Metodologia da Decomposição e Composição para Resolução de Problemas em LiVE*

Adriana Bordini[1], **Simone André da Costa Cavalheiro**[1], **Luciana Foss**[1]

[1]Programa de Pós-Graduação em Computação – Universidade Federal de Pelotas (UFPel)
96.010-610 – Pelotas – RS – Brasil

{adriana.bordini, simone.costa, lfoss}@inf.ufpel.edu.br

Abstract. *The goal of this work is to propose a decomposition and composition methodology for problem solving, based on Computational Thinking techniques, allowing these strategies to be applied at various levels of education and in different areas of knowledge. The methodology was proposed for the visual language LiVE. The work presents a case study, which was carried out to evaluate the proposed methodology. The study showed that most students reached a solution with an appropriated level of detail and accuracy.*

Resumo. *O objetivo deste trabalho é propor uma metodologia de decomposição e composição para resolução de problemas, fundamentada nas técnicas do Pensamento Computacional, permitindo que essas estratégias possam ser aplicadas em vários níveis de ensino e em diversas áreas do conhecimento. A metodologia foi proposta para a linguagem visual LiVE. O trabalho apresenta um estudo de caso, que foi realizado para avaliar a metodologia proposta. O estudo mostrou que a maioria dos alunos chegou a uma solução com um nível adequado de detalhamento e exatidão.*

1. Introdução

Com os avanços das tecnologias e a sua inserção em nossa vida diária, a forma de pensar e resolver problemas vem se modificando, com isso novas habilidades devem ser trabalhadas na educação. Habilidades estas que podem ser desenvolvidas por meio do Pensamento Computacional (PC) [Tabesh 2017, Yadav et al. 2016, Mohaghegh and McCauley 2016]. O PC é definido como um processo de resolução de problemas que inclui (mas não está limitado a) habilidades, dimensões e conceitos da ciência da computação [ISTE and CSTA 2011].

Uma série de trabalhos abordam a resolução de problemas por meio do PC, estando geralmente relacionados à programação, simulação, jogos e/ou robótica, muitas vezes em ambientes de programação visual [Souza 2019, Cunha and Nascimento 2018, Silva and Javaroni 2018, Pellas and Vosinakis 2018, Rezende and Bispo 2018, Maquil et al. 2018, Malizia et al. 2017, Oliveira et al. 2016, Martins et al. 2016]. Alguns autores [Gardeli and Vosinakis 2017, Koh et al. 2010, Howland et al. 2009] questionam, tanto o uso da programação quanto os próprios ambientes visuais de programação no

*O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. O projeto também conta com o apoio da PREC e PRPPG / UFPel.

desenvolvimento de habilidades do PC. São questionados quais conceitos do PC são realmente aprendidos e até que ponto a aquisição de habilidades do PC tem sido um efeito colateral de aprender a programar. Afirmam ainda que: o PC é mais amplo do que programação e deve ser trabalhado em um nível mais alto de abstração [Howland et al. 2009]; a utilização de um ambiente computacional fácil de usar, por si só, não é suficiente para a aquisição efetiva de habilidades do PC [Gardeli and Vosinakis 2017]; e as linguagens de programação visual existentes, por serem mais lúdicas e intuitivas, acabam por "mascarar" os conceitos de computação que estão sendo trabalhados e, consequentemente, impedem o desenvolvimento de habilidades do PC [Howland et al. 2009].

Neste trabalho é proposta uma metodologia para resolução de problemas, fundamentada nas técnicas de decomposição e composição do PC. A metodologia é proposta para a linguagem de especificação visual LiVE, a qual permite usar de forma explícita diferentes técnicas da computação para resolver problemas, sem a necessidade de preocupar-se com detalhes específicos de implementação. A metodologia visa sistematizar a aplicação da técnica de decomposição e composição na resolução de problemas. Com essa sistematização pretende-se facilitar a disseminação do pensar computacionalmente, como propõe Wing [Wing 2006]. Além disso, um estudo de caso foi desenvolvido para avaliar o uso metodologia proposta.

O trabalho está organizado como segue. A Seção 2 apresenta a linguagem LiVE e a Seção 3 descreve a metodologia proposta, detalhando um exemplo de aplicação. Na Seção 4 relata-se o estudo de caso. E por fim, as considerações finais são delineadas na seção 5.

