UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
FACULDADE DE COMPUTAÇÃO

JOÃO VICTOR PATERNOSTRO CORREA

Um Estudo sobre Transferência de Conhecimento de uma Linguagem de Programação Visual para uma Linguagem de Programação Textual

Belém/PA
2017

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
FACULDADE DE COMPUTAÇÃO

Um Estudo sobre Transferência de Conhecimento de uma Linguagem de Programação Visual para uma Linguagem de Programação Textual

Trabalho de conclusão de curso apresentado como um dos requisitos para obtenção de grau de bacharel em Sistemas de Informação na Faculdade de Computação, Instituto de Ciências Exatas e Naturais.

Orientadora: Profª. Dra. Marcelle Pereira Mota

Belém/PA

2017

AGRADECIMENTOS

Agradeço, acima de tudo, à Deus por sempre me abençoar e me permitir concluir o curso de Sistemas de Informação, na renomada, Universidade Federal do Pará. Algo que eu sempre terei muito orgulho.

Agradeço aos meus pais, João Júnior e Angelica, por investirem anos na minha educação, colocando muitas vezes as minhas necessidades na frente das deles. Sem eles eu não teria chegado onde cheguei. Eu reconheço toda a luta deles para que nunca faltasse nada em minha vida.

Agradeço a minha irmã, Karinne, a minha eterna babá, Tereza, e as minhas princesas de quatro patas, Fofucha, Esmeralda e Kristal, por fazerem parte dos meus dias e de alguma forma sempre deixando a casa mais alegre.

Agradeço a minha orientadora, Marcelle, por acreditar em mim e me permitir fazer parte do seu projeto, graças a sua paciência e por compartilhar comigo seus conhecimentos, esses últimos meses na faculdade foram os que eu mais aprendi e cresci como acadêmico da faculdade de computação.

Agradeço ao meu colega Pablo pela ajuda e paciência durante o projeto. Ele também foi fundamental para essa jornada.

Agradeço a minha amada namorada, Vanessa, por me apoiar, incentivar e acreditar no meu potencial. Também por me dar energias quando eu estava cansado e sem dúvidas ela foi um dos meus maiores motivadores.

Agradeço ao meu colega André Avelino, pois desde o primeiro semestre me ajudou nas dificuldades encontradas no curso.

Agradeço aos meus grandes amigos que fazem parte da minha vida, em especial ao David, Júnior, Micuanski e André, estes são irmãos que a vida me deu.

Agradeço ao Sandro sem dúvidas o meu melhor supervisor durante os projetos no Tribunal Regional do Trabalho.

Agradeço ao meu primo, Renato, por ser o meu espelho desde criança, por ser como um irmão mais velho e por ter sido fundamental na escolha do curso. Será sempre o meu tutor!

Agradeço a todos que de alguma forma fizeram parte dessa caminhada, que me ajudaram por perto, ou que torceram por mim quando estavam longe. Eu quero retribuir esse carinho e essas boas energias que todos me deram.

RESUMO

A necessidade de aprender uma linguagem de programação é cada vez maior na sociedade. No atual contexto, tanto profissional, quanto cultural e até mesmo pessoal, cada vez mais as pessoas buscam aprender alguma forma de expressar suas ideais por meios de programas, pois pensaram em algo a ser implementado, passando então a serem usuários participativos. Porém, algumas barreiras dificultam essa busca, pois a falta de conhecimento na área de computação acaba desmotivando muitas pessoas de outras áreas de fora da computação. Este trabalho busca aplicar um modelo de design da comunicação que acontece entre o design e o usuário considerando o contexto da ferramenta Blockly, da empresa Google, que utiliza uma linguagem de programação visual em blocos para programar. O resultado da aplicação do modelo são facetas de significados de programas que formam uma documentação ativa voltada para o processo de ensino e aprendizado de programação e consequentemente o desenvolvimento do raciocínio computacional que é muito importante para solução de problemas multidisciplinares na atualidade. Utilizaremos a ferramenta Blockly junto a facetas para buscar proporcionar uma melhor experiência ao usuário iniciante, e assim realizar um estudo sobre como é possível fazer uma transferência de conhecimento de linguagem de programação visual (Blockly) para as linguagens de programação textuais (Python e JavaScript). Os resultados deste estudo indicam que as facetas desenvolvidas podem colaborar positivamente na transição entre estas duas linguagens.

Palavras-chave: raciocínio computacional; documentação ativa; polifacets; engenharia semiótica; blockly; facetas ; transferência de conhecimento.

ABSTRACT

The act of learnig code programmig has became stronger nowdays. In the present context, both professional, cultural and even personal, more and more people seek to learn some form of expressing their ideals through program methods, because they want to have applications implemented, and then become active users. However, some constrains turn this journey hard , since the lack or bad experience in coding ends up discouraging many people from other areas outside of computer science. This work aims at implementing the PoliFacets model through Google's Blockly tool, which uses a block visual programming language to generate code. The PoliFacets model is based on Semiotic Engineering and the main focus of the study is the metacommunication ontology. It is essential to define categories of facets of programs that may exist in the design of an active documentation focused on the teaching and learning process of programming and consequently the development of computational reasoning which is very important to solve multidisciplinary problems nowdays. We will use Blockly facets to provide a better user experience, and thus make a transfer of knowledge from visual programming language to a textual programming language.

Keywords: computational thinking; active documentation; semiotic engineering; polifacets; blockly; facets; transfer of knowledge.

TABELA DE SIGLAS

TIC	Tecnologias da Informação e Comunicação
IHC	Interação Humano Computador
RC	Raciocínio Computacional
DPBI	Desenvolvimento PoliFacets-Blockly I
DPBII	Desenvolvimento PoliFacets-Blockly II
LP	Linguagem de Programação

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

As tecnologias da informação e comunicação (TIC) trazem consigo o objetivo de processar e trocar as informações de maneira satisfatória às necessidades do usuário nas mais diversas plataformas (Barbosa e Silva, 2010). E ao se discutir a importância que as TIC causam na sociedade atual, percebe-se um grande impacto em várias áreas sejam elas profissional, pessoal e até mesmo cultural. Demandando de todos uma mudança no olhar sobre as novas dependências (Pretto e Silveira, 2008). No site da Comunidade Brasileira de Sistemas de Informação, há uma matéria mostrando que em alguns países mais desenvolvidos como EUA e outros países da Europa como a Inglaterra e na Oceania a Austrália, a programação em suas várias formas e linguagem se faz presente nas escolas desde o ensino fundamental e faculdades desde o seu entendimento, mas também como forma de pensamento para criação de novas ideias, pois programar nada mais é que interagir com a máquina para ela fazer o que você quer. O aluno conquista mais autonomia, raciocínio lógico e confiança por meio do desenvolvimento dessa capacidade (Florenzano, 2017).

Estas ideias podem servir tanto para solucionar problemas, mas também como inovação para novas formas de comunicação social. Esta ação se dá porque pessoas com conhecimento computacional ou domínio em alguma linguagem de programação, conseguiram expressar os seus pensamentos em ordem lógica. Devido ao seu aprendizado ligado a conteúdos de matemática e ciência que são matérias essencialmente ligadas ao raciocínio computacional (Wolz et al., 2010).

Esta forma de pensar comum a programadores, chamada raciocínio computacional, é muito importante ser desenvolvida na sociedade, tanto quanto a habilidade de leitura, escrita e aritmética, não somente para as pessoas na área da computação. Pois, o raciocínio computacional possibilita analisar problemas mais complexos presentes na sociedade atual o pensamento computacional é uma atividade de abstração a ser empregada na resolução de problemas, observando-se um ciclo previamente definido, a começar pela identificação do problema, passando pela definição e representação e na sequência a elaboração de estratégias; e buscar aplicar formas de solucioná-los (Ramos, 2015).

Porém, quando as pessoas se deparam com esse tipo de raciocínio lógico, algumas podem ficar desmotivadas a aprender devido a sua complexidade. Podemos observar isso durante o ensino de programação para pessoas sem formação técnica em computação. Os métodos comuns abordados em sala de aula de inserir o conhecimento da linguagem de

programação ao aluno por meio de códigos, instruções textuais, muitas vezes podem fazer o aluno desistir.

Neste trabalho de conclusão de curso implementamos facetas, representações do programa, que habilitam uma comunicação com o usuário para uma melhor exploração dos conceitos, significados e representações envolvidos na criação de programas construídos em uma ferramenta de linguagem visual (possui interface gráfica e não possui erros sintáticos, diferentemente das LP textuais como Java e C++ por exemplo), com o objetivo de auxiliar o usuário durante o processo de aprendizado de programação.

1.1 CONTEXTO DA PESQUISA

Ao se discutir o rumo do aprendizado da linguagem de programação percebesse a diferença, que em países desenvolvidos se tornou mais comum o ensino, devido as mudanças na área da educação implantadas por seus governos. Já em países subdesenvolvidos ainda não houve uma mudança de impacto nesse sentido, apesar do Brasil já possuir alguns projetos acontecendo dentro dele, como o Code Club Brasil[1], ainda há um longo caminho pela frente, fazendo assim o país ser um dos países com o menor número de pessoas com conhecimento de programação. Inclusive, em fevereiro de 2013, Mark Zuckerberg e Bill Gates apareceram apoiando, em um vídeo do Youtube, uma ONG destinada a ensinar linguagem de programação (Codeorg, 2013).

A falta de candidatos com competência para as áreas da computação, ciências, tecnologias, engenharia e matemática durante as trajetórias de educação e formação no ensino secundário e posteriormente no ensino universitário fazem sentir-se com intensidade e constituem hoje motivo de preocupação. Tornando necessária uma mudança nos métodos de ensino das TIC (Ramos, 2015).

