Pensamento Computacional e Engenharia de Software: primeiros passos em direção a uma proposta de sistematização de resolução de problemas

Júlia de Avila dos Santos[1], **Simone André da Costa Cavalheiro**[1], **Luciana Foss**[1], **Leomar S. da Rosa Jr.**[1]

[1]Programa de Pós Graduação em Computação
Universidade Federal de Pelotas (UFPEL)

{julia.asantos, simone.costa, lfoss, leomarjr}@inf.ufpel.edu.br

Abstract. *Computational Thinking (CT) involves concepts and techniques from Computer Science that can help in problem solving. Software Engineering (SE) encompasses different actions related to the systematic organization for software development, focusing on the quality and effectiveness of the delivered product. Both CT and SE constitute approaches that involve problem solving grounded in the fundamentals of Computing. This work addresses the first steps towards an interrelationship between the areas that will allow the foundation of future proposals that aim to systematize the problem-solving process.*

Resumo. *O Pensamento Computacional (PC) envolve conceitos e técnicas oriundos da Ciência da Computação que podem auxiliar na resolução de problemas. A Engenharia de Software (ES) engloba diferentes ações relacionadas à organização sistemática para o desenvolvimento de software, com foco na qualidade e eficácia do produto entregue. Tanto o PC quanto a ES constitui abordagens que envolvem a solução de problemas alicerçadas nos fundamentos da Computação. Esse trabalho aborda os primeiros passos em direção a uma inter-relação entre as áreas que permita embasar futuras propostas que visem sistematizar o processo de solução de problemas.*

1. Introdução

O Pensamento Computacional (PC) inclui um conjunto de conceitos e técnicas da Ciência da Computação que dão suporte à resolução de problemas em diversos contextos, tais como abstração, decomposição, pensamento algorítmico, entre outros. A Engenharia de Software (ES) integra diferentes aspectos do processo de desenvolvimento de software, com o objetivo na qualidade e eficácia da solução. A ES abrange diversos processos, métodos e ferramentas que auxiliam na compreensão e no desenvolvimento de um produto final.

PC e ES são áreas que envolvem a solução de problemas alicerçadas nos fundamentos da Computação. Em Santos et al. (2021), é apresentada uma revisão sistemática da literatura investigando de que forma essas duas áreas vem sendo integradas. Como resultado, observa-se que elas são interconectadas de diferentes maneiras, identificando-se trabalhos que: utilizam os fundamentos do PC no processo de desenvolvimento de software; aplicam conceitos do PC no ensino de ES; adotam processos de ES para avaliar o

PC; aplicam técnicas de ES no desenvolvimento do PC. Dos trabalhos que se enquadram nesta última categoria, em Fronza et al. (2016) aborda-se a metodologia do pensamento visual para dar suporte a fase de projeto do processo de resolução de problemas do PC. Já em Fronza et al. (2017), é proposto um *framework* que conta com um conjunto de princípios e práticas, baseado em métodos ágeis de engenharia de software, que podem ser mapeados para o desenvolvimento de projetos de programação (que desenvolvem o PC). Em ambas as propostas o produto final é um software, o que pode restringir a aplicação destas metodologias em diferentes cenários.

No contexto educacional, não há um processo metodológico para formulação e resolução de problemas complexos que seja usualmente aplicado. A proposta de uma abordagem mais abrangente, que integre a sistematização dos processos da ES e os conceitos do PC, pode embasar futuras metodologias que visem sanar este problema. Essa integração se beneficiaria da explicitação do uso dos fundamentos do PC em um processo sistematizado para resolução de problemas. O primeiro passo para se obter esta metodologia consiste em identificar as inter-relações existentes entre as áreas, PC e ES, o qual é o objetivo principal deste trabalho. Para isso, é feito uma releitura das atividades dos processos de ES, generalizando-as no contexto de resolução de problemas e relacionando-as com os conceitos/técnicas do PC.