2. Linguagem de Especificação Visual LiVE

A linguagem visual LiVE foi projetada não para programar, mas sim para especificar uma solução em um nível de abstração mais alto. LiVE tem uma descrição para ações e para o fluxo de dados, o qual representa as dependências entre estas ações. Ela utiliza o conceito de parâmetros, representado por portas. As portas são similares a endereços de memória que não precisam ser mantidos de forma direta, pois são tratadas apenas como um meio de passar a informação. A linguagem acaba mantendo o essencial para representar uma sequência de passos e as relações entre estas dependências, descrevendo de forma abstrata a solução do problema. Ela também é simples o suficiente para representar a solução em si, sem sobrecarregar com outros detalhes que dizem respeito só a implementação.

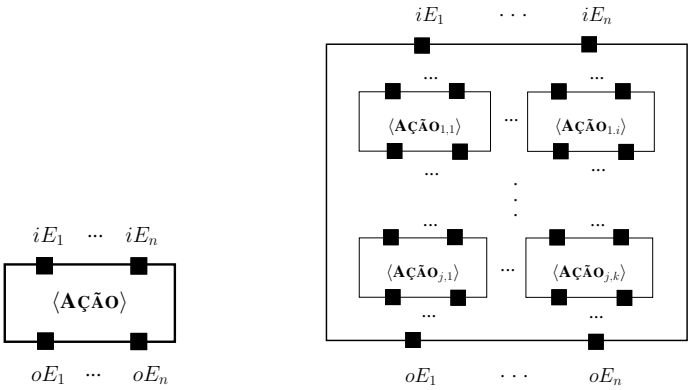
2.1. Sintaxe

Uma especificação na linguagem LiVE descreve uma solução de um problema. Ela inclui um conjunto de componentes elementares que descrevem de forma abstrata todas as ações que podem ser realizadas. Um destes componentes é o principal, o qual descreve a ação que soluciona o problema como um todo.

Um **Componente Elementar** (CE), ilustrado na Figura 1(a), descreve uma ação a ser executada sem detalhá-la, apenas especificando os tipos das informações de entrada necessárias para sua execução, bem como os tipos dos resultados obtidos. Um CE (representado por um retângulo) é definido por um nome, descrevendo a ação, uma lista de portas de entrada e uma lista de portas de saída (quadrados pretos na borda superior e inferior dos componentes, respectivamente). As portas de entrada/saída são os meios pelos

quais as informações/resultados são recebidos/enviados pelo CE. Por sua vez, cada porta possui um tipo que restringe o tipo de elementos que podem ser associados a ela.

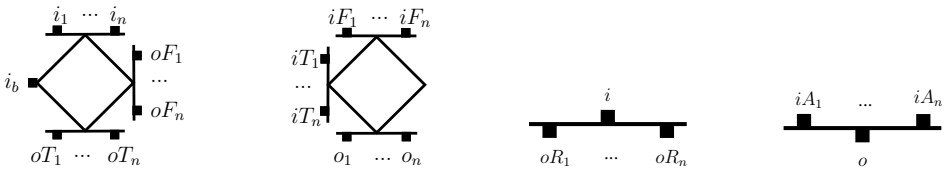
Caso sejam necessários mais detalhes, um componente elementar pode ser associado a um componente decomposto o qual define tais detalhes. O **Componente Decomposto** (CD), esquematizado na Figura 1(b), consiste de um conjunto de instâncias de CE interligados por conexões. Além disso, pode-se fazer uso de outros componentes como nós condicionais, nós de refinamento e nós de abstração. Esses componentes também estão interligados entre si e/ou com as instâncias de CE por conexões, as quais conectam as portas dos componentes, estabelecendo caminhos por onde os dados circulam. Cabe salientar que quando se referenciam as portas de entrada/saída de um CD, na verdade se estão referenciando as respectivas portas do CE associado.



(a) Elementar (b) Decomposto

Figura 1. Componentes para descrever ações.

