Engenharia Didático-Informática para desenvolver projetos de software, foi a compreensão de que havia a emergência de aprimoramento dessa metodologia no sentido de que fosse possível fornecer informações mais detalhadas de cada parte do processo, que houvesse uma atualização dos referenciais teóricos e metodológicos, pois, os desenvolvedores que utilizaram a EDI indicaram em seus estudos que algumas etapas não estavam objetivas ao ponto de se fazerem compreendidas, confirmando assim a terceira hipótese.

Tanto da análise das engenharias dos produtos consolidados em um resgate histórico, quanto da investigação dos produtos que usaram a EDI, em uma abordagem analítica, verificamos que ambas as engenharias ocorreram em ambientes de pesquisa. Isso facilita a criação de um ciclo de desenvolvimento, de retornos, de revisitação aos referenciais teóricos levantados, de percepção sobre contribuições, entre outros aspectos que são facilitados pelo ambiente acadêmico.

Como exemplo, observamos, na construção dos software supracitados, quando alguma das funcionalidades criadas deixava a desejar em algum aspecto, os referenciais eram reanalisados, repensados e os requisitos redefinidos – configurando um processo de idas e voltas para garantir a máxima qualidade possível do produto em criação. Além disso, após lançada a versão inicial, os programas e sua utilização continuavam sendo investigados, com implementações a partir das experiências dos usuários, dos pesquisadores e da própria equipe de desenvolvedores, fazendo com que novas análises fossem realizadas considerando mais uma vez contribuições didáticas, cognitivas, epistemológicas, tecnológicas e de outras naturezas. O ambiente acadêmico e de pesquisa possibilitam a ação de celeridade no trabalho da equipe, que está focada na pesquisa e construção do produto.

## 9.2 ATENDIMENTO AOS OBJETIVOS

Com o exposto, concluímos que a presente investigação alcançou os objetivos elencados. Aperfeiçoamos a Engenharia Didático-Informática como uma proposta de metodologia de desenvolvimento de software educativo, contemplando aspectos sobre o ensino, a aprendizagem, a epistemologia e as tecnologias digitais, além de características relevantes de outras naturezas. As implementações na EDI foram:

1. Caracterização da Engenharia Didático-Informática como uma metodologia para desenvolvimento de software educativo;

Antes da conclusão da presente pesquisa, classificávamos a EDI como “referencial teórico-metodológico” para desenvolver software educativo e o modelo de processo de software era a metodologia desse referencial. Com os estudos realizados, percebemos que a EDI se situa dentro da área de pesquisa e estudos da Engenharia de Software Educativo e é uma, das demais existentes, metodologia para o desenvolvimento de software, com um modelo de processo que orienta e especifica as etapas de criação de software.

2. Atualização dos fundamentos teóricos e metodológicos das engenharias Didática e de Software;

Ao avaliar os fundamentos teóricos e metodológicos que compõe a EDI, percebemos que alguns estudos considerados estavam desatualizados, visto que outras discussões e indicativos surgiram ao longo do tempo. Quanto a Engenharia Didática, consideramos pressupostos e aportes do que a literatura classifica como Engenharia Didática de Segunda Geração, a EDI estava fundamentada apenas com os subsídios da primeira geração (ou Engenharia Didática Clássica), assim, utilizamos o que se discute na primeira e, também, na segunda geração desse referencial.

Quanto a Engenharia de Software, consideramos estudos mais recentes, visto que especificações de programa, hardware, recursos tecnológicos e funcionalidades progridem de modo rápido e constante no contexto de evoluções das tecnologias atuais.

3. Articulação teórica explícita entre as engenharias Didática e de Software;

Ainda sobre a fundamentação teórica, percebemos a necessidade de explicitar de forma mais concisa as relações estabelecidas entre as engenharias que alicerçam a EDI. Com os estudos e investigações, percebemos que um modo eficaz de criar tecnologias digitais educativas pode ser considerado com a união de elementos da Engenharia Didática e da Engenharia de Software.

Da ES contemplamos os avanços tecnológicos, as técnicas de levantamento de requisitos, a estrutura e organização das equipes de desenvolvedores e as etapas organizacionais para a concepção e desenvolvimento de software. Já a ED ampara a compreensão de como as tecnologias podem ser utilizadas para contribuir com o ensino e a aprendizagem, trazendo a reflexão sobre aspectos didáticos, cognitivos, epistemológicos e de outras naturezas, que

auxiliam no levantamento dos requisitos do software, na criação de situações de uso, bem como na análise da utilização, e na validação do produto em construção.