O restante do artigo está organizado como segue. Na Seção 2 são apresentados os principais conceitos/técnicas do Pensamento Computacional. Uma visão geral da ES é relatada na Seção 3. A Seção 4 descreve os primeiros passos de uma proposta que relaciona os diferentes fundamentos do PC e da ES. Por fim, a conclusão encontra-se na Seção 5.

2. Pensamento Computacional

Papert em 1990 foi o primeiro a abordar o termo e alguns dos conceitos que envolvem o PC. Desde aquela época ele já falava sobre crianças usarem computadores como instrumentos para aprenderem e para aumentar a criatividade, inovação, habilidades e desenvolver o PC [Papert 1990]. Mas só em 2006 que Wing popularizou o termo e chamou a atenção para o tema [Wing 2006]. De acordo com a autora, o PC envolve a resolução de problemas, a concepção de sistemas e a compreensão do comportamento humano, com base nos conceitos fundamentais da Ciência da Computação (CC) [Wing 2006]. Em 2014, Wing [Wing 2014] descreve o PC como o processo de pensamento envolvido na formulação de um problema e na expressão de suas soluções, de tal forma que um computador - humano ou máquina - possa efetivamente ser executado.

O PC inclui uma gama de ferramentas mentais que refletem a amplitude da área da Computação. Desse modo, desenvolve habilidades cognitivas e raciocínio lógico, consequentemente, auxiliando nas resoluções de problemas. Entretanto, o PC não é apenas sobre a resolução de problemas, mas também sobre a formulação de problemas [Wing 2014]. Explorar habilidades do PC faz com que os indivíduos se tornem seres pensantes, agentes e questionadores. Como mencionou Wing(2006), PC é uma habilidade fundamental para todos, não apenas para cientistas da computação.

Para a Sociedade Brasileira de Computação [SBC 2019], o PC desenvolve habilidade de compreender, definir, modelar, comparar, solucionar, automatizar e analisar problemas (e soluções) de forma metódica e sistemática, através da construção de algorit-

mos. Já a *International Society for Technology in Education* (ISTE) e a *Computer Science Teachers Association* (CSTA) colaboraram com líderes do ensino superior, da indústria e do ensino fundamental e médio para desenvolver uma definição de PC. Sendo assim, [CSTA/ISTEA 2011] sugerem que o PC é um processo de resolução de problemas que inclui (mas não se limita) as seguintes características: formular problemas de uma forma que nos permita usar um computador e outras ferramentas para ajudar a resolvê-los; organizar e analisar dados logicamente; representar dados por meio de abstrações, como modelos e simulações; automatizar soluções por meio de pensamento algorítmico (uma série de etapas ordenadas); identificar, analisar e implementar soluções possíveis com o objetivo de alcançar a combinação mais eficiente e eficaz de etapas e recursos; generalizar e transferir este processo de resolução de problemas para uma ampla variedade de problemas. Em outra perspectiva, o PC envolve: definir, entender e resolver problemas; raciocinar em múltiplos níveis de abstração; compreender, aplicar, automatizar e analisar (a adequação de) as abstrações [Lee et al. 2011].

Entretanto, não há um consenso sobre o conceito de PC. No contexto desta pesquisa, compreende-se que o PC envolve conceitos e técnicas oriundos da CC que podem auxiliar na resolução de problemas, propondo o uso de práticas e o desenvolvimento de habilidades cognitivas e comportamentais. Diante disso, discutiremos na sequência as definições dos conceitos/técnicas que serão considerados neste trabalho.

Abstração (ABS): abrange realizar uma ação de separação das características do problema a ser solucionado. É necessário colocar em destaque o que é mais relevante e dispensar detalhes irrelevantes. A abstração é o processo de tornar um artefato mais compreensível através da redução de detalhes desnecessários [Csizmadia et al. 2015].

Automação (AUT): envolve mecanizar a solução (ou partes) permitindo que as máquinas ajudem a solucionar problemas [Ribeiro et al. 2020]. Ou seja, a automação está relacionada em fazer os computadores (referindo-se às nossas máquinas modernas) fazerem o trabalho por nós [Lee et al. 2011].