Um **nó condicional** descreve uma escolha entre dois possíveis caminhos, pelos quais os dados podem seguir. Estes caminhos são delimitados por um **nó de decisão** (no início) e um **nó de junção** (no final), ilustrados nas Figuras 2(a) e 2(b), respectivamente. Ao chegar em um nó de decisão (através das portas $\langle i_1, \ldots, i_n \rangle$), os dados são enviados por um dos caminhos (através das portas $\langle oT_1, \ldots, oT_n \rangle$ ou $\langle oF_1, \ldots, oF_n \rangle$), dependendo de um valor booleano ($true$ ou $false$, respectivamente), recebido através da porta i_b. Os dados chegam em um nó de junção por um dos dois caminhos (através das portas $\langle iT_1, \ldots, iT_n \rangle$ ou $\langle iF_1, \ldots, iF_n \rangle$) e seguirão o seu fluxo (através das portas $\langle o_1, \ldots, o_n \rangle$). Deve existir uma equivalência de tipos entre as portas de entrada e portas de saída, tanto para os nós de decisão quanto para os de junção.



(a) Decisão (b) Junção (c) Refinamento (d) Abstração

Figura 2. Componentes para descrever fluxo e detalhamento dos dados.

As informações de entrada ou saída de um CD também podem ser mais detalhadas ou abstraídas, respectivamente. Tais detalhes são descritos por **nós de refinamento e de**

abstração (representadas por barras horizontais, conforme as Figuras 2(c) e 2(d)). Um nó de refinamento permite que um tipo de dado mais abstrato (i) seja detalhado em uma lista de tipos mais concretos ($\langle oR_1 \cdots oR_n \rangle$). Já um nó de abstração permite que uma lista de tipos mais concretos ($\langle iA_1 \cdots iA_n \rangle$) sejam abstraídos num tipo mais abstrato (o).

3. Metodologia da decomposição e composição para a linguagem LiVE

A **decomposição** refere-se à solução de um problema a partir da sua divisão em subproblemas mais simples, cujas soluções são combinadas para resolver o problema maior. Já a **composição** estabelece como as soluções dos subproblemas devem ser combinadas para resolver o problema original. A **Metodologia da Decomposição e Composição** sistematiza a solução de um problema pela combinação destas duas técnicas. Nesta metodologia, quer-se descrever a solução de um problema a partir da divisão do problema original em subproblemas, da definição de CEs que solucionam cada subproblema e da composição destes CEs (ações) que definem um CD, o qual resolve o problema original.

A solução do problema é dada por uma especificação, obtida por meio da construção de um CD a partir de instâncias de CE e conexões entre esses componentes. A solução deve seguir as seguintes etapas (sintetizadas na Figura 3):

- D1 Dividir o problema maior em subproblemas menores (mais simples) onde as saídas de cada subproblema dependem exclusivamente de suas entradas, cujas combinações das soluções permitam resolver o problema original. É desejável que para quaisquer dois subproblemas menores, nenhum seja subproblema do outro. Represente a divisão estabelecida como uma árvore: onde a raiz é o problema e os filhos são os subproblemas identificados.
- D2 Dividir os subproblemas até que eles tenham soluções triviais. Para cada subproblema a ser dividido, selecione na árvore o nó que representa esse subproblema e adicione um filho para cada uma de suas divisões.
- D3 Definir um CE para cada subproblema trivial (i.e., subproblemas relacionados às folhas da árvore) distinto identificado nas etapas anteriores (D1 ou D2), estabelecendo suas entradas e saídas.
- C1 Adicionar uma instância de CE para resolver cada um dos subproblemas triviais e marcá-los como resolvidos.
- C2 Identificar os menores subproblemas (i.e., os de nível mais baixo na árvore) que ainda não foram resolvidos. Selecionar um subproblema e identificar as ações necessárias para resolvê-lo combinando soluções anteriores. Definir um CE para cada ação de combinação (i.e., ação (de convergência) que liga outras ações para a resolução de um subproblema) distinta. Adicionar no CD em construção as instâncias necessárias das ações de combinação e estabelecer as conexões necessárias para resolver o subproblema selecionado. Cada entrada de uma ação de combinação deve ser conectada a uma saída de alguma ação das ações combinadas. Repetir a etapa C2 até que o problema principal seja resolvido.
- C3 Definir o CE cuja decomposição foi estabelecida nas etapas anteriores. Criar para o CE: uma porta de entrada para cada entrada independente do CD (portas de entrada das instâncias de CE que compõem o CD e que não estão conectadas a outras portas), estabelecendo as conexões entre tais entradas; e, uma porta de saída para cada saída independente do CD (portas de saída das instâncias de CE que compõem o CD e que não estão conectadas a outras portas), estabelecendo as

conexões com tais saídas. Além disso, um nome deve ser escolhido para este CE que será o componente principal da especificação.