Muitos projetos têm surgido por causa da necessidade de criação de ferramentas que facilitem e melhorem a experiência de usuários com o aprendizado e ensino de programação como o Scratch e o Alice 3D, que trabalham com uma linguagem de programação visual. E hoje ainda há uma necessidade além da programação, mas também de desenvolver também o raciocínio computacional por causa dos tipos de problemas que enfrentamos hoje em dia.

Outro ponto que é importante destacar é a mudança no perfil dos usuários, pois cada vez mais os usuários estão passando de usuários passivos para usuários ativos, muitos já desenvolvem programas e aplicativos que possam ajudar em determinada tarefa, seja profissional, comercial ou de uso pessoal. Esse tipo de usuário é conhecido como *End Users*

[1] https://www.codeclubbrasil.org.br/

Developers, usuários finais, que por meio de plataformas como o Joomla ou Unity 3D, podem, respectivamente, criar ou modificar sites e jogos 3D sem a necessidade de domínio de uma linguagem profissional.

O projeto pioneiro ao ensino da lógica de programação surgiu em 1967. Foi criada a linguagem de programação "Logo" por uma equipe do Instituto Tecnológico de Massachusetts (MIT), como projeto pioneiro para o apoio ao ensino de programação, liderada pelo Professor Seymour Papert, uma ferramenta para ajudar o aprendizado de linguagem de programação para crianças, jovem e até adultos (Logo, 1980). Esta simples ferramenta, porém sofisticada possui três características da computação como: Procedural, orientada a objetos e funcional. Ela consiste em uma tartaruga, como cursor, que pode se mover no plano por meio de comandos básicos como para frente, para trás e para os lados de acordo com o determinado número de passos estabelecidos (Prado, 1999).

O uso da linguagem Logo relaciona as ações da tartaruga com os comandos de programação. Definindo em procedimentos uma sequência de ação para a tartaruga seguir, estas ações ficam armazenadas no computador para serem executadas quando desejadas. Repare na Figura 1 como a tartaruga se comporta após a execução das instruções.

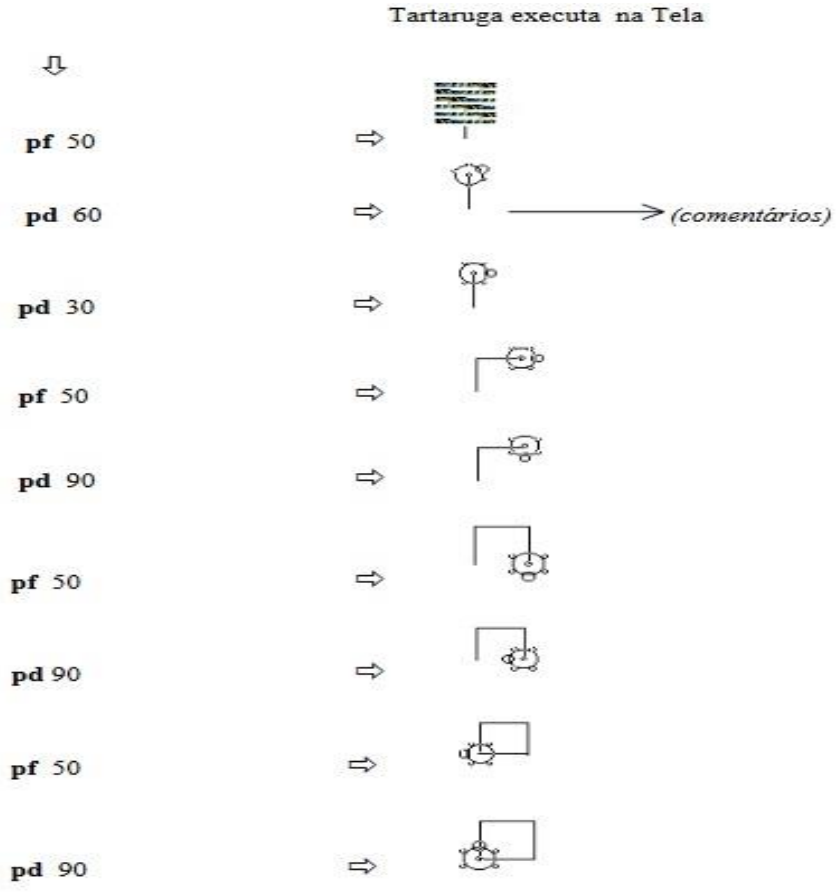


Figura 1 – Ferramenta Logo (Prado, 1999)

A tartaruga executa os passos de acordo com o implementado pelo usuário, parecido com as instruções dos códigos de programação.

A ferramenta Logo permite ao professor e ao aluno uma reflexão “na ação” e uma reflexão “sobre a ação” quase que simultaneamente. Esta abordagem pedagógica permite uma análise computacional sobre a lógica da linguagem de programação, ajudando assim no ensino e aprendizado de programação de forma acessível e podendo ser utilizada por usuários de diversas áreas (Prado, 1999). Desde então muitos outros projetos surgiram com diferentes abordagens, mas com o mesmo objetivo.

Por exemplo, o projeto AgentSheets de Alexander Repenning (Repenning e Ioannindou, 2004), uma linguagem de programação visual, usada para aplicações de ciência computacional. Essa ferramenta permite o usuário “arrastar” e “soltar” elementos visuais que são equivalentes a comandos para a criação de jogos e simulações. E também o PoliFacets-AgentSheets, que utiliza a LP visual e mostra informações criadas, aos alunos por meio de uma documentação ativa dentro das facetas. Desta forma os alunos puderam entender melhor os significados e conceitos evidenciados, colaborando aos aprendizes terem uma compreensão mais completa dos programas que eles desenvolveram (Mota, 2014).

Além disso, existem habilidades dentro do RC que podem ser aplicadas em diferentes contextos (Ramos, 2014). Como:

a) Quebrar um problema em partes ou etapas ‘’Dividir para conquistar’’;
b) Reconhecer e generalizar padrões e tendências em regras, princípios ou ideias;
c) Identificar o que é mais relevante sem se preocupar muito com detalhes "abstração".

RC não é apenas programação. Programação é apenas uma ferramenta do RC (Ramos, 2014). No ponto de vista técnico, somente isso não seria suficiente. Há todo um conjunto de aspectos que precisam ser levados em consideração. A máquina é apenas uma extensão da capacidade da nossa forma de pensar, nos dando mais velocidade, mais recursos, para poder chegar mais longe. Mais na realidade, é algo do fundamento humano. Então usar o RC não é pensar só como uma máquina.

1.2 MOTIVAÇÃO

Nem todos os problemas se resolvem com computação, mas uma parte deles pode ser. Escolher e entender a limitação das técnicas e analisar formas de resolver o problema, como dividir um problema grande em peças (dividir e conquistar) e assim achar soluções para cada peça e depois juntá-las. Técnicas utilizadas dentro da computação podem ser utilizadas

para resolver problemas de outras áreas. Buscar melhorar o contato com aprendizes de programação é muito importante para a sociedade. Isso assegura a continuidade desses alunos para o aprendizado de uma linguagem de programação textual.

Este trabalho busca aplicar uma forma de transferência de conhecimento do Raciocínio Computacional, por meio do modelo PoliFacets, por meio de uma documentação ativa em uma ferramenta que melhore a experiência de alunos e também de professores durante o aprendizado de programação de uma linguagem de Programação Visual e posteriormente uma Linguagem de programação Textual. Este estudo apresenta um potencial de grande utilidade para os aprendizes e professores de programação e possui a Engenharia Semiótica como a principal contribuição científica para o modelo utilizado neste trabalho.

1.3 OBJETIVOS

Este trabalho tem como objetivo geral apresentar um estudo sobre transferência de conhecimento entre uma linguagem de programação visual para uma linguagem de programação textual.

1.3.1 OBJETIVOS ESPECÍFICOS

Os objetivos específicos são:

- Aplicar o modelo PoliFacets na ferramenta Blockly do Google, criando facetas que ajudem o a gerar uma boa experiência ao usuário durante o processo de ensino e aprendizado de programação por meio de uma linguagem de programação visual;
- Oferecer ao usuário o suporte de diferentes tipos de informação dentro de facetas específicas para saber como, por que e onde proceder na criação do programa por meio de uma documentação ativa dentro delas;
- Realizar a avaliação das facetas desenvolvidas por meio de testes de usabilidade;
- Consolidar os resultados para melhorar a implementação das facetas.

1.4 METODOLOGIA

O desenvolvimento do PoliFacets-Blockly (DPB) ocorreu pela necessidade de uma abordagem diferente ao ensino e aprendizado de programação. É muito importante que a introdução dos fundamentos de computação não seja uma experiência traumatizante, pois a continuação dos estudos dependerá muito da abordagem inicial. Já que elas norteiam o resto do curso, pois possuem os principais fundamentos teóricos computacionais. Então, se torna

determinante superar essa dificuldade para o ingresso e evolução do aluno durante o curso (Aureliano e Tedesco, 2012).

Com base nisso aplicamos o modelo PoliFacets na ferramenta de linguagem de programação visual do Google, o Blockly (uma ferramenta de código aberto, que utiliza uma linguagem de programação visual em blocos, veremos melhor esse programa nos próximos capítulos). E desenvolvemos facetas de significados de programa, onde o principal objeto de estudo é a ontologia de metacomunicação, informação da informação, para a criação da documentação ativa, que é uma informação representada de forma interativa em "tempo real" para ajuda ao usuário dentro de facetas. O modelo PoliFacets (Mota, 2014) tem como principal base teórica a Engenharia Semiótica (de Souza, 2005), uma teoria de IHC, que possui como o seu principal foco a metacomunicação entre o designer e o usuário por intermédio dos sistemas computacionais.