Avaliação (AVA): é o processo de identificar se existe uma solução viável, que seja adequada ao propósito e esteja correta [Csizmadia et al. 2015, Ribeiro et al. 2020]. Ou seja, engloba analisar a solução do problema e observar a eficiência, investigar se foram escolhidos os melhores caminhos. Para avaliar a solução pode-se testar e verificar propriedades para encontrar possíveis erros, aferindo a eficácia de cada etapa.

Dados (DAD): esse conceito envolve (1) a coleta de informações pertinentes ao problema a ser solucionado; (2) a representação das informações de forma organizada (por exemplo, por matrizes, gráficos, entre outros) para uso e/ou análise; (3) a análise dos dados obtidos a partir das informações coletadas, dando sentido a esses dados, encontrando padrões e tirando conclusões [CSTA/ISTEA 2011].

Decomposição (DEC): está relacionado em analisar o problema e dividi-lo em subproblemas, os quais são resolvidos independentemente, e cujas soluções são combinadas para construir a solução do problema original. Deste modo, permite uma melhor organização e visualização do problema e de sua solução; facilita o trabalho em grupo; e permite que seja possível reutilizar as soluções dos subproblemas em outros problemas.

Generalização/Reconhecimento de Padrões (GEN): é uma técnica que consiste em

construir uma solução (algoritmo) mais genérica a partir de outra, permitindo que o novo algoritmo seja utilizado em outros contextos [Ribeiro et al. 2020]. Está relacionada a identificar semelhanças nas etapas da solução do problema, permitindo a classificação destes em categorias para que seja possível identificar problemas parecidos que já foram solucionados anteriormente [Brennan and Resnick 2012] e possibilitar o reuso das soluções.

Paralelismo (PAR): a técnica na qual várias tarefas são planejadas para serem executadas ao mesmo tempo para alcançar um objetivo, a solução do problema ou parte dele. Sendo assim, envolve a ocorrência de múltiplas sequências [Brennan and Resnick 2012], onde mais de uma instrução é executada simultaneamente.

Pensamento Algorítmico (PAL): envolve a elaboração e execução de etapas para solucionar um problema ou parte dele. Essas etapas devem ser objetivas, claras e não ambíguas. A definição de algoritmos consiste na projeção de instruções lógicas e ordenadas [Shute et al. 2017]. O pensamento algorítmico precisa entrar em ação quando problemas semelhantes precisam ser resolvidos repetidamente. Eles não precisam ser pensados de novo a cada vez, é necessária uma solução que funcione sempre [Csizmadia et al. 2015].

3. Engenharia de Software

O termo Engenharia de Software foi proposto em 1969, na conferência da Organização do Tratado do Atlântico Norte (OTAN), devido a discussão de problemas relacionados com desenvolvimento de software: grandes softwares atrasavam, não entregavam as funcionalidades necessárias, custavam mais do que o esperado e não eram confiáveis [Sommerville 2011]. Desta maneira, a ES foi desenvolvida para que se tenha êxito no resultado final da criação de um software. Quer-se que o software desenvolvido seja eficiente e eficaz no que ele se propõe. Além disso, o software deve ser de fácil utilização e estar apto a modificações a qualquer momento, durando um longo período de tempo.

Como os softwares estão sempre evoluindo (tamanho e complexidade), a ES deve estar em constante adaptação de seus processos de desenvolvimento de software para atender todos os tipos de demandas, e continuar preservando a qualidade e eficácia do produto. Por isso, a área trabalha com diferentes aspectos do processo de desenvolvimento de software, baseado em um sistema organizacional e focado em uma solução adequada e de qualidade. Sendo assim, abrange processos, um conjunto de métodos (práticas) e um leque de ferramentas que possibilitam aos profissionais desenvolverem software de altíssima qualidade [Pressman and Maxim 2016].