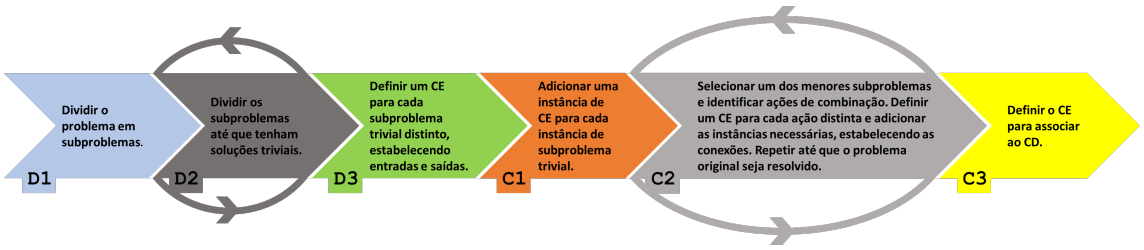


Figura 3. Síntese da Metodologia da Decomposição e Composição.

3.1. Exemplo de aplicação

Para exemplificar a aplicação da Metodologia da Decomposição e Composição, descreve-se a solução do problema de desenhar a imagem ilustrada na Figura 4.

Figura 4. Imagem utilizada no problema apresentado no estudo de caso.

A imagem pode ser dividida em rosto (círculo amarelo), olhos (dois círculos pretos posicionados lado a lado), boca (um retângulo amarelo sobreposto a um círculo vermelho) e um laço (dois triângulos lado a lado). Para desenhar essa imagem, foram consideradas as seguintes ações triviais: desenhar **retângulo** de determinada cor e tamanho; desenhar **triângulo** equilátero de determinada cor e tamanho de lado; desenhar **círculo** de determinada cor e raio; posicionar uma figura **ao lado** de outra figura; **deslocar** uma figura com relação a outra, fornecendo as coordenadas (a primeira figura fica no (0,0) e a segunda é deslocada sobre a primeira); e **girar** uma figura no sentido horário em qualquer ângulo. Para resolver o problema Desenhar Imagem foram realizadas as etapas descritas a seguir:

D1 O problema foi dividido nos subproblemas menores desenhar o rosto (círculo amarelo), olhos, boca e laço, como representado no primeiro nível da árvore ilustrada na Figura 5.

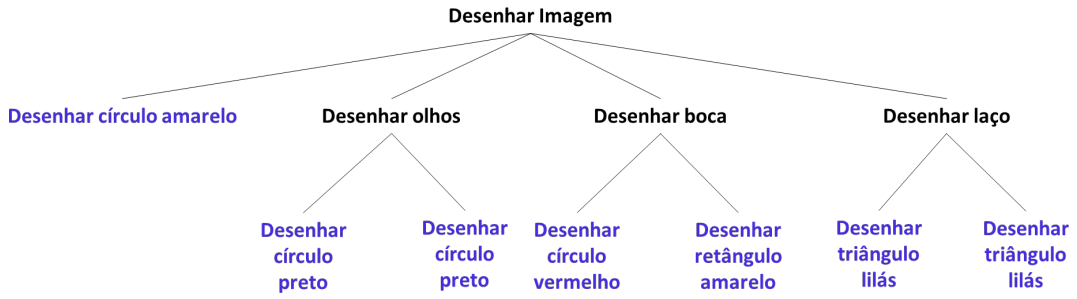


Figura 5. Árvore de subproblemas divididos até soluções triviais.

D2 Nesta etapa, os subproblemas da etapa D1 foram divididos até que tivessem soluções triviais. A divisão pode ser observada na árvore ilustrada na Figura 5. As folhas em azul representam as ações triviais.

D3 Foram definidos os CEs ilustrados na Figura 6, onde as entradas e saídas estão indicadas em cada componente. Note-se que alguns subproblemas compartilham soluções e que apenas um CE foi descrito para cada subproblema trivial distinto. Para o subproblema de desenhar retângulo amarelo, foi definido o CE **Retângulo** com três entradas: a largura (*l*) e a altura (*a*), ambas do tipo numérico (*num*), e a cor (*c*) do tipo *string*; e uma saída (*ret*) do tipo *imagem*. Para os subproblemas de desenhar círculo amarelo, preto ou vermelho, foi definido o CE **Círculo** com duas entradas: raio (*r*) do tipo numérico (*num*) e cor (*c*) do tipo *string*; e uma saída (*cir*) do tipo *imagem*. Por fim, para os subproblemas de desenhar os triângulos lilás, foi definido o CE **Triângulo** com duas entradas: lado (*l*) do tipo numérico (*num*) e cor (*c*) do tipo *string*; e uma saída (*tri*) do tipo *imagem*.