Este trabalho foi dividido basicamente na etapa inicial de fundamentação teórica, junto ao estudo do modelo PoliFacets. Para posteriormente o desenvolvimento do PoliFacets-Blockly I (DPB I). Aplicação do 1º teste com usuários (Teste 1) e 1ª amostra de resultados (Resultado 1).

Após os resultados obtidos no 1º teste com usuários, podemos aplicar melhorias, correções e sugestões para o desenvolvimento do PoliFacets-Blockly II (DPB II). Assim foi feito uma nova rodada de teste com usuários (Teste 2) e novos resultados (Resultado 2). Isto será melhor abordado nos próximos capítulos. Na Figura 2, observe o fluxograma das fases de desenvolvimento do projeto, o desenvolvimento da primeira fase do trabalho, junto ao teste 1 e os seus resultados, após isso o desenvolvimento da segunda fase, junto ao teste 2 e os seus resultados.

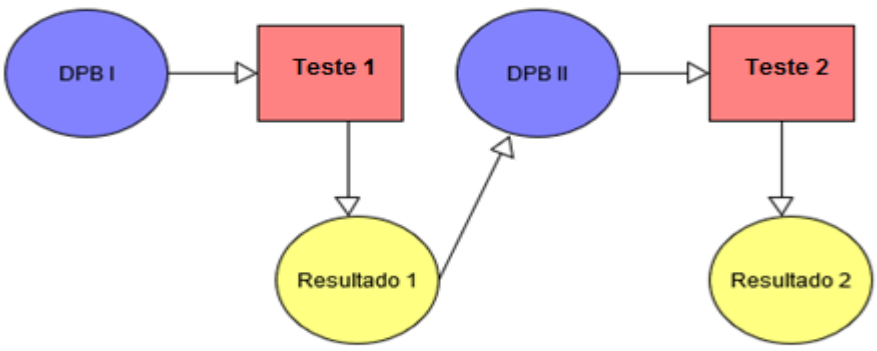


Figura 2 - Fases de desenvolvimento do PoliFacets-Blockly

2 TRABALHOS RELACIONADOS E FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a teoria necessária para o entendimento desse trabalho. Nele está descrito os trabalhos relacionados, a principal fundamentação teórica, o modelo e a ferramenta utilizada para alcançarmos os objetivos do nosso estudo que foi desenvolver uma ferramenta para apoiar o ensino aprendizado de programação por meio de uma linguagem de programação visual apoiada por facetas de significados de programas com uma documentação ativa específica dentro de cada uma, de forma que auxiliem o usuário a utilizar e compreender a construção do seu programa para posterior aprendizado de uma linguagem de programação textual.

2.1 TRABALHOS RELACIONADOS

Diversos trabalhos tem surgido com o objetivo de tornar o ensino de programação mais fácil, como o Alice 3D, Scratch e o AgentSheets que utilizam linguagem de programação visual para iniciantes em programação, falaremos melhor deles logo abaixo. Muitos destes buscando formas de desenvolver o raciocínio lógico e o raciocínio computacional aos usuários. Apresentaremos a seguir os trabalhos correlatos que foram considerados mais importantes para o desenvolvimento deste trabalho de conclusão de curso.

O programa Scratch (Dias e Serrão, 2014), muito conhecido, trabalha com a utilização de blocos de instrução para a manipulação de mídias, como imagens e sons, no processo de criação de jogos e animações e serve como ferramenta para iniciar o aprendizado de programação. O Scratch é uma ferramenta para introduzir os alunos a aprenderem outras linguagens de programação como Java e Python, por exemplo, pois por meio dela é possível trabalhar a lógica da programação sem a preocupação com a perda de tempo e desvio de foco para a correção de erros de sintaxe, dificuldade encontrada na maioria das linguagens (Dias e Serrão, 2014). Veja na Figura 3, como a ferramenta oferece um ambiente visual de programação, buscando torna-la mais acessível para o seu público alvo, por meio de uma interface que permite que programas sejam montados utilizando blocos virtuais.

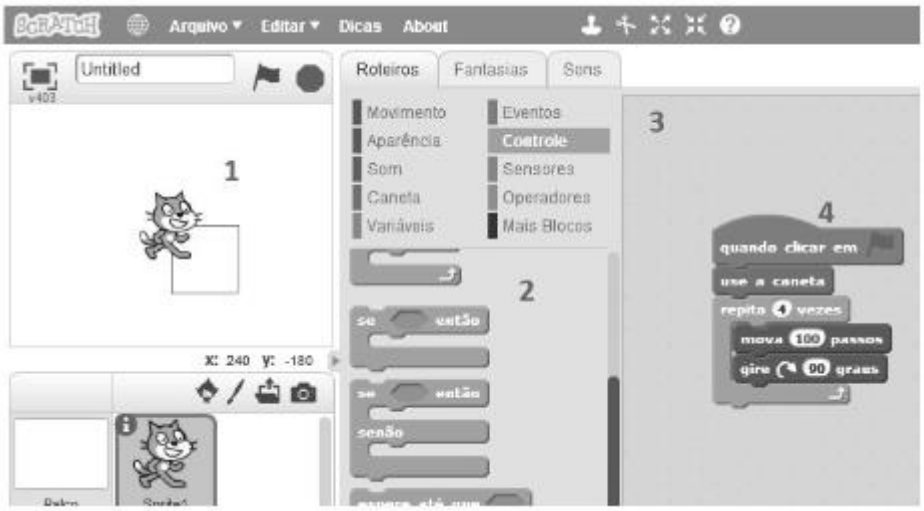


Figura 3 - Tela do Scratch (Dias e Serrão, 2014)

Uma pesquisa na universidade de Harward (Malan e Leitner, 2007), por meio de uma enquete para saber se a ferramenta Scratch havia ajudado no entendimento da LP Java, 76% dos alunos confirmaram a influência positiva. Enquanto 16% disseram não ter influência, por já terem experiência com a linguagem de programação.

Outra plataforma nessa linha é o Alice 3D (Gondim et al., 2011). Possui a mesma finalidade do Scratch e permite criações de animação e jogos por meio de um ambiente gráfico interativo em três dimensões. O usuário escreve comandos simples, similares a Java e C++, que tem efeito sobre o controle e a aparência dos objetos colocados no cenário. Dessa forma, o aluno vai aprendendo a lógica da programação sem se preocupar com as dificuldades desmotivadoras de uma linguagem de programação. Veja na Figura 4, a tela da plataforma Alice 3D, como o usuário pode criar um cenário por meio de manipulação da câmera, luz, chão como objetos e métodos dentro da linguagem de programação orientada a objetos.

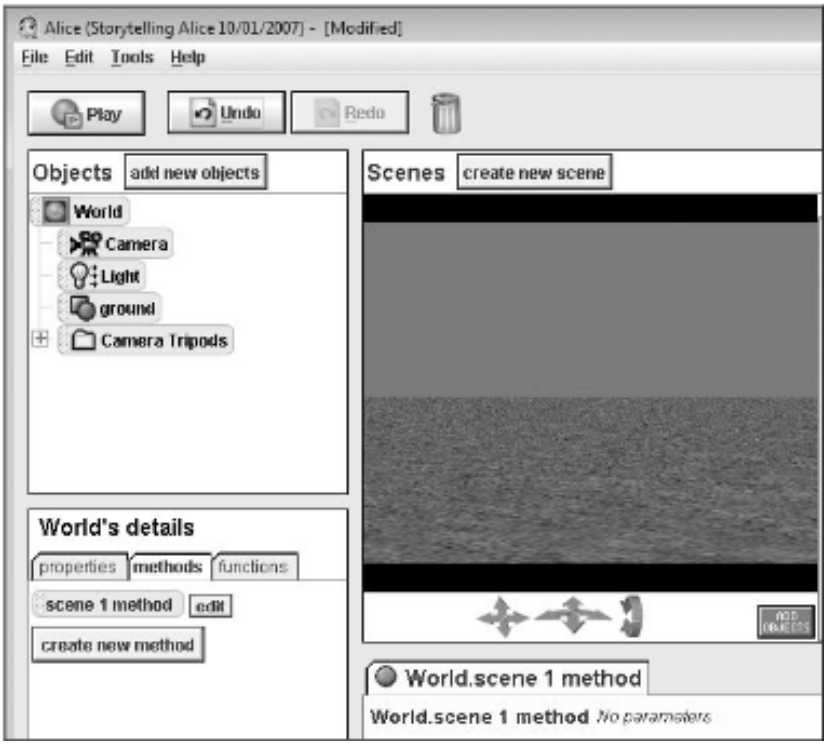


Figura 4 - Plataforma Alice 3D (Gondin, 2011)

Também pesquisamos como referência o AgentSheets, uma tecnologia proprietária comercializada pela AgentSheets Inc., que disponibiliza um ambiente de aprendizado do raciocínio computacional por meio da criação de jogos ou simulações utilizando uma linguagem de programação visual em 2D. A programação é feita por meio de "arrastar e soltar" de condições e ações que formam as regras do comportamento dos personagens no jogo. Na interação do aprendiz com os elementos visuais do ambiente, os elementos são equivalentes a comandos dentro da programação. A complexidade dos jogos e simulações desenvolvidos dentro do AgentSheets é compatível com a de programas desenvolvidos por alunos de graduação de informática. Veja um exemplo de jogo do AgentSheets na Figura 5.