Pressman and Maxim (2016) afirmam que a ES é uma tecnologia em camadas, cuja base é a camada de **processos**, que define uma metodologia que deve ser estabelecida para permitir a entrega efetiva do software, desenvolvido de forma racional e dentro do prazo. O processo de software constitui a base para o controle do gerenciamento de projetos. A camada de **métodos** fornece as informações técnicas para desenvolver o software. Os métodos envolvem uma ampla variedade de tarefas, que incluem: comunicação, análise de requisitos, modelagem de projeto, construção de programa, testes e suporte. Os métodos da engenharia de software se baseiam em um conjunto de princípios básicos que governam cada área da tecnologia e incluem atividades de modelagem e técnicas descritivas. Já a camada de **ferramentas** fornece suporte automatizado ou semi automatizado

para o processo e para os métodos.

Um modelo de processo é uma representação abstrata de um processo de software. Os processos podem seguir diferentes modelos que evoluem com o tempo. Os modelos de processos foram propostos para trazer uma organização sistemática em atividades para o desenvolvimento de software. Desta maneira, um modelo de processo apresenta um guia para o desenvolvimento de software. Ele define o fluxo de todas as atividades, ações e tarefas, o grau de iteração, os artefatos e a organização do trabalho a ser feito [Pressman and Maxim 2016]. Na literatura existem diversos modelos de processos, e não existe um ideal, muitas vezes as empresas criam seus próprios processos, mesclando diferentes modelos que se encaixam para sua forma de negócio.

3.1. Atividades de Processos

Sommerville (2011) apresenta que o processo de software inclui todas as atividades envolvidas no desenvolvimento do software. Em um processo, existem muitas atividades diferentes e a nomenclatura varia na literatura. Essas atividades são consideradas ações metodológicas e de apoio, tarefas a serem realizadas no processo para garantir a qualidade.

Para Pressman and Maxim (2016) o processo de software incorpora cinco atividades estruturais: (1) comunicação: relacionada ao entendimento do problema e ao levantamento dos requisitos; (2) planejamento: essa atividade observa as tarefas a serem feitas para organizar estimativas de entrega, um cronograma de todo o projeto, além de definições de acompanhamento do projeto pela equipe; (3) modelagem: nesta atividade, criam-se modelos que permitam compreender o software como um todo, qual será a sua arquitetura em termos das partes constituintes, como elas se conectam, e várias outras características. Os modelos podem ser refinados para a adição de mais detalhes, permitindo entender melhor as necessidades do software e o projeto que vai atender a essas necessidades; (4) construção: essa atividade está relacionada com a geração de código do software, testes e validações da solução; e (5) entrega: nessa atividade ocorre a entrega final do produto, sendo necessário ainda observar feedbacks e realizar suporte.

Sommerville (2011) define quatro atividades fundamentais, entretanto é possível observar que muitas ações dessas atividades estão relacionadas com as abordadas por Pressman and Maxim (2016). Elas são: (1) especificação de software: atividade na qual a funcionalidade do software e as restrições de seu funcionamento devem ser definidas; (2) projeto e implementação de software: etapa na qual o software deve ser produzido para atender às especificações; (3) validação de software: atividade que deve garantir que o software atenda as demandas do cliente; e (4) evolução de software: ação que garante que o software possa evoluir para atender às necessidades de mudanças dos clientes. Contudo, é necessário compreender que essas quatro atividades também incluem subatividades, como validação de requisitos, projeto de arquitetura, testes unitários etc. Existem também as atividades que dão apoio ao processo, como documentação e gerenciamento de configuração de software.

4. Integração entre ES e PC

Esta seção propõe os primeiros passos para a elaboração de uma metodologia para sistematização da resolução de problemas genéricos com base nos processos de desenvolvimento de software integrados ao PC. A Subseção 4.1 descreve as atividades de processos

que serão consideradas para a elaboração desta metodologia. A Subseção 4.2 apresenta algumas formas de integrar os conceitos/técnicas do PC nas atividades relacionadas.