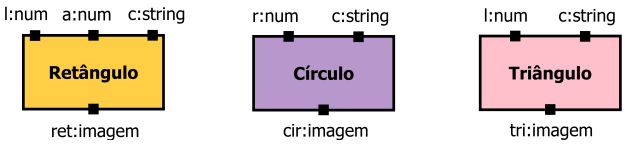


Figura 6. CEs para subproblemas triviais.

C1 Nesta etapa, adicionou-se uma instância de CE para cada instância de subproblema trivial definido na etapa D2, conforme ilustrado na parte superior da Figura 8.

C2 Esta etapa foi repetida 4 vezes. Na primeira repetição identificaram-se três menores subproblemas, isto é, subproblemas que ainda não foram solucionados, mas cujos subproblemas filhos já o foram: *Desenhar olhos*, *Desenhar boca* e *Desenhar laço*, conforme ilustrado na Figura 7. Escolheu-se o subproblema *Desenhar olhos* para resolver. Como sua solução depende da combinação das soluções dos subproblemas *Desenhar círculo preto*, identificou-se a necessidade de uma ação de combinação para posicionar os círculos lado a lado. Com isso, definiu-se o CE **Deslocar** com três entradas: duas figuras (*f1* e *f2*) do tipo imagem e uma coordenada (*xy1*) do tipo *coord* (par de valores (x,y)); e uma saída (*fig*) do tipo imagem. Adicionou-se uma instância do **Deslocar** estabelecendo as conexões (Figura 8).

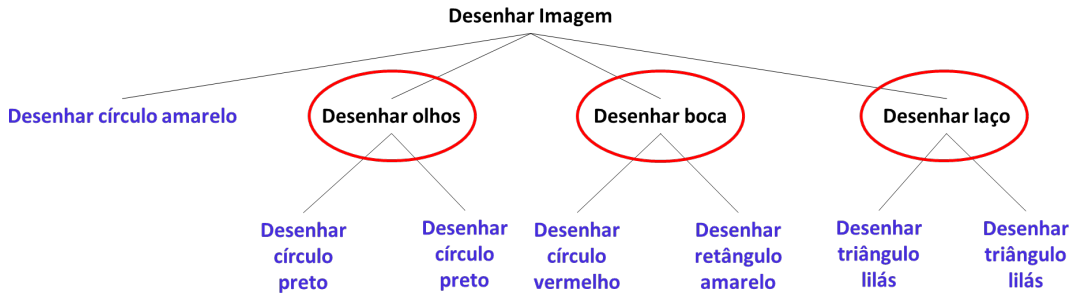


Figura 7. Subproblema menores.

Após a quarta repetição desta etapa, verificou-se que o problema foi resolvido, isto é, foi construído o componente decomposto que descreve a solução do problema **Desenhar Imagem** (Figura 9). Nessa etapa foram criados os CEs para as ações de combinação: **Descolar**, **Girar** e **Ao lado**.

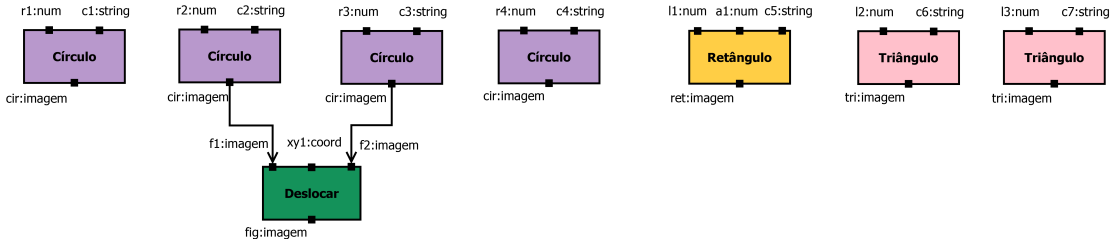


Figura 8. Combinação dos círculos dos olhos com a ação Deslocar.