4. Apresentação esclarecedora de cada fase/ciclo do modelo de processo de software;

Uma das percepções de implementação, analisando a utilização da EDI para desenvolver os software Conics Studium 3D, Function Studium e Magnitude Studium, foi relativa às fases do modelo de processo. Os desenvolvedores dos referidos projetos elencaram algumas dificuldades em como proceder em algumas das etapas, visto isso e considerando a atualização dos referenciais da EDI, apresentamos de forma detalhada como proceder em cada uma das fases e ciclos.

5. Flexibilização do modelo de processo considerando um ciclo, removendo etapas rígidas e lineares;

Um dos indicativos, originário dos estudos e discussões quanto a estrutura da EDI, foi a percepção de que as etapas e fases elencadas no modelo de processo apresentavam uma estrutura linear, como se as etapas fosse sequenciais e impossibilitassem retornos a momentos anteriores. Percebemos que até o design gráfico do modelo de processo denotava a ideia linear de segmento obrigatório de cada etapa designada.

Desse modo, com a atualização dos fundamentos teóricos, a compreensão das modelizações das engenheiras pesquisadas e da análise da utilização da EDI, optamos por considerar um processo cíclico, com um design que transparecesse essa premissa em que as fases são conectadas, dependentes e algumas acontecem de forma simultânea a outras.

Concluímos, assim, que os objetivos do presente estudo foram alcançados. A Engenharia Didático-Informática se configura como uma metodologia para a produção de software educativo que alia características das Engenharias Didática e de Software em uma perspectiva cíclica de desenvolvimento.

## 9.3 ENCAMINHAMENTOS FUTUROS

Perante a finalização dessa pesquisa, observamos questões relevantes quanto a sua continuidade e algumas lacunas que podem ser investigadas para dar seguimento a esse estudo quanto a problemática de desenvolvimento de software educativo ao passo que novas questões que norteiem outras investigações podem surgir. Além disso, considerando a constante evolução das tecnologias digitais e das pesquisas sobre o ensino e a aprendizagem, com o auxílio desses recursos, vislumbramos uma atualização constante do modo de se pensar na

criação e no uso de tais tecnologias. Organizamos a seguir algumas indicações para a continuidade da investigação nessa temática.

### 9.3.1 Pôr em prática a nova EDI

O primeiro encaminhamento que julgamos importante é quanto à utilização dos novos encaminhamentos da Engenharia Didático-Informática. Aperfeiçoamos a referida metodologia, conforme os objetivos da pesquisa, elaboramos um novo modelo de processo fundamentado nessa atualização, porém a metodologia não foi colocada em funcionamento. Desse modo, utilizar a EDI para desenvolver software educativo, micromundos/ambientes de simulação inicialmente, será de grande valia para verificar como essa abordagem metodológica poderá contribuir para o desenvolvimento dos produtos, além de perceber as limitações.

### 9.3.2 Tipologia de software

A Engenharia Didático-Informática, antes do início da presente investigação, foi utilizada para desenvolver projetos de software caracterizados como ambientes de simulação/micromundos. Assim, observamos a possibilidade de que outros estudos, pesquisas, investigações e projetos de desenvolvimento de software educativo possam utilizar a EDI a fim de que se criem outros tipos de software: jogos digitais, tutoriais em páginas online, aplicativos para smartphones, entre outros, para verificar as contribuições e limitações da EDI nas referidas construções.

Acreditamos que, à medida em que outras tipologias de software forem desenvolvidas com a EDI, teremos uma diversidade de pontos de análise para fazer com que essa metodologia seja aperfeiçoada e evoluções aconteçam. Diferentes especificações, outros referenciais teóricos considerados para o auxílio do ensino e da aprendizagem, composição de equipes de desenvolvedores com outros profissionais, outros contextos, outras vivências, outras possibilidades de compreensão de como desenvolver software educativo.