4.1. Atividades de Processos Adaptadas para Resolução de Problemas

Considerando as atividades envolvidas nos processos de desenvolvimento de software e seus desdobramentos, nesta seção são apresentadas as atividades que devem fazer parte da metodologia a ser proposta. Foi realizada a releitura das atividades de processos, considerando as diferentes etapas mencionadas pelos autores citados [Pressman and Maxim 2016, Sommerville 2011]. O procedimento seguido para realização desta releitura foi efetuado como segue:

1. As atividades de processos especificadas por ambos os autores foram comparadas, identificando-se as semelhanças e diferenças;
2. Os detalhes e subdivisões de cada atividade foram analisados para fundamentar as atividades incluídas nesta proposta.
3. As atividades consideradas foram elencadas e generalizadas para o contexto de solução de problemas de um modo geral e não só para o desenvolvimento de software.

Como resultado deste procedimento obteve-se as seguintes atividades:

- **Formulação e descrição do problema**, identificando os requisitos necessários para sua solução, possíveis restrições e o objetivo a ser alcançado.
- **Levantamento e derivação de requisitos** obtidos a partir da interação com atores relacionados com o problema.
- **Análise de viabilidade**, incluindo o levantamento de custos (tempo, financeiro, etc.) e verificando se os recursos disponíveis são suficientes para se alcançar o objetivo estabelecido.
- **Validação dos requisitos** identificados, observando sua consistência e abrangência, realizando ajustes quando necessário.
- **Estruturação da resolução do problema**, identificando os subproblemas e os seus relacionamentos.
- **Planejamento de execução**, elaborando o cronograma de resolução dos subproblemas, observando as dependências entre eles e registrando os possíveis riscos envolvidos, bem como alocando a equipe de execução para cada etapa descriminada no cronograma;
- **Descrição da resolução de cada subproblema**, identificando objetivos, restrições e possibilidades de reúso.
- **Definição das interfaces** da resolução de cada subproblema, identificando os recursos necessários e resultados esperados de forma precisa para que os demais componentes que o utilizem não precisem conhecer os detalhes de sua resolução.
- **Resolução dos subproblemas** (criação, reúso ou adaptação), descrevendo de forma detalhada e não ambígua as etapas envolvidas na sua resolução.
- **Integração das soluções** dos subproblemas, fazendo uso das interfaces estabelecidas e obtendo a resolução completa do problema inicial.
- **Verificação da resolução de cada subproblema**, de forma independente, analisando a sua descrição detalhada para identificar possíveis falhas com relação aos objetivos e restrições pré-estabelecidos, corrigindo-as quando necessário.

- **Verificação da resolução completa do problema**, analisando a descrição integrada das resoluções dos seus subproblemas para identificar possíveis falhas com relação ao objetivo e restrições pré-estabelecidas e às interfaces dos subproblemas, corrigindo-as quando necessário.
- **Execução da resolução do problema**, envolvendo os atores relacionados, para identificar a aceitação da solução com relação às expectativas dos atores, readequando-a quando necessário.
- **Proposição de extensões/adaptações** à resolução do problema para atender a novos objetivos, que possam surgir após a interação com os atores.

A Tabela 1 mostra as correspondências das atividades destacadas pela principais referências ([Sommerville 2011, Pressman and Maxim 2016]) e as adaptações propostas neste trabalho. Cabe observar que para relacionar as atividades adaptadas, as atividades de planejamento e modelagem do Pressman and Maxim (2016) foram agrupadas. Além disso, foram considerados e integrados os desdobramentos detalhados pelos autores para definir as atividades desta proposta.

Tabela 1. Atividades de processos adaptadas e as correspondências com as atividades de processos de software.

Sommerville	**Atividades Adaptadas**	**Pressman**
Especificação de software	Formulação e descrição do problema	Comunicação
	Levantamento e derivação de requisitos	
	Análise de viabilidade	
	Validação dos requisitos	
Projeto e implementação de software	Estruturação da resolução do problema	Planejamento e modelagem
	Planejamento de execução	
	Descrição das resoluções dos subproblemas	
	Definição das interfaces	
	Resolução dos subproblemas	Construção
	Integração das soluções	
Validação de software	Verificação da resolução dos subproblemas	
	Verificação da resolução do problema	
	Execução da resolução do problema	
Evolução de software	Proposição de extensões/adaptações	Entrega

4.2. Integração do PC às Atividades de Processos

Nesta seção é apresentada uma discussão de quais conceitos/técnicas do PC podem ser considerados em cada uma das atividades, bem como de que forma eles estão relacionados. Esta proposta de integração não é exaustiva, sendo dada ênfase às principais relações que podem ser consideradas em cada atividade. Os conceitos/técnicas não considerados em alguma atividade, podem ainda estar associados de forma menos explícita.