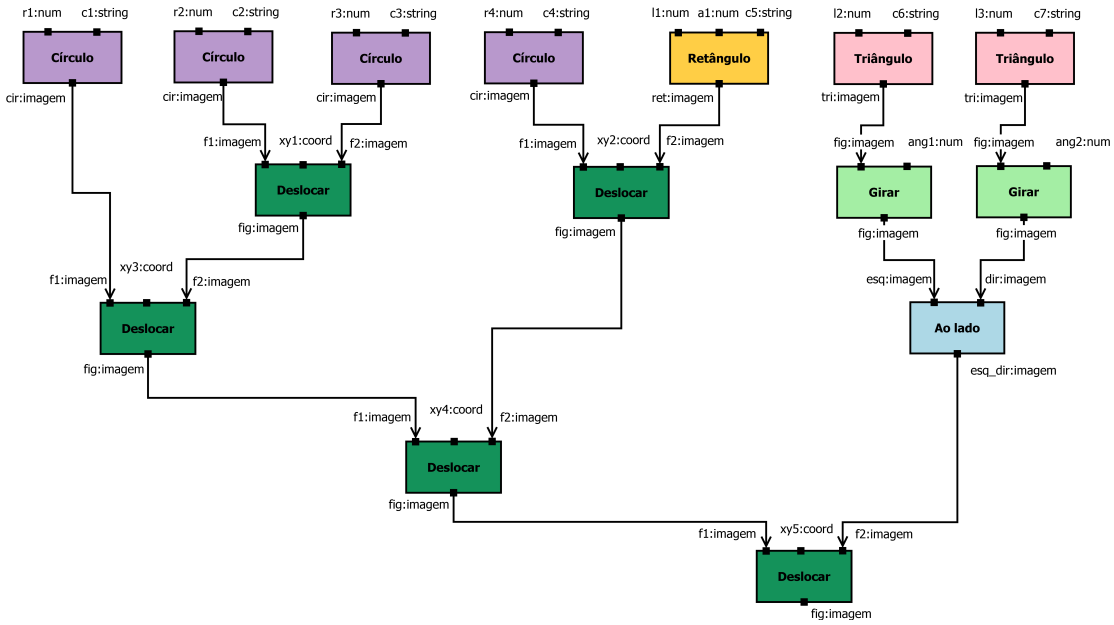


Figura 9. Componente Decomposto para o problema de desenhar a imagem.

C3 Finalmente definiu-se o CE (Figura 10) para associar ao CD definido na etapa anterior, denominado **Desenhar Imagem**, abstraindo toda a estrutura interna do CD. Esse CE tem uma porta de entrada/saída para cada porta de entrada/saída desconectada no CD definido na etapa anterior.

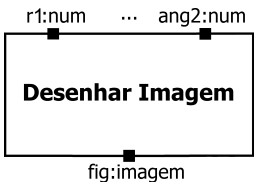


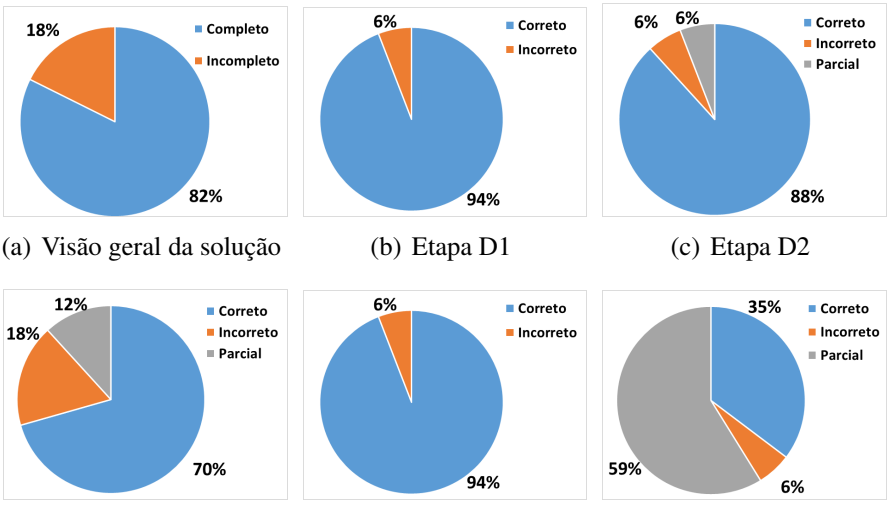
Figura 10. CE principal Desenhar Imagem.