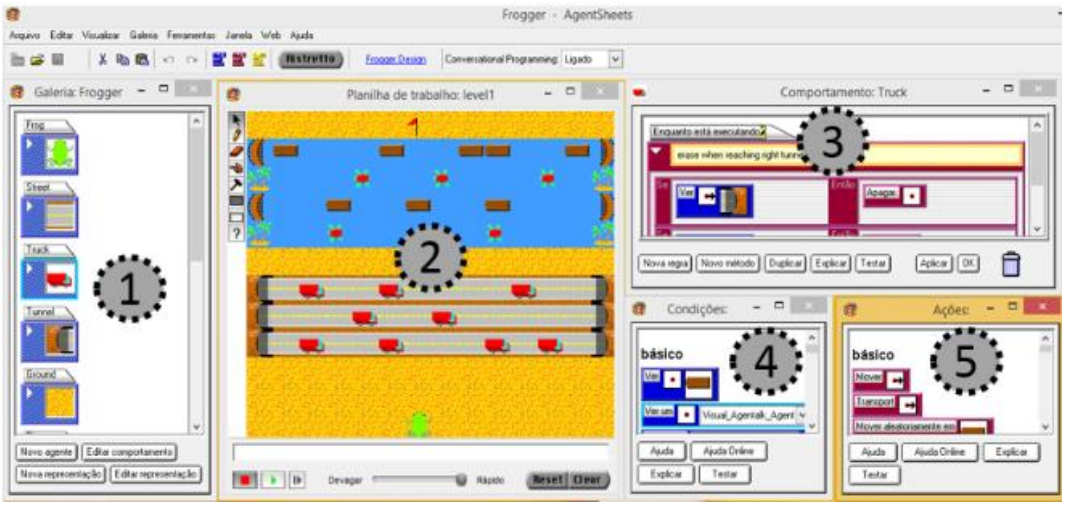


Figura 5 - Exemplo de jogo no AgentSheets (Bastos, 2015)

Dentro do AgentSheets todos os projetos possuem seus agentes (Número 1), uma planilha (Número 2) uma grade bidimensional onde estão posicionados (estaticamente) e atuam (dinamicamente) todos os objetos do jogo. As regras (Número 3) com os comandos “Se-Então” definindo o comportamento lógico de produção. O usuário programa os agentes (Número 4) arrastando e soltando as regras de condições associadas aos comandos “Se” e, por último, as ações (Número 5) associadas aos comandos “Então”.

Em um trabalho anterior ao projeto PoliFacets-Blockly. Foi utilizado uma versão de uma ferramenta chamada IvProg, ferramenta de LP visual em 2D que permite a criação de programas por meio de blocos de instruções desenvolvida em um laboratório de informática na educação da Universidade de São Paulo. O IvProg é uma aplicação OpenSource com aplicação direta na internet e nele foram implementadas as facetas para introdução de conceitos de lógica de programação, onde elas puderam evidenciar ao usuário conteúdos ligados ao apoio do aprendizado e otimização dos programas (Barata e Mota, 2016). Os resultados obtidos no PoliFacets-IvProg foi de grande importância aos rumos, posteriormente, para o início do PoliFacets-Blockly.

Veja na Figura 6 - Tela de instruções do IvProg (Barata e Mota, 2016) adiante um exemplo de um programa para verificar a paridade dos números no IvProg. Nele devem ser solicitado uma quantidade de números a serem lidos, e em seguida identificar quais números são pares e quais são ímpares.

Figura 6 - Tela de instruções do IvProg (Barata e Mota, 2016)

Outro trabalho com grande influência nos rumos da nossa pesquisa foi o GreenFoot (Kölling, 2010) aplicado ao apoio de transferência de conhecimento de raciocínio computacional de LP visuais para LP textuais (Bastos, 2015). Desenvolvido na PUC-Rio, esta pesquisa utilizou o GreenFoot que é um ambiente de programação textual para auxílio da aquisição de raciocínio computacional por meio da criação de jogos e simulações em 2D. Esta ferramenta possui uma proximidade com a linguagem de programação profissional Java. Observe na Figura 7 a seguir o workspace do GreenFoot. Onde nós podemos implementar o mundo (Número 1) em uma grade bidimensional onde os atores poderão interagir. Os atores são classes que se destinam a aparecer no mundo (Número 2), cada um possui seu próprio comportamento e aparência programados individualmente. E ambos os atores e mundos devem ser programados por meio da linguagem Java e o framework do Greenfoot. Todos estes juntos constituem o código fonte do projeto (Número 3 e 4).

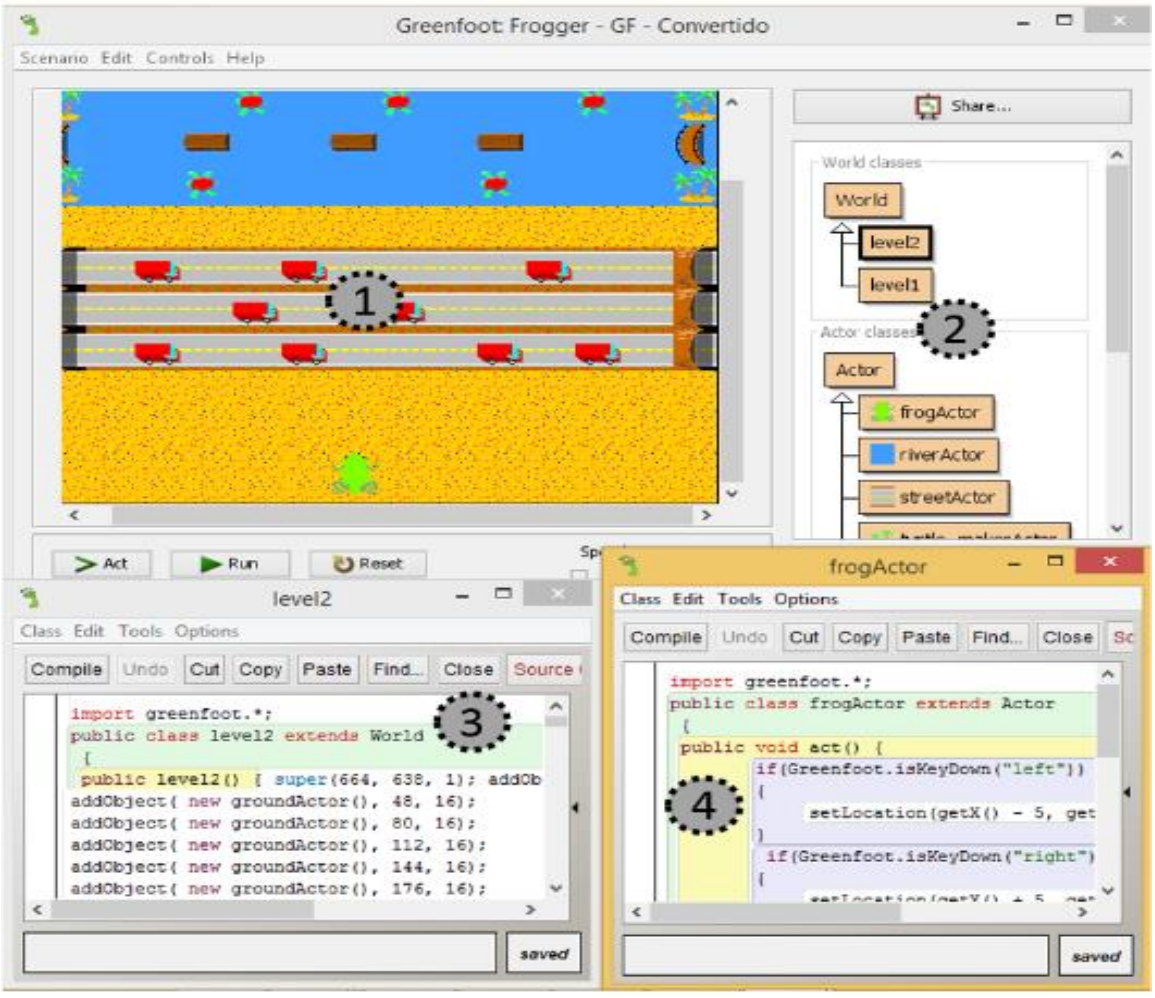


Figura 7 - Workspace do GreenFoot (Bastos, 2015)

Os trabalhos que foram apresentados tiveram fundamental importância para o desenvolvimento deste trabalho de conclusão de curso, onde a busca dos mesmos objetivos junto à análise dos resultados, definiram as nossas atividades dentro do projeto.

O PoliFacets-Blockly possui como semelhança aos projetos mencionados a utilização de uma linguagem de programação visual em Blocos, porém a diferença é que foi desenvolvidas facetas de significado de programas, que através de uma documentação ativa buscam ajudar o aprendiz na construção do seu programa. Para assim poder contar com uma ferramenta a mais para compreensão da lógica de programação e, posteriormente, para a transferência de conhecimento para uma LP textual de forma mais suave.

2.2 ENGENHARIA SEMIÓTICA

A Engenharia Semiótica é uma teoria de IHC que caracteriza aplicações computacionais como artefatos de comunicação sobre a comunicação que ocorre entre o designer (programador) e o usuário do sistema computacional interativo. Um objeto que transmita uma mensagem do designer ao usuário sobre a comunicação usuário–sistema, sobre como ele pode e deve utilizar o sistema, por que e com que efeitos (de Souza, 2005).

A comunicação proposta por essa teoria envolve um modelo, criado por Jakobson em 1960 (Jakobson, 1960), composto por três interlocutores basicamente: O designer, o código e o usuário. A comunicação entre eles vai acontecer porque o designer sente a necessidade, em um determinado contexto, de passar uma informação, explicação ou ideia por meio de um canal ou código ao usuário. Para que esse possa utilizar o sistema adequadamente. Ou seja, o designer quer interagir com o usuário, por meio de uma informação que possa ajuda-lo na utilização do sistema tornando o processo mais fácil de modo que evite desmotivação do usuário com as dificuldades que possam ser encontradas durante o manuseio do sistema.