Formulação e descrição do problema: a formulação envolve a descrição *abstrata* do problema, identificando informações e recursos necessários para a sua resolução e resultados esperados. Nesse momento, não é realizada uma descrição de como, mas sim o que deve ser feito para se obter a solução do problema. Nesta atividade, deve-se identificar explicitamente os *dados* envolvidos na resolução do problema, podendo ser agrupados por tipos abstratos (como conjuntos numéricos, equipamentos, textos, etc.), quando for o caso.

Levantamento e derivação de requisitos: diversas características e restrições relacionadas ao problema são identificadas durante o levantamento de requisitos (coleta de *dados*). A partir disso, os requisitos serão derivados e a técnica de *abstração* pode ser aplicada para selecionar as características e restrições relevantes, descartando aquelas irrelevantes. Os requisitos devem ser suficientemente detalhados de forma que facilite a descrição do problema, evitando excesso de detalhes que possa complexificar essa descrição. Conforme a necessidade, os requisitos podem ser *decompostos* baixando o nível de abstração (e aumentando o nível de detalhamento). O processo de organização de derivação e gerenciamentos dos requisitos pode ser *automatizada* por meio de ferramentas específicas. No caso em que se identifique diferentes fontes para realização do levantamento e derivação de requisitos, estes podem ser agrupados em categorias, que podem ser tratadas de forma *paralela* por diferentes equipes.

Análise de viabilidade: ao se fazer o levantamento dos custos e recursos disponíveis, estas informações (*dados*) devem ser organizadas de modo a facilitar a *avaliação* da viabilidade de se resolver o problema. O levantamento e análise pode ser *decomposto* em diferentes aspectos (tempo, custo, equipamentos, etc.), permitindo a *paralelização* de suas execuções.

Validação dos requisitos: a *avaliação* da correção do levantamento dos requisitos é realizada verificando se não há contradições entre os requisitos estabelecidos e se abrangem todos os aspectos desejados.

Estruturação da resolução do problema: para se definir a estrutura da resolução do problema, deve-se *decompô-lo* em subproblemas, identificando os componentes que resolvem cada um dos subproblemas. A forma em que esses componentes se relacionam deve ser estabelecida com base na dependência que cada componente tem dos demais, o que determinará a maneira em que as suas soluções serão *compostas*. Podem ser identificados subproblemas que seguem um mesmo *padrão* para que se possa *generalizar* o componente que os resolve.

Planejamento de execução: a definição do cronograma envolve a identificação da ordem na qual as ações relacionadas as demais atividades serão realizadas e a alocação da equipe para realizar as diferentes ações. Neste planejamento devem ser identificadas ações que podem ser realizadas em *paralelo*, definindo fluxos independentes de execução, destacando pontos de sincronização. O *pensamento algorítmico* deve ser aplicado na especificação desta ordem, levando em conta as dependências entre os subproblemas e as demais atividades.

Descrição das resoluções dos subproblemas: uma descrição *abstrata* (sem detalhamento) da resolução de cada subproblema indicando o seu objetivo e as restrições envolvidas deve ser apresentada. Ainda, podem-se identificar *padrões* entre os objetivos de diferentes subproblemas para *generalizar* as resoluções e permitir o reúso.

Definição das interfaces: para cada subproblema, é necessário representar com precisão as informações/recursos (*dados*) necessários para a resolução de cada subproblema, bem como os resultados esperados, levando em conta os relacionamentos definidos entre os subproblemas.