### 9.3.3 Verificar como software educativos são desenvolvidos

Em nossa revisão de literatura, constatamos que software educativos são produzidos, em síntese, considerando três orientações metodológicas: 1. Utilizando metodologias padronizadas da Engenharia de Software; 2. Adaptando as metodologias para os contextos locais de concepção; 3. Integrando conhecimentos educativos com os métodos de criação. Isso

posto, propomos que essa investigação, quanto a atual construção de produtos tecnológicos voltados para a Educação, seja expandida numa perspectiva de revisão sistemática ou outra forma que apresente uma ampla amostra de como se desenvolvem esses produtos.

Tal levantamento pode auxiliar na compreensão das percepções de professores e pesquisadores quanto à utilização e ao desenvolvimento de software educativo, o que pode favorecer no enriquecimento dos estudos da área, bem como na ampliação da problemática em que essas questões estão inseridas.

## REFERÊNCIAS

ALMEIDA, R. L. F.; ALMEIDA, C. A. S. **Fundamentos e análise de software educativo**. 2. ed. Fortaleza: Editora da Universidade Estadual do Ceará, 2015.

ALMOULOUD, S. A.; COUTINHO, C. Q. S. Engenharia Didática: características e seus usos em trabalhos apresentados no GT-19/ANPEd. **Revemat: Revista Eletrônica de Educação Matemática**, v. 3, n. 1, 2008. Disponível em: https://periodicos.ufsc.br/index.php/revemat/article/view/1981-1322.2008v3n1p62. Acesso em: 10 jan. 2020.

ALMOULOUD, S. A.; SILVA, M. J. F. . Engenharia didática: evolução e diversidade. **Revemat**: **Revista Eletrônica de Educação Matemática**, v. 7, n. 2, 2012. Disponível em: https://periodicos.ufsc.br/index.php/revemat/article/view/1981-1322.2012v7n2p22. Acesso em: 24 ago. 2020.

AMARAL, E. C.; GUEDES, U. T. V. Análise de construção de software educativo com qualidade: Sugestão de ficha para registro e avaliação de software educativo. In: WORKSHOP DOS CURSOS DE COMPUTAÇÃO APLICADA DO INPE, 5. (WORCAP)., 2005, São José dos Campos. **Anais.** São José dos Campos: INPE, 2005. CD-ROM, On-line. Disponível em: http://urlib.net/rep/dpi.inpe.br/hermes2@1905/2005/10.03.21.08.

ANDRADE, M. E. **Simulação e modelagem computacional com o software Modellus**: aplicações práticas para o ensino de física. São Paulo: Editora Livraria da Física, 2016.

ARTIGUE, M. Engenharia Didática. In: BRUN, J. (Org). **Didáctica das Matemáticas**. 1. ed. Lisboa: Instituto Piaget, 1996. p. 193-217.

ARTIGUE, M. Ingénierie didactique: quel rôle dans la recherche didactique aujourd'hui?. **Les dossiers des sciences de l'éducation**, v. 8, 2002. Disponível em: https://www.persee.fr/doc/dsedu_1296-2104_2002_num_8_1_1010. Acesso em 05 ago. 2019.

ARTIGUE, M. L´ingénierie didactique: un essai de synthèse. In: XVª ÉCOLE D´ÉTE DE DIDACTIQUE DES MATHEMATIQUES, 2009, Clermont-Ferrand (PUY-de-Dôme). **En amont et en aval des ingénieries didactiques**. Grenoble: La Pensée Sauvage, v. 1.

BALACHEFF, N. La transposition informatique. Note sur un nouveau problème pour la didactique. In: ARTIGUE, M. et al. (orgs.). **Vingt ans de didactique des mathématiques en France.** 1 ed**.** Grenoble: La Pensée Sauvage, 1994.

_____. Contribution de la didactique et de l'épistémologie aux recherches em EIAO. In: XIII Journées francophones sur l'informatique, Genève, France. **Actes des 13ème Journées Francophones sur l'Informatique, Formation Intelligemment Assitée par Ordinateur**, Genève,1991.

BELLEMAIN, F., SILVA, A. D. P. R.; TIBURCIO, R. Validação do conhecimento da Engenharia de Software Educativo. In: VII EPEM-ENCONTRO PERNAMBUCANO DE

EDUCAÇÃO MATEMÁTICA, 2017, Garanhuns-PE. **Anais.** Recife: Sociedade Brasileira de Educação Matemática - Diretoria Regional Pernambuco, 2017.