A engenharia semiótica pode ser compreendida por (de Souza, 2005):

a) Processos de significação: Envolvem signos e semiose;
b) Processos de comunicação: Envolvem intenção, conteúdo e expressão nos níveis de comunicação;
c) Os interlocutores: Envolvidos nos processos de significação e de comunicação: São estes o designer, o sistema e o usuário.
d) O espaço de design de IHC: Baseado no modelo de espaço de comunicação de Jakobson: Interlocutores, contextos, códigos, canais e mensagens.

Para que haja sucesso nessa comunicação, o designer deve escolher cuidadosamente a expressão para o conteúdo que deseja comunicar, por meio de um código que o usuário seja

capaz de interpretar, e no caso de IHC, que o sistema seja capaz de processar (Barbosa e de Silva, 2010).

Este código deve atender a certos cuidados para que possa haver sucesso no processo de comunicação (de Souza, 2005). Por exemplo:

a) Utilizar metáforas, analogias são bem vindas, mas é necessário ter cuidado com o excesso para não confundir o usuário;
b) Evitar ambiguidades ou falta de clareza para o manuseio da ferramenta. Diminuindo assim incertezas e a angústia do usuário para com ela.

2.3 MODELO POLIFACETS

O modelo PoliFacets, fundamentado na teoria da Engenharia Semiótica, foi concebido por meio de diversos estudos empíricos que resultaram em informações sobre como diferentes formas de representação de programas podiam ser projetadas e exploradas (Mota, 2014). O modelo é um auxilio para criação de documentação ativa, representada por meio de facetas de significados sobre programas. Em cada faceta desenvolvida, buscamos estabelecer uma comunicação entre o designer para com o usuário afim de ressaltar informações consideradas importantes para ajudá-lo no aprendizado de programação, nesse caso especificamente, o foco é a transferência de conhecimento partindo de uma programação visual para uma programação textual.

O modelo PoliFacets é uma ferramenta epistêmica para conceber facetas de significados de programas por meio de uma ontologia de metacomunicação. As facetas são o produto final da aplicação do modelo, e se propõe a funcionar como uma conversa entre os usuários do software o software produzido pelo designer (Mota, 2014). As diferentes facetas tem como objetivo ajudar o usuário a compreender o objeto de estudo por meio de visualizações diferentes de acordo com cada categoria delas, provocando assim uma reflexão mais abrangente sobre o programa. No processo de criação dessa conversa entre o designer e o usuário é importante salientar a necessidade de ligação entre as facetas, para que não ocorram rupturas na conversa.

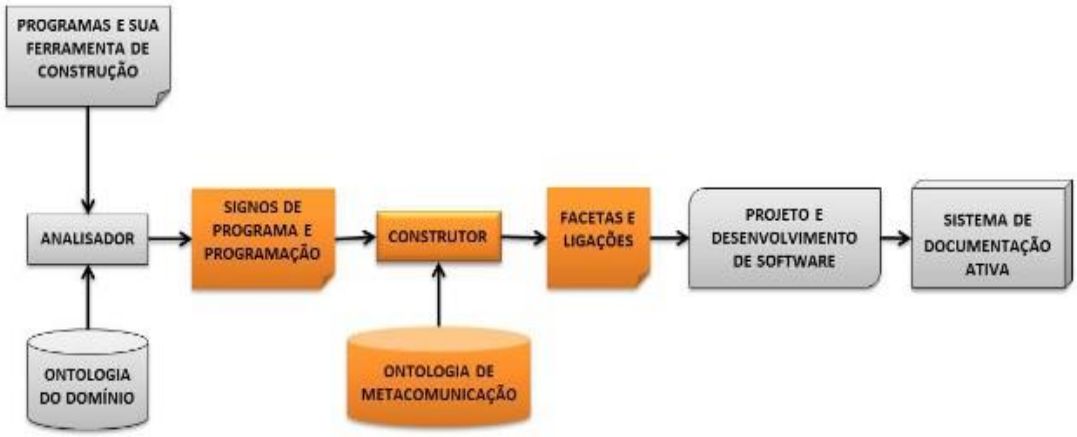


Figura 8 - Modelo PoliFacets no contexto de design da tecnologia (Mota, 2014).

A Figura 8 acima mostra a estrutura do modelo PoliFacets durante o processo de análise da ferramenta e das informações, para que elas possam ser organizadas para posterior criação da documentação ativa para desenvolvimento das categorias de facetas, destacando informações importantes, ontologia de metacomunicação, para análise do usuário servindo assim de apoio ao ensino e aprendizado de programação e do raciocínio computacional.

A ontologia do domínio é extraída de dados da aplicação por meio da análise do código fonte ou da representação dos programas e suas ferramentas, para que estas sejam organizadas do ponto de vista léxico, sintático e semântico para a qual a documentação ativa será criada. A ontologia do domínio representa tudo o que podemos conhecer sobre a aplicação e, em conjunto com os programas e ferramentas de construção, será utilizado pelo analisador para interpretação das informações (Mota, 2014).

A ontologia de metacomunicação é a parte principal do modelo. Ela é essencial para a definição de categorias de facetas de programas que podem existir no design de documentação ativa voltada para o processo de ensino e aprendizado de programação. A análise e interpretação dessas informações permite uma posterior explicitação dos conceitos escondidos através da criação de outras facetas.

2.3.1 DOCUMENTAÇÃO ATIVA

Os documentos ativos tem sido criados e definidos de diversos modos. E um destes modos nós temos eles para sistemas Web que possuem como algumas funções a gerência de documentos, automatização de processos e compartilhamento de conhecimentos. Bastando apenas o preenchimento de alguns dados para o documento completo ser gerado automaticamente. Estes sistemas possuem seus documentos criados com base em templates

pré-definidos, dispensando assim a formatação do arquivo. Como exemplo há o PAE (Processo administrativo Eletrônico) muito utilizado por instituições e tribunais pelo Brasil como (Tribunal Superior do Trabalho, Tribunal Regional do Trabalho e outros) que possuem modelos de cartas e processos já pré-definidos para ajuda e agilização do usuário no sistema[2].

A documentação ativa deste trabalho se alinha a pesquisa de Aßmann (2005), que propõe vários estilos de arquitetura para documentos ativos. Esta linha define que o documento deve reagir de forma interativa, ou seja, o campo reflete certas ações ao serem preenchidos. Estes documentos podem ser formados por dados e macros; ou dados e scripts. Os documentos ativos possuem componentes que são derivados de forma automática a um conjunto de base e exploram potencialmente a programação para representar conteúdos de forma clara.

2.4 BLOCKLY - GOOGLE

O Google disponibilizou uma linguagem visual de programação baseada em blocos chamada Blockly para facilitar a programação de aplicações, bastando o mouse para arrastar e soltar as peças, dispensando, parcialmente, a utilização do teclado durante o processo de criação.

Ela permite gerar o código dos blocos em linguagens como Java, JavaScript, Lua, Dart e XML. O Blockly é uma biblioteca para desenvolvedores, que adiciona um editor de código visual para aplicações web ou Android. Além disso, ela é uma ferramenta Open Source (código aberto) permitindo assim a manipulação de dados para implementação da forma que desejarmos.

Essa Ferramenta classifica os blocos em categorias de: Lógica, Laços, Matemática, Texto, Listas, Cor, Variáveis e Funções. Dentro de cada categoria há os blocos referentes aos comandos usados na programação, para que o usuário possa escolher e “montar” o código. Veja na Figura 9 a tela do Blockly e repare as categorias dos blocos à esquerda e os blocos no meio para construção do programa.

[2] http://www.tst.jus.br/web/autoatendimento/pae-processo-administrativo-eletronico

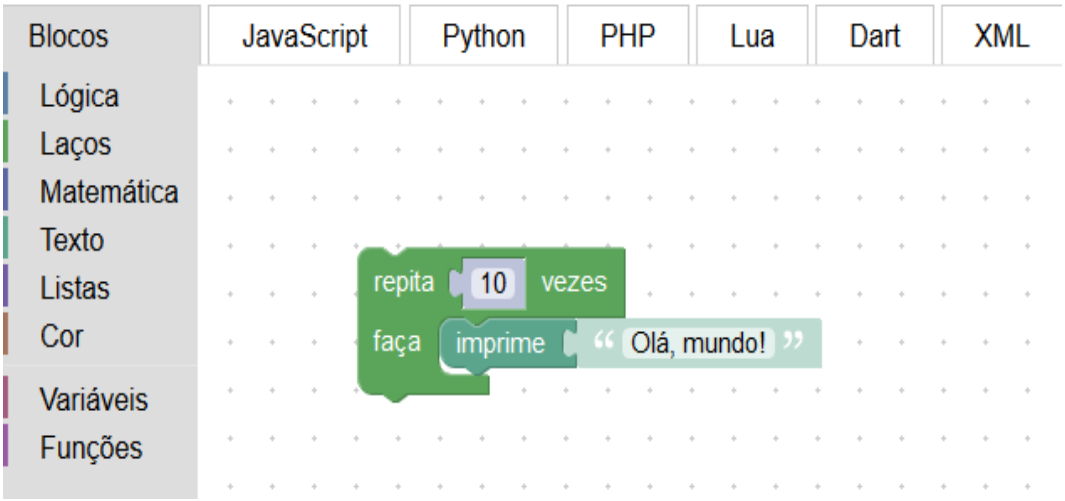


Figura 9 – Tela inicial da Ferramenta Blockly do Google

Assim aplicamos o modelo PoliFacets ao Blockly, para construirmos as facetas desenvolvendo uma documentação ativa dos programas criados pelos usuários, buscando facilitar o entendimento dos programas e promover a transferência de conhecimento entre linguagens.

A escolha do Blockly se deu pela sua linguagem ser constituída em JavaScript e poder se exportar os blocos em XML para posterior manipulação destes como objeto de estudo (ontologia do domínio) para a criação das facetas. Portanto, estabelecemos uma ligação entre conceitos de raciocínio computacional e a ontologia de metacomunicação para idealizarmos as facetas do Blockly (Barata e Mota, 2016).