Resolução dos subproblemas: utiliza-se o *pensamento algorítmico* para a elaboração e a descrição detalhada das etapas da resolução de cada subproblema, indicando explicitamente o uso dos recursos para se alcançar o objetivo. Em caso de reúso, quando adaptações forem necessárias, a resolução deve ser descrita com as respectivas modificações. Para as resoluções que devem solucionar mais de um subproblema, a descrição das etapas precisa ser *genérica* para que se aplique a todos os casos identificados. Pode-se fazer uso de dispositivos computacionais para *automatizar* algumas das resoluções, implementando a descrição detalhada correspondente.

Integração das soluções: as resoluções dos subproblemas devem ser *compostas* de forma lógica, respeitando as interfaces e a estruturação do problema, considerando as *paralelizações* previamente identificadas. Essa composição deve ser descrita de modo a se obter um *algoritmo* que constitua a solução detalhada do problema original. Se for o caso, a resolução pode ser *automatizada* através da implementação do algoritmo descrito.

Verificação da resolução do(s) (sub)problema(s): a resolução do(s) (sub)problema(s) pode(m) ser simulada(s), considerando quando for o caso diferentes possibilidades de recursos de entrada, para *avaliar* a sua correção. Isto é, verificar se, seguindo os passos da resolução com os recursos dados, é possível chegar no objetivo estabelecido. A *abstração* pode ser utilizada para selecionar características relevantes permitindo categorizar os recursos de entrada (*dados*) em classes. Um elemento de cada classe pode ser selecionado para realização das simulações, simplificando o processo de verificação.

Execução da resolução do problema: os passos descritos na resolução do problema são executados em um ambiente real (*pensamento algorítmico*) para sua validação.

Proposição de extensões/adaptações: a execução da resolução pode fazer emergir novos objetivos que devem ser descritos de forma *abstrata* para que extensões e adaptações sejam adicionadas a descrição da solução do problema.

A Tabela 2 sumariza as relações estabelecidas entre as atividades de processos adaptadas e o PC. Pode-se observar que a maioria das atividades envolve pelo menos dois conceitos/técnicas do PC. Por sua vez, cada conceito/técnica do PC se enquadra em pelo menos três atividades. Isso atesta uma relação sólida entre as duas áreas. Os conceitos/técnicas mais frequentes são abstração e dados. Pode-se dizer que a abstração é um dos conceitos mais abrangentes do PC, uma vez que está relacionado tanto com a representação de informações quanto com os processos que lidam com as mesmas. Embora nem sempre que uma atividade esteja associada a dados e pensamento algorítmico, ela também esteja associada a abstração, essa relação ocorre de maneira implícita. Isso porque toda descrição algorítmica faz usa de abstrações para processos. Esse tipo de relação não foi explicitamente estabelecido.

Outras relações com os conceitos/técnicas do PC podem surgir quando forem considerados os modelos de processos. O modelo de processo definirá o fluxo de execução das atividades propostas. Um ou mais modelos de processos podem ser ado-

Tabela 2. Relações entre as atividades de processos adaptadas e os conceitos/técnicas do PC.

	ABS	AUT	AVA	DAD	DEC	GEN	PAR	PAL
Formulação e descrição do problema	X			X				
Levantamento e derivação de requisitos	X	X		X	X		X	
Análise de viabilidade			X	X	X		X	
Validação dos requisitos			X					
Estruturação da resolução do problema					X	X		
Planejamento de execução							X	X
Descrição das resoluções dos subproblemas	X					X		
Definição das interfaces				X				
Resolução dos subproblemas		X				X		X
Integração das soluções		X			X		X	X
Verificação da resolução dos subproblemas	X		X	X				
Verificação da resolução do problema	X		X	X				
Execução da resolução do problema								X
Proposição de extensões/adaptações	X							

tados/adaptados e, dependendo da(s) escolha(s), outras atividades podem ser incluídas/adaptadas.