4. Estudo de caso

Um estudo de caso foi realizado com 37 alunos do primeiro semestre do curso superior em Ciência da Computação da LiVE. Foi solicitado que os alunos utilizassem a metodologia da decomposição e composição para resolver o problema apresentado na seção anterior, considerando as ações triviais apresentadas na mesma seção. Inicialmente foi apresentada a linguagem e a metodologia, introduzidas por meio de um outro exemplo.

A turma foi dividida em 17 grupos, dos quais 15 duplas, 1 trio e 1 individual. Cada grupo recebeu um kit, contendo: uma folha de papel tamanho 42 cm × 89 cm, onde deveria ser construída a solução, um formulário para preenchimento passo a passo da atividade e sete conjuntos de diferentes cores, contendo oito papeizinhos (4,2 cm × 2,7 cm) para representar as instâncias dos CEs triviais. A atividade teve duração de 4h. A solução deveria ser realizada seguindo as etapas solicitadas no formulário de avaliação, que consistiam das etapas da decomposição (geração da árvore de solução e definição dos CEs) e das etapas da composição (construção da solução), conforme descrito a seguir: D1 - Desenhar a árvore gerada pela divisão do problema em subproblemas menores; D2 - Completar a árvore gerada na etapa anterior, dividindo os subproblemas até que tenham soluções triviais; D3 - Definir um CE para cada subproblema trivial distinto, estabelecendo entradas e saídas; C1 - Adicionar uma instância de CE para cada instância de subproblema trivial; C2 - Selecionar um dos menores subproblemas e identificar ações de combinação. Definir um CE para cada ação distinta e adicionar as instâncias necessárias, estabelecendo as conexões. Repetir até que o problema original seja resolvido; C3 - Definir CE para associar ao CD.

Os gráficos apresentados na Figura 11 resumem a análise da execução das etapas da metodologia na resolução do problema proposto no estudo de caso. Nesta análise, pode-se observar que a maioria dos grupos, 14 (82%), conseguiu completar o exercício com todas as ações triviais necessárias para a solução, assim como com todos os CEs, todas as conexões e todas as portas de entrada e saída. Apenas 3 grupos (18%) não conseguiram completar a solução, devido a ausência de um componente que represente a ação **Deslocar** e de conexões entre CEs. Analisando as etapas de divisão do problema (D1 e



(a) Visão geral da solução (b) Etapa D1 (c) Etapa D2

(d) Etapa D3 (e) Etapa C1 (f) Etapas C2 e C3

Figura 11. Análise da construção da solução com base na metodologia proposta.

D2), a grande maioria, 16 grupos (94%) conseguiu propor uma divisão adequada à solução do problema original, enquanto apenas 1 grupo não conseguiu. Já na realização das sucessivas divisões, a maioria (88%) desenhou a árvore completa, bem como identificou as ações triviais. Apenas um grupo (6%) não conseguiu realizar a atividade, enquanto outro (6%) conseguiu uma solução parcial, faltando detalhes (tais como, omitir o desenho do círculo do rosto e não girar os triângulos da gravata). Na etapa D3 a maioria, 12 grupos

(70%), definiu os CEs corretamente, identificando suas portas de entrada e saída. Dois grupos (12%) definiram os CEs sem portas de entrada e/ou saída e os outros 3 grupos (18%) não definiram os CEs. A etapa C1 da composição (inclusão dos CEs das atividades triviais obtidas na decomposição) foi realizada adequadamente por quase todos os grupos, com exceção de um. Já as etapas C2 (onde as ações de combinação deviam ser tratadas para a construção do CD) e C3 (onde a criação do CE depende do resultado da etapa C2) motraram-se mais desafiadoras. Apenas 6 grupos (35%) conseguiram construir um CD completo e que solucionava o proplema proposto. Um grupo não conseguiu sequer obter um CD aproximado, pois criou ações de combinação que não existiam e ainda usou a ação **Girar** de forma inadequada, não respeitando a quantidade entradas e seus tipos. Porém, a maioria, 10 grupos (59%), conseguiu uma solução parcial, onde as maiores dificuldades foram a definição das portas de entrada/saída das ações de combinação e a definição e uso da ação de combinação **Deslocar**.