2.5 TRANSFERÊNCIA DE CONHECIMENTO

A transferência de conhecimento é uma etapa de continuação do modelo PoliFacets, posterior aos indicativos de eficiência de um melhor entendimento dos conceitos relacionados a programação considerando apenas uma linguagem. Nesta etapa fica subentendido que o aprendiz passou pela experiência de utilizar uma LP visual e agora com conhecimentos de raciocínio computacional, poderá prosseguir em direção ao contato com uma LP textual apoiada pelo modelo de transferência de conhecimento.

A transferência de conhecimento serve para analisarmos os conceitos e elementos mais utilizados na programação pelos usuários aprendizes durante a experiência de utilizar uma LP Visual, e se eles conseguiram compreender os conceitos de programação para assim coloca-los em contato com uma LP textual. Ele foi utilizado para obtermos as provas de sucesso e falhas no estudo empírico de transferência de conhecimento pela nossa ferramenta, o PoliFacets-Blockly.

3 POLIFACETS-BLOCKLY

De acordo com o modelo PoliFacets, utilizamos duas formas de paráfrase na criação das facetas do Polifacets-Blocly. Vale dizer que paráfrase significa apresentar uma mesma ideia de forma diferente. As facetas desenvolvidas podem ser do tipo que apresentam reescrita ou reformulação. As facetas com paráfrase do tipo reescrita acontecem quando há uma representação diferente de um mesmo conteúdo. Conseguindo assim gerar informações contextualizadas, por meio de novos tipos de combinações e assim fornecendo novas informações ao usuário (Mota, 2014). Esse tipo de mudança busca fornecer novas informações e destacar outras que nós consideramos importante durante a criação das facetas, como é o caso das facetas ‘Suporte’, ‘Tags’ e ‘Chart’ que foram criadas durante o desenvolvimento do PoliFacets-Blockly. Estas facetas serão descritas individualmente abaixo.

As facetas do tipo Paráfrase - Reformulação acontecem quando estas expressam o mesmo conteúdo de modo diferente, mas sem modificar a semântica. Para o usuário a “expressão” e a “intenção”, para o sistema “léxico e/ou sintático. Ou seja, a possibilidade de uma nova representação quanto a expressão ou forma de comunicação da nova intenção. Dentro do sistema deve acontecer uma mudança no vocabulário e/ou na gramática da linguagem (Mota, 2014). Isso seria reescrever o mesmo só que de formas diferentes por exemplo um comando de “Write” em uma linguagem como o Pascal e o comando “Imprimir” em uma linguagem natural. Dentro desta categoria utilizamos as facetas ‘JavaScript’ e ‘Python’, já existentes no Blockly, porém aplicamos algumas mudanças nelas. E também, ainda na mesma categoria, desenvolvemos as facetas ‘Requisitos’ e ‘Instruções’. Todas elas serão descritas individualmente em detalhes a seguir.

3.1 DESENVOLVIMENTO POLIFACETS-BLOCKLY I

No desenvolvimento do PoliFacets-Blockly I, algumas modificações na tela do Blockly foram feitas como a questão das cores para podermos utilizar melhor as informações geradas nas facetas. As facetas utilizadas do PoliFacets-Blockly I foram as ‘Instruções’, ‘Suporte’, ‘Tags’ e ‘Código JS’. Observe na Figura 10 (Barata et al., 2017) adiante como ficou o nosso protótipo do Blockly. Repare as novas cores dadas as categorias à esquerda (No campo Toolbox) e um exemplo de programa na LP visual para verificar a paridade de um número no espaço de trabalho (Workspace).

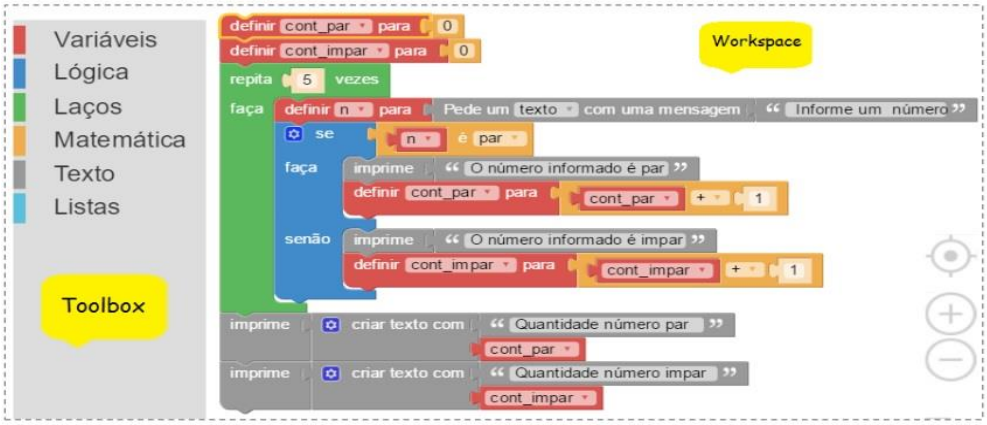


Figura 10 - Protótipo do Blockly – PoliFacets. Exemplo: Verificar número. (Barata et al., 2017)

Em cima disso foram feitos dois protótipos de programa. Para o primeiro teste com usuários, a versão inicial, para ela foram utilizadas as 4 facetas citadas acima para a construção de um espaço de reflexão sobre significados e conceitos dos programas construídos no workspace do Blockly. Estas facetas foram desenvolvidas inicialmente a partir da ideia que os aprendizes terão que em algum momento transitar da linguagem de programação visual para uma linguagem de programação textual. No primeiro teste com usuários a linguagem escolhida foi o JavaScript, por causa da sua ampla utilidade em navegadores Web nos dias de hoje (Implementações Cliente-Servidor).

No Quadro 1 - Classificação das facetas do PoliFacets-Blockly I mostramos as facetas utilizadas na primeira fase do protótipo e sua classificação de acordo com o modelo PoliFacets. Durante a experiência do aprendiz há uma ordem proposital na posição delas para que ele possa ir subindo de nível para o aprendizado de programação. A ordem das facetas ficou da seguinte forma 1) Instruções; 2) Suporte; 3) Tags e 4) Código JS. Onde estas serão melhor detalhadas a seguir.

Tipos de Faceta	**Facetas do PoliFacets-Blockly I**
Facetas de Paráfrase-Reformulação	• Instruções; • Código JS.
Facetas de Paráfrase-Reescrita	• Suporte; • Tags.

Quadro 1 - Classificação das facetas do PoliFacets-Blockly I

3.1.1 FACETA 'INSTRUÇÕES'

Esta é a primeira faceta, ela representa o programa de uma forma próxima a linguagem natural. Esta busca ajudar o usuário iniciante a compreenderem um pouco mais o seu programa, para posteriormente fazer a transferência conhecimento para uma LP textual

como a JavaScript ou Python. Cada comando referente a um bloco possui um hiperlink para ligação com a faceta ‘Suporte’. Veja na Figura 11 a seguir o exemplo na faceta ‘Instruções’ e repare na linguagem natural descrevendo cada instrução do programa para que o aprendiz possa utilizar essa faceta como para compreender em uma LP textual, mesmo que seja uma linguagem de natural, o seu programa.

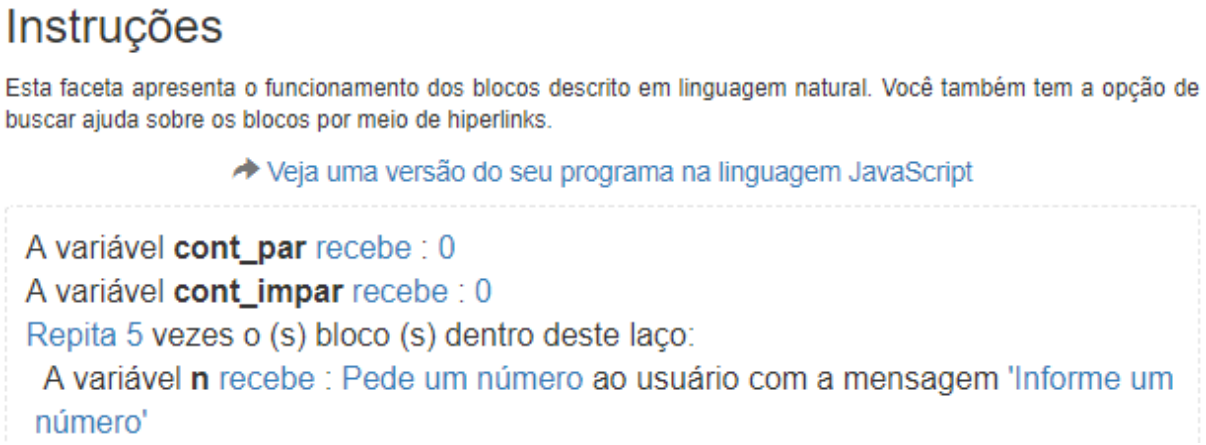


Figura 11 - Exemplo da faceta Instruções

3.1.2 FACETA ‘SUPORTE’

Esta é a segunda faceta, ela dá nome aos blocos, pois eles não possuíam nomes, então percebemos a necessidade de dar nome a eles e classificamos estes pelas cores de suas categorias, para os usuários identificá-los mais rapidamente. Para darmos os nomes foi necessário primeiro fazermos um mapeamento deles e das categorias de cada um em uma planilha. Como na Figura 12 adiante.