5. Conclusão

Este trabalho teve como objetivo realizar uma primeira discussão das relações entre os conceitos/técnicas do PC com as atividades de processos do desenvolvimento de soft-

ware da ES. Apesar de alguns autores mencionarem o PC como uma metodologia, de fato não há uma sistematização do processo de resolução de problemas. O PC envolve conceitos/técnicas da CC que podem auxiliar na resolução de problemas, além de desenvolver habilidades cognitivas, técnicas e comportamentais. Por sua vez, a ES engloba diversos processos, métodos e ferramentas para desenvolver um software, ou seja, resolver um problema automatizando a solução. A integração entre essas áreas pode favorecer o estabelecimento de uma metodologia para o PC que se aplique a diferentes contextos, incluindo até mesmo o processo de desenvolvimento de software. Com as relações estabelecidas, pode-se perceber que as atividades de processos adaptadas são como etapas a serem percorridas num método de resolução de problemas e os conceitos/técnicas do PC embasam as ações que devem ser observadas e trabalhadas em cada atividade.

A inter-relação proposta permite entender melhor como profissionais de software desenvolvem habilidades de pensamento e resolução de problemas. Fazendo uso de algumas das atividades de processos adaptadas definidas seria possível identificar no processo de resolução de problemas em qualquer contexto educacional quais são as habilidades do PC que estão envolvidas nesse processo.

Este trabalho apresenta os primeiros passos que fundamentarão a elaboração de uma metodologia para o PC com base nos processos da ES. As próximas etapas envolvem a escolha dos modelos de processos a serem adotados definindo o fluxo de execução das atividades, a identificação de novas relações e conceitos a serem incorporados. Além disso, pretende-se identificar, adaptar ou criar ferramentas que deem suporte a concretização desta proposta.

Referências

Brennan, K. and Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada*, volume 1, page 25.

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., and Woollard, J. (2015). Computational thinking - A guide for teachers. Guide, Computing at School. `https://eprints.soton.ac.uk/424545/1/150818_Computational_Thinking_1_.pdf`.

CSTA/ISTEA (2011). Computational Thinking in K–12 Education: leadership toolkit. `https://cdn.iste.org/www-root/2020-10/ISTE_CT_Leadership_Toolkit_booklet.pdf`.

Fronza, I., El Ioini, N., and Corral, L. (2016). Teaching software design engineering across the k-12 curriculum: Using visual thinking and computational thinking. In *Proceedings of the 17th Annual Conference on Information Technology Education*, SIGITE '16, page 97–101, New York, NY, USA. Association for Computing Machinery.

Fronza, I., Ioini, N. E., and Corral, L. (2017). Teaching computational thinking using agile software engineering methods: A framework for middle schools. *ACM Trans. Comput. Educ.*, 17(4).

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., and Werner, L. (2011). Computational thinking for youth in practice. *Acm Inroads*, 2(1):32–37.

Papert, S. (1990). Children, computers and powerful ideas.

Pressman, R. and Maxim, B. (2016). *Engenharia de Software-8ª Edição*. McGraw Hill Brasil.

Ribeiro, L., Foss, L., and da Costa Cavalheiro, S. A. (2020). Entendendo o pensamento computacional. In Raabe, A., Zorzo, A. F., and Blikstein, P., editors, *Computação na educação básica: fundamentos e experiências*. Penso Editora.

Santos, J., Cavalheiro, S., Foss, L., and Jr., L. R. (2021). Relações entre o pensamento computacional e a engenharia de software: Uma revisão sistemática da literatura. In *Anais do XXXII Simpósio Brasileiro de Informática na Educação*, pages 1027–1038, Porto Alegre, RS, Brasil. SBC.

SBC (2019). Diretrizes para ensino de computação na educação básica. `https://www.sbc.org.br/educacao/diretrizes-para-ensino-de-computacao-na-educacao-basica`.

Shute, V. J., Sun, C., and Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, 22:142–158.

Sommerville, I. (2011). Engenharia de software. *9a. edição, Pearson Prentice Hall*.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3):33–35.

Wing, J. M. (2014). Computational thinking benefits society. *40th Anniversary Blog of Social Issues in Computing*, 2014:26.