BELLEMAIN, F. A transposição didática na engenharia de softwares educativos. In: I SIPEM - SEMINÁRIO INTERNACIONAL DE PESQUISA EM EDUCAÇÃO MATEMÁTICA, 2000, Serra Negra-SP. **Anais.** São Paulo, 2000.

BELLEMAIN, F. O Paradigma micromundo. In: CARVALHO, L. M.; GUIMARÃES, L. C. (Orgs.). **História e Tecnologia no Ensino de Matemática**. Rio de Janeiro: IME-UERJ, 2002, p. 49-60.

BELLEMAIN. F.; BELLEMAIN, P. M. B.; GITIRANA. V. Elementos de engenharia de software educativos para a concepção de ferramentas computacionais para o CSCL. In ROSA, M.; BAIRRAL, M. A.; AMARAL, R. B. **Educação Matemática, Tecnologias Digitais e Educação a Distância: pesquisas contemporâneas**. Natal-RN: Livraria da Física, 2014.

BENITTI, F. B. V., SEARA, E. F. R., SCHLINDWEIN, L. M. Processo de Desenvolvimento de Software Educacional: Proposta e Experimentação. **Revista Novas Tecnologias na Educação** – **CINTED UFRGS**, v. 3, n. 1, 2005.

BOURQUE, P.; FAIRLEY, R. **Guide to the Software Engineering Body of Knowledge, Version 3.0**. Disponível em: https://www.computer.org/education/bodies-of-knowledge/software-engineering. Acesso em: 10 jan. 2020.

BROUSSEAU, G. **La problématique et l'enseignement des mathématiques**. XXVIIIème rencontre de la CIEAEM, Louvain la Neuve, reproduit dans Recherches en Didactique. 1976.

BROUSSEAU, G. **Problèmes de didactique des décimaux**. Recherches en Didactique des Mathématiques, Vol. 2.3, pp. 37-127. 1981.

CASYOPÉE. **Um environnement d'apprentissage dédié aux fonctions**, 2020. Site do Grupo Casyopée. Disponível em: https://casyopee.math.univ-paris-diderot.fr/index.php?lng=fr. Acesso em: 10, janeiro, 2020.

CHEVALLARD Y. **La transposition didactique**, Ed. La Pensée Sauvage, Grenoble. 1985.

CONFREY, J.; MALONE, A. Research-design interactions in building function probe software. In BLUME, G.; HEID, M. K. (orgs.) **Research on technology and the teaching and learning of mathematics**, vol 2: Cases and perspectives (pp.183-209). 2008. Greenwich.

CONFREY, J.; SMITH, E. Function Probe: Multi-Representational Software for Learning about Functions. In: MALCOM, S.; ROBERTS, L.; SHEINGOLD, K. (orgs.). New York State Association for Computers and Technology in Education Journal. Washington, DC: American Association for the Advancement of Science. 1992.

COSTA, A. P. **Metodologia Híbrida de Desenvolvimento Centrado no Utilizador**. 2012. Tese (Doutorado em Multimidia em Educação) – Departamento de Educação, Universidade de Aveiro, Aveiro.

COSTA, A. P., COSTA, E. B. Contributos para o Desenvolvimento de Software Educativo tendo por base Processos Centrados no Utilizador. **EM TEIA | Revista de Educação Matemática e Tecnológica Iberoamericana**, v. 4, n. 2, p. 1–15, 2013.

GARCIA, V. C.; MACHADO, C. **Teorias de pesquisa em Educação Matemática: a influência dos franceses**. GARCIA V. C. (coord.). Disponível em: http://143.54.226.61/~vclotilde/disciplinas/pesquisa/CLAUDIA_FRANCESES.D%20OC.pdf. Acesso em: 17 set. 2018.

GITIRANA, V., TELES, R. A. M., BELLEMAIN, P. M. B., CASTRO, A.T., ALMEIDA, I.A.C., LIMA, P. F., BELLEMAIN, F. **Jogos com Sucata na Educação Matemática**. 1a. ed. Recife: Editora Universitária da Universidade Federal de Pernambuco, 2013. v. 1.

LAGRANGE, J. -B. Teaching and learning about functions at upper secondary level: designing and experimenting the software environment Casyopée. **International Journal of Mathematical Education in Science and Technology**, 41:2, p. 243-255.