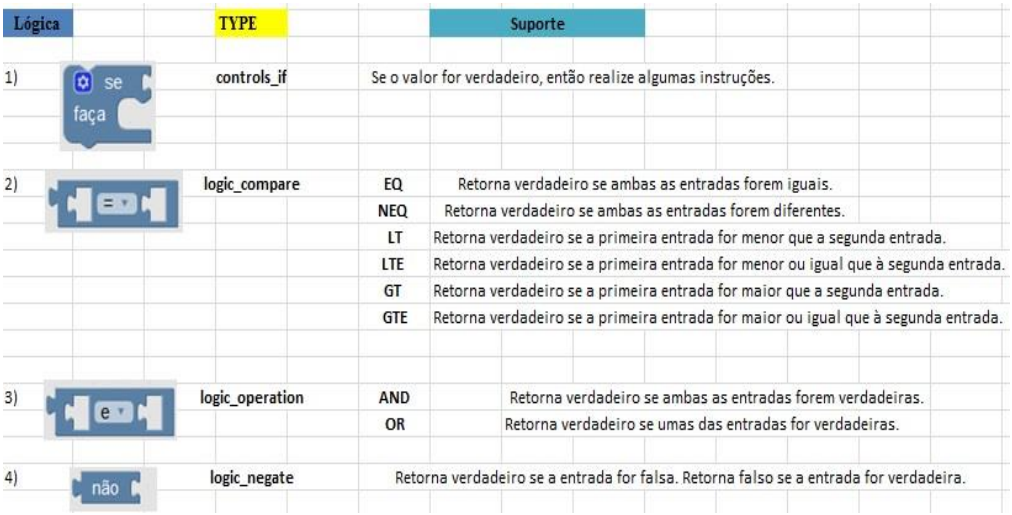


Figura 12 - Mapeamento do Blockly

Fizemos anotação das suas descrições, junto a quantidade de vezes que o bloco é utilizado por meio de um contador, e adicionamos uma espécie de ajuda, onde cada bloco possui um link próprio que leva até a página do Mozilla Developer Network (MDN).

Figura 13 - Faceta 'Suporte'

O portal MDN fornece informações sobre os comandos de programação referente a JavaScript, assim o usuário poderá associar o comportamento de uma LP visual à sintaxe de uma LP textual. A Figura 13 exibe a faceta suporte dentro de um exemplo de programa para demonstração, repare como como fica representado os blocos por nome, a quantidade de vezes que ele foi utilizado, junto a sua definição e um hiperlink para melhor aprendizado que leva a página da Figura 14, a ajuda do bloco ‘Repetição’ na página do MDN.

Ela vem após o usuário verificar a faceta Instruções, onde esta possui um hiperlink para dentro da faceta suporte para que o usuário possa compreender melhor o comando e tirar dúvidas dentro do MDN e assim ter o seu primeiro contato com a LP visual JavaScript.

Declaração for

Um laço for é repetido até que a condição especificada seja falsa. O laço for no JavaScript é similar ao Java e C. Uma declaração for é feita da seguinte maneira:

```
for ([expressaoInicial]; [condicao]; [incremento])
  declaracao
```

Quando um for é executado, ocorre o seguinte:

1. A expressão `expressao Inicial` é inicializada e, caso possível, é executada. Normalmente essa expressão inicializa um ou mais contadores, mas a sintaxe permite expressões de qualquer grau de complexidade. Podendo conter também declaração de variáveis.
2. A expressão `condicao` é avaliada. caso o resultado de `condicao` seja verdadeiro, o laço é executado. Se o valor de `condicao` é falso, então o laço terminará. Se a expressão `condicao` é omitida, a `condicao` é assumida como verdadeira.
3. A instrução é executada. Para executar múltiplas declarações, use uma declaração em bloco ({ ... }) para agrupá-las.
4. A atualização da expressão `incremento`, se houver, executa, e retorna o controle para o passo 2.

Figura 14 - Ajuda do 'Repetição' associado ao comando 'Declaração for' na página do MDN

3.1.3 FACETA ‘TAGS’

Esta faceta foi criada para representação da quantidade de vezes que os blocos foram utilizados por meio de um gráfico Treemap. Variando em tamanho, maior o retângulo para os mais utilizados e menor para os menos utilizados, e em cores para referenciar as categorias dos blocos. Partimos do princípio que esta faceta sirva como uma espécie de resumo do programa que o usuário desenvolveu. Veja na Figura 15 um exemplo da faceta Tags, repare que quanto mais vezes o bloco for utilizado maior é a área do retângulo referente a ele.

A Faceta Tags vem após a faceta ‘Suporte’, esta última possui um hiperlink que trás para dentro da Tags para que o usuário possa ver o resumo dos blocos utilizados e pensar em uma possível otimização do seu código.

Figura 15 - Exemplo da faceta Tags

3.1.4 FACETA ‘CÓDIGO JS’

Por última a faceta ‘Código JS’. Ela representa uma versão do programa em código JavaScript. Colocando finalmente o aprendiz em contato com um código de LP textual. Preservamos características dos comandos do Blockly, como as cores dos blocos para o usuário poder encontrar mais facilmente o comando equivalente ao bloco que ele utilizou. Fazendo assim a transferência de conhecimento para o JavaScript de uma forma mais prática. Também destacamos comandos específicos da linguagem que não aparecem no Blockly, como por exemplo a declaração de variáveis. Veja na Figura 16 um exemplo de como ficou a faceta ‘Código JS’ e como as cores das linhas de comando fazem referência à cor do bloco equivalente a elas.

Figura 16 - Exemplo de programa na faceta Código JS

3.2 TESTE COM USUÁRIOS – PARTE I

Para uma avaliação qualitativa do PoliFacets-Blockly I, foi realizado um teste com um grupo 5 usuários aprendizes de programação do curso de Sistemas de Informação na Universidade Federal do Pará. Todos estes iniciando o curso e vivenciando os primeiros contatos com disciplinas de programação. Durante as aulas eles tiveram contato com a ferramenta de LP visual Scratch. Para garantir a anonimização dos participantes, eles foram

identificados como P1, P2, P3, P4 e P5. Para visualizar o roteiro do teste consulte o Apêndice – Roteiro do teste 1.

3.3 RESULTADOS - PARTE I

Para o estudo dos resultados utilizamos métodos exploratórios com avaliação qualitativa dos dados coletados. Buscamos compreender e interpretar as opiniões e expectativas de cada participante com base nas respostas provenientes do questionário aplicado.

Também analisamos os códigos criados dos programas dos participantes em busca das evidências que mostrassem compreensão dos usuários com os conceitos da LP textual JavaScript enquanto programavam na LP visual Blockly. Um dos pontos positivos foi que quatro dos cinco participantes conseguiram construir um programa funcional nessas duas LP's. Por meio da nossa análise foi possível dividir em três categorias de significados, são eles: compreensão das facetas e suas ligações, reconhecimento de colaboração das facetas na compreensão do programa em JavaScript e análise dos programas em JavaScript.

3.3.1 COMPREENSÃO DAS FACETAS E SUAS LIGAÇÕES

Por meio das respostas do questionário de avaliação pelos participantes, podemos considerar que a ideia geral das facetas foi bem compreendida. Observe como P1, definiu o que é uma faceta e quais os objetivos delas no Blockly, P1 as definiu da seguinte forma: "a*s facetas são guias que nos ajudam a entender o funcionamento de cada bloco, enumeram os mais usados, dá nome a eles, e nos mostra o código que eles representam no JavaScript [...] na faceta do 'Código JS' são mostrados quais códigos estão faltando para que o programa escrito seja executado corretamente*". Notamos que P1 compreendeu as facetas corretamente e que elas o ajudaram na transferência de conhecimento para o JavaScript. O participante P2 ao responder a mesma questão diz que "*facetas são 'abas' que têm o objetivo de auxiliar o usuário no aprendizado de lógica de programação e na implementação em uma linguagem real*" percebemos que o participante P2 se referiu a JavaScript como uma linguagem real, mostrando assim entendimento que esta é diferente de uma LP de ensino visual, sendo assim esta oferece muito mais possibilidades.

Ao perguntarmos como os participantes poderiam descrever a experiência de explorar as facetas do Blockly, o participante, P3 descreve então: "*as facetas possuem um visual familiar em abas e apresentam dados de forma clara, utilizam ferramentas que facilitam a interpretação do algoritmo. Uma das ferramentas que utilizei muito foram os*

hiperlinks que levam a explicação mais detalhada do comando em JavaScript". Ele entedeu as ligações entre as facetas como 'ferramentas', e ressaltou a importância delas para a melhor compreensão do seu programa. Quando respondeu a mesma questão, P5 disse "*extremamente satisfatória, pois em certas ocasiões, em que o programador acaba se perdendo em alguns comandos, as facetas garantem um mapeamento do programa que está sendo feito*". Ele compreendeu a ideia de que as facetas 'descontroem' os programas e os apresenta de forma mais detalhada. P5 comentou o que achou da faceta 'Instruções': "*com os blocos usando a linguagem natural essa compreensão aumenta muito*". Esta era exatamente intenção que queríamos proporcionar por meio da faceta ao ponto de vista dos aprendizes.

Sobre a faceta 'Tags' ao pedirmos a opinião dos participantes, P4 diz: "i*mportante para quem faz grandes programas, pois mapeia os comandos. Muito útil*". Ele chama atenção para a importância da faceta nos casos onde os existam programas extensos e que de alguma forma possam ser diminuídos ou seja preciso fazer melhorias, por meio da informação da quantidade de cada bloco (instrução) essa reflexão se torna mais fácil. Para a mesma questão, a resposta de P2 foi: "*achei um pouco desnecessária, visto que ela só mostra a quantidade de blocos utilizados*". De posse dessa resposta, P2 não conseguiu ter a mesma visão sobre a faceta 'Tags' que P4, provavelmente porque o exemplo utilizado no teste não ressalta o valor da faceta 'Tags', pelo seu nível de complexo ser baixo. Percebemos que esta faceta exige um maior nível em sua complexidade para melhor exploração das informações e consequentemente compreensão por parte dos usuários de sua importância.