5. Considerações Finais

Este trabalho propôs uma metodologia que sistematiza a resolução de problemas fundamentada nas técnicas de decomposição e composição usadas na construção de sistemas computacionais. Essa metodologia faz uso da linguagem de especificação visual LiVE. No estudo de caso, observou-se que a maioria dos estudantes conseguiu utilizar a metodologia corretamente, chegando à solução do problema proposto. Das vantagens citadas pelos alunos, em relação ao uso da metodologia, as que se destacaram foram: que é de fácil aplicação e que segue uma sequência lógica que conduz a solução do problema. Cabe destacar que, no estudo de caso, as ações triviais foram dadas, o que significa que as etapas D2 e C2 não foram completamente executadas pelos alunos e que pode ser uma dificuldade identificar tais ações. Além disso, com base na observação e análise do estudo de caso, constatou-se que os alunos tiveram dificuldades na identificação e uso das ações de combinação.

Assim, como trabalhos futuros, pretende-se rever as etapas da composição, em especial a etapa C2 (que trata das ações de combinação), para que possam ser melhor compreendidas e incluir ações de combinação não triviais. Além disso, pretende-se definir novas metodologias fundamentadas em outros conceitos da computação, tais como, paralelismo, simulação e automação, bem como desenvolver uma ferramenta de edição que dê suporte ao uso dessas metodologias.

Referências

Cunha, F. and Nascimento, C. R. (2018). Uma abordagem baseada em robótica e computação desplugada para desenvolver o pensamento computacional na educação básica. In *Simpósio Brasileiro de Informática na Educação*, pages 1845–1849.

Gardeli, A. and Vosinakis, S. (2017). Creating the computer player: an engaging and collaborative approach to introduce computational thinking by combining unplugged activities with visual programming. *Italian Journal of Educational Technology*, 25(2):36–50.

Howland, K. et al. (2009). Language-based support for computational thinking. In *Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 147–150. IEEE.

ISTE and CSTA (2011). Computational thinking leadership toolkit. Disponível em: `https://bit.ly/2NEYWkx`. Acesso em: 15 de março de 2020.

Koh, K. H. et al. (2010). Towards the automatic recognition of computational thinking for adaptive visual language learning. In *Proceedings of IEEE Symposium on Visual languages and human-centric computing*, pages 59–66. IEEE.

Malizia, A. et al. (2017). Tapasplay: A game-based learning approach to foster computation thinking skills. In *Proceedings of IEEE Symposium on Visual languages and human-centric computing*, pages 345–346. IEEE.

Maquil, V. et al. (2018). Kniwwelino: A lightweight and wifi enabled prototyping platform for children. In *Proceedings of the Twelfth International Conference on Tangible, Embedded, and Embodied Interaction*, pages 94–100. ACM.

Martins, L. et al. (2016). Ensinando lógica de programação aplicada a robótica para alunos do ensino fundamental. In *Simpósio Brasileiro de Informática na Educação*, pages 31–41.

Mohaghegh, D. M. and McCauley, M. (2016). Computational thinking: the skill set of the 21st century. *International Journal of Computer Science and Information Technologies*, pages 1524–1530.

Oliveira, E. et al. (2016). Pensamento computacional e robótica: Um estudo sobre habilidades desenvolvidas em oficinas de robótica educacional. In *Simpósio Brasileiro de Informática na Educação*, pages 530–539.

Pellas, N. and Vosinakis, S. (2018). The effect of simulation games on learning computer programming: A comparative study on high school students' learning performance by assessing computational problem-solving strategies. *Education and Information Technologies*, pages 1–30.

Rezende, C. M. and Bispo, E. L. (2018). Comparison between the use of pseudocode and visual programming in programming teaching: An evaluation from scratch tool. In *Proceedings of the 13th Iberian Conference on Information Systems and Technologies*, pages 1–5. IEEE.

Silva, E. and Javaroni, S. L. (2018). Pensamento computacional e atividades com robótica para a promoção da aprendizagem sobre o significado do resto da divisão euclidiana. In *Simpósio Brasileiro de Informática na Educação*, pages 815–824.

Souza, O. (2019). Joglog-jogos de raciocínio lógico para alunos do ensino fundamental: Um estudo de caso utilizando gamificação e pensamento computacional. In *Simpósio Brasileiro de Informática na Educação*, volume 30, pages 1022–1031.

Tabesh, Y. (2017). Computational thinking: A 21st century skill. *Olympiads in Informatics*, 11:65–70.

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

Yadav, A. et al. (2016). Computational thinking for all: pedagogical approaches to embedding 21st century problem solving in k-12 classrooms. *TechTrends*, 60(6):565–568.