LAGRANGE, J. –B., ARTIGUE, M., CAZES, C., GÉLIS, J. M., VANDEBROUCK, F. **Représenter des Mathématiques avec l'ordinateur. In M. Abboud-Blanchard**, & A. Fluckiger (Dir.), Actes du séminaire national de didactique des mathématiques (pp. 67-100). Paris : IREM de Paris 7. 2011.

LAGRANGE, J.-B. Curriculum, classroom practices and tool design in the learning of functions through technology-aided experimental approaches. **International Journal of Computers in Mathematics Learning**10: 143–189. 2005.

LAGRANGE, J.-B.; GELIS, J.-M. The Casyopée project: A Computer Algebra Systems environment for students' better access to algebra. **International Journal of Continuing Engineering Education and Life-Long Learning**. 186. 10.1504/IJCEELL.2008.022164. 2008.

LAPOLLI, F. R.; MOTTA, C.L.R.; OLIVEIRA, C. E. T.; CRUZ, C. M. Modelo de Desenvolvimento de Objetos de Aprendizagem Baseado em Metodologias Ágeis. In: XX SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 20., 2009, Florianópolis. **Anais...** Porto Alegre: Sociedade Brasileira de Computação, 2009. p. 250-261.

LEMOS, A. Cibercultura. Alguns pontos para compreender a nossa época. In: LEMOS, A., CUNHA, P. **Olhares sobre a cibercultura**. 1. Ed. Porto Alegre: Sulina, 2003.

NUSEIBEH B., EASTERBROOK S. Requirements Engineering: A Roadmap, The Future of Software Engineering. In: 22ND INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ACM-IEEE, 22.,Irlanda. **Anais…** Irlanda: ICSE, 2000.

NUSEIBEH, B.; EASTERBROOK, S. **Requirements Engineering**: A Roadmap. Disponível em: http://mcs.open.ac.uk/ban25/papers/sotar.re.pdf. Acesso em: 05 nov. 2012.

TCHOUNIKINE, P. **« Platon-1 : quelques Dimensions pour l'analyse des travaux de recherche en conception d'EIAH ».** Rapport de l'Action Spécifique « Fondements théoriques et méthodologiques de la conception des EIAH », département STIC du CNRS. 2004.

PAPERT, S., **Mindstorms: children, computers and powerful ideas**. Basic Books, New York. 1980.

PERRIN-GLORIAN, M. J. L'ingénierie didactique a l'interface de la recherche avec l'enseignement. Développement des ressources et formação des enseignants. in Margolinas et all.(org.): En amont et en aval des ingénieries didactiques, **XVª École d´Été de Didactique des Mathématiques** – Clermont-Ferrand (PUY-de-Dôme). Recherches em Didactique des Mathématiques. Grenoble : La Pensée Sauvage, v. 1, p. 57-78, 2009

PRESSMAN, R. S. **Engenharia de Software**. 6. Ed. McGraw-Hill, 2006.

RAMOS. C. S. **Princípios da engenharia de software educativo com base na engenharia didática: uma prototipação do bingo dos racionais.** 2014. f 110. Dissertação de Mestrado (Programa de Pós-Graduação em Educação Matemática e Tecnológica – Edumatec). Recife, UFPE. 2014.

SANTOS, G. L. Alguns princípios para situações de engenharia de software educativos. **Interação**, Goiás, v. 34, n. 1, 2009. Disponível em: http://www.revistas.ufg.br/index.php/interacao/article/view/6540/4801. Acesso em: 06 set. 2016.

SILVA, A. D. P. R. **Prototipação, desenvolvimento e validação de um micromundo com suportes para o ensino de área e perímetro**. 2019. f 388. Tese de Doutorado (Programa de Pós-Graduação em Educação Matemática e Tecnológica – Edumatec). Recife, UFPE. 2019.

SILVA, C. T. J. ; TIBÚRCIO, R S. ; GITIRANA, V. ; BELLEMAIN, F. . Function Studium: um software para o desenvolvimento do raciocínio covariacional. In: II Congresso sobre Tecnologias na Educação - Ctrl+E 2017, 2017, Mamanguape. **Anais**. Mamanguape: Universidade Federal da Paraíba, 2017.

SILVA, C. T. J. **A Engenharia Didático-Informática na prototipação de um software para abordar o conceito de taxa de variação**. 2016. f 163. Dissertação de Mestrado (Programa de Pós-Graduação em Educação Matemática e Tecnológica – Edumatec). Recife, UFPE. 2016.

SILVA, C. T. J. O Uso de Simulações no Ensino de Função: uma sequência didática para abordar taxa de variação. In: XVIII EBRAPEM - ENCONTRO BRASILEIRO DE ESTUDANTES DE PÓS-GRADUAÇÃO EM EDUCAÇÃO MATEMÁTICA, 18., 2014, Recife. **Anais...** Recife: LEMATEC, 2014.

SILVEIRA, J. A. Construcionismo e inovação pedagógica: uma visão crítica das concepções de Papert sobre o uso da tecnologia computacional na aprendizagem da criança. **Revista Themis**. V 10, p. 119 – 138. 2012.

SIQUEIRA, J. E. **Articulando os registros de representação semiótica das Curvas Cônicas através da integração de recursos computacionais**. 2019. f 318. Tese de Doutorado (Programa de Pós-Graduação em Educação Matemática e Tecnológica – Edumatec). Recife, UFPE. 2019.

SOARES, M. S. Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software. INFOCOMP - **Journal of Computer Science**, *3*(2), p. 8-13. 2004.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson Education, 2011.

TAVARES, J. L. **Modelos, técnicas e instrumentos de análise de softwares educacionais** 2017. f 98. Trabalho de Conclusão de Curso. Licenciatura em Pedagogia. Centro de Educação da Universidade Federal da Paraíba. João Pessoa. UFPB. 2017.

TCHOUNIKINE, P. Pour une ingénierie des Environnements Informatiques pour l'Apprentissage Humain. **Revue I3 Information Interaction Intelligence**. v 2, n. 1, p. 59-95. 2002.

TCHOUNIKINE, P. **Précis de recherche de Ingénierie des EIAH**. 2009. Disponível em: https://www.researchgate.net/publication/39064374_Precis_de_recherche_en_Ingenierie_des_EIAH. Acesso em jan de 2020.

TEODORO. V. D. **Modellus: Learning Physics with Mathematical Modelling**. 2002. f 248. Tese de Doutorado. Faculdade de Ciências e Tecnologia. Lisboa. Universidade Nova de Lisboa. 2002.

TIBÚRCIO, R S. Método de Elicitação de Requisitos para Software Educativo: um estudo a partir da prototipação de um software para função em plataformas móveis. In: XVIII EBRAPEM - Encontro Brasileiro de Estudantes de Pós-Graduação em Educação Matemática, 18., 2014, Recife. **Anais...** Recife: LEMATEC, 2014.

TIBÚRCIO, R. S. **Processo de desenvolvimento de software educativo**: um estudo da prototipação de um software para o ensino de função. 2016. f. 112. Dissertação de Mestrado (Programa de Pós-Graduação em Educação Matemática e Tecnológica – Edumatec). Recife. UFPE. 2016.

TIBÚRCIO, R. S.; BELLEMAIN, F.; RAMOS, C. Engenharia de software educativos, o caso do bingo dos racionais. In: VI SEMINÁRIO INTERNACIONAL DE PESQUISA EM EDUCAÇÃO MATEMÁTICA - SIPEM, **ANAIS.** GOIÁS. 2015.

TIBÚRCIO, R. S.; BELLEMAIN, F. Process of educational software development: epistemological and experimental analysis in the creation environment Lematec-Studium. In: Re(s)sources 2018 International Conference, 2018, Lyon. **Proceedings** of the Re(s)sources 2018 International Conference, 2018.

TIBÚRCIO, R. S.; BELLEMAIN, F. . Processo De Desenvolvimento De Software Educativo: Análise Teórica E Experimental Com Recurso A Teoria Da Orquestração Instrumenta. In: I Simpósio Latino-Americano de Didática da Matemática, 2016, Bonito. **ANAIS** - TRABALHOS 1º LADIMA, 2016.

ZAINA, L. M. **A influência da interação humano-computador no desenvolvimento de software**. Cruzeiro do Sul, 2014. Disponível em: https://www2.jornalcruzeiro.com.br/materia/587427/a-influencia-da-interacao-humano-computador-no-desenvolvimento-de-software. Acesso em: 10 de jan de 2020.