P5 faz um comentário muito interessante sobre a faceta 'Tags' ele diz: "*a Tags ajuda no controle da quantidade de blocos que é usado no programa, porém ela acaba sendo um pouco confusa de se entender por conta dos espaços que ele usa para simbolizar essa quantidade*". Sobre estas duas observações negativas da faceta 'Tags', fizeram nos questionar a escolha do treemaps como modelo gráfico para visualização da quantidade de blocos. Talvez um outro modelo de gráfico mais comum para representação de quantidade fosse mais eficiente em atrair a atenção dos aprendizes. Esse tipo de visualização da informação não é popular, talvez por isso esta rejeição pelos nossos participantes ao treemaps.

Sobre a faceta 'Suporte' P1 explica: "*essa faceta é uma ótima ferramenta para a compreensão dos blocos, todas as informações contidas nos links disponibilizados são válidas*". O que nos leva a interpretação de que ele visitou os links de ajuda para o MDN, e que mesmo apresentando uma linguagem diferente da linguagem visual colaborou para a compreensão das funcionalidades dos blocos.

Em contrapartida P1 diz: "*deveria ter mais instruções próprias ao invés de simplesmente terceirizar toda a informação*". Ele sugere que a faceta apresente mais informações dos próprios blocos. Esse comentário nos fez pensar na possibilidade de disponibilizar dois links de ajuda, um sobre o comportamento dos blocos no próprio Blockly, como uma Wiki e o outro para a página do MDN com a sintaxe JavaScript.

Por último nesta categoria a faceta 'Código JS' P2 diz: "*o blockly é mais interativo que o JavaScript e essa faceta vem para fazer esse link (com JavaScript), que consegue fazer de forma suave*". Ou seja, o participante pôde compreender o objetivo dela, que é construir uma ligação entre as duas linguagens, da visual para a textual, na tentativa de promover a transferência do conhecimento adquirido de uma linguagem para a outra.

3.3.2 RECONHECIMENTO DE COLABORAÇÃO DAS FACETAS NA COMPREENSÃO DO PROGRAMA EM JAVASCRIPT

Esta categoria aborda em como as facetas puderam ajudar na compreensão dos programas em JavaScript P1 diz: "*as facetas ajudam de forma didática a entender cada bloco usado (no Blockly) em JavaScript, principalmente para quem é iniciante e está diante desta ferramenta pela primeira vez*". Ele resume que por meio das facetas foi possível compreender a lógica de programação dos blocos também em JavaScript, e ressalta ainda a importância deste tipo de estratégia no aprendizado de uma linguagem de programação textual.

P3 também explica que as facetas possibilitaram a compreensão da lógica do seu programa em blocos, assim foi mais fácil para transferi-la para a outra linguagem. Ele diz: "*as facetas foram positivas da seguinte forma: auxiliaram o entendimento de cada etapa e comando do programa, facilitando assim a aplicação da lógica do mesmo a outra linguagem*". Isto também nos leva a interpretar que o método utilizado pelas facetas de "dividir para conquistar" no programa, foi aproveitado por esse participante para que pudesse ter um maior entendimento sobre o que ele desenvolveu.

Ao responder a questão sobre como as facetas ajudaram de alguma forma a entender o JavaScript, o participante P2 diz: "*me ajudou a entender melhor as funções para transferir a lógica para a sintaxe da linguagem (JavaScript)*". Esta resposta nos leva a crer que ele se refere as funções dos blocos na LP Blockly, e que assim como para P1 e P3 compreende-las, foi essencial para a transferência de conhecimento para a LP JavaScript.

Na pergunta "qual a relação entre os blocos no Blockly com as instruções de código JavaScript", P5 diz: "*possuem as mesmas categorias, onde a lógica aplicada no Blocky foi a mesma aplicada no JavaScript mudando apenas a sintaxe própria da linguagem JavaScript*".

Percebemos que que ele conseguiu assimilar o efeito que desejavamos provocar separando em cores as categorias dos blocos e também no código JavaScript da faceta ‘Código JS’.

Na pergunta da opinião sobre a faceta ‘Código JS’ P4 diz: “*essa faceta também é uma ótima ferramenta para a compreensão do programa que está sendo feito, pois ele auxilia no que precisa para que este mesmo programa funcione (em JavaScript)*”. O comentário dele faz referência aos elementos destacados que complementam a LP JavaScript, e que não aparecem no Blockly. Por meio dessa faceta o participante, P4, pôde perceber que o programa dele precisava de alguns comandos adicionais em JavaScript para que pudesse funcionar

3.10.3 ANÁLISE DOS PROGRAMAS EM JAVASCRIPT

Finalmente ao chegarmos na parte do teste onde os participantes tiveram que trabalhar com a LP JavaScript, pedimos a eles que comentassem os seus códigos. Nós adotamos essa estratégia levando em consideração que eles poderiam apenas copiar o código oferecido pela faceta ‘Código JS’. Os comentários nos trouxeram provas mais claras de que os participantes compreenderam o que determinado trecho do programa faz, mesmo sem nunca terem tido contato com esta LP JavaScript antes. Mostrando que foi possível fazer transferência de conhecimento.

Observe a captura de tela dos programas desenvolvidos pelos participantes nas figuras (Figura 17, Figura 18, Figura 19, Figura 20 e Figura 21. Os participantes comentaram grande parte dos seus códigos. Todos eles comentaram o comando de declaração de variáveis, isto indica que eles compreenderam tal necessidade que não existe no Blockly, e era destacada pela faceta ‘CódigoJS’. É possível observar nos comentários de P1 e P3 Figura 17-linha 5, Figura 21-linha 7, respectivamente, que eles explicam o laço ‘for’ evidenciando que existe uma variável contadora, junto a uma condição, que precisa ser declarada e inicializada previamente, aspectos que também são evidenciados na mesma faceta.

```
1  var impar,Par,A; // declaração de variáveis
2  impar=0;//atribuição de variáveis
3  Par=0;
4  var repeat_end = 5;//declaração da variável contadora
5  for (var count = 0;count < repeat_end;count++) {// inicia o laço de repetição até que
6  //count seja menor que repeat_end
7    A=parseFloat(window.prompt('Digite um numero'));// pede um número e atribui a variável A
8    if (A % 2 == 0) { // verifica a condição da expressão
9      window.alert('é par');//exibe a mensagem na tela
10     Par=Par+1;// soma a própria variável par + 1
11   } else {
12     window.alert('é impar'); //exibe a mensagem na tela
13     impar=impar+1;// soma a própria variável impar + 1
14   }
15 }
16 //exibe os resultados de quantidades par ou impar
17 window.alert('Existem ');
18 window.alert(Par);
19 window.alert('Numeros pares');
20 window.alert('Existem ');
21 window.alert(impar);
22 window.alert('Numeros impares');
```

Figura 17 - Código do P1 em JavaScript

P1 conseguiu construir corretamente seu programa, como solicitado na tarefa. Observamos uma possível apropriação da linguagem utilizada na mensagem que escolhemos apresentar na página do MDN. Percebemos que ele utilizo u em seu comentário sobre o comando ‘if’ na linha 8, ele usa as palavras ‘condição’ e ‘expressão’, estas mesmas palavras são encontradas na página fornecida do MDN para explicação do comando ‘if’, você encontra essa página pelo link do bloco ‘Se verdadeiro’. Uma evidencia da ajuda oferecida a ele pela faceta ‘Suporte’. Isto nos fez considerar que ele utilizou a ajuda desta feceta e compreendeu o seu conteúdo.

Observe que P2 conseguiu realizar a tarefa nas duas linguagens. Todas as linhas de código do seu programa foram comentadas corretamente. Pode-se observar também traços da ajuda de apropriação da linguagem da mensagem de ajuda da faceta suporte. No comentário dele Figura 18-linha 9, explicando o comando de leitura dos números, ele usa o termo ‘caixa de conversa’, o mesmo presente na ajuda do bloco ‘Obter entrada do usuário’ na faceta ‘Suporte’. Essas referências a linguagem do Blockly sugerem características do processo de transição para o JavaScript.

Outro participante que saiu muito bem foi o P3, ele conseguiu realizar todas as fases da tarefa. O fato interessante deste participante foi que ele preferiu usar uma sintaxe diferente no seu código. Utilizando apenas o ‘alert’ nos comandos das linhas 9, 17, 23, 29 e 31 da Figura 19. Esta opção é possível na linguagem JavaScript, porém diferente do que foi apresentado a ele nas facetas ‘Código JS’ e ‘Suporte’. Talvez pela possibilidade dele ter ido além da mensagem de ajuda da faceta ‘Suporte’, e pesquisou outros vários exemplos de programas contidos em diferentes páginas do guia JavaScript oferecido pelo MDN.

```javascript
var variavel,contador_par,contador_impar,Vari_C3_A1vel;
// Esta é a parte onde fica a declaração de variáveis


var repeat_end = 5;
//  Esta é a estrutura de repetição, que pode repetir n vezes
for (var count = 0;count < repeat_end;count++) {
  variavel=parseFloat(window.prompt('Digite um número inteiro e positivo:'));
 // caixa de conversa que pede para o usuário que escreva um valor
  if (variavel % 2 == 0) {
 // Esta é a estrutura condicional
    window.alert(String(variavel) + String(' é um número par!'));
    // se esta condição for verdadeira então ele exibe a mensagem informando que o número é par
  } else {
    window.alert(String(variavel) + String(' é um número impar!'));
    // se esta condição for verdadeira então ele exibe a mensagem informando que o número é impar
  }
}
if (Vari_C3_A1vel % 2 == 0) {
// esta é uma estrutura condicional
  contador_par=contador_par+1;
  // se o resto da divisão do número por 2 for igual a zero, então ele é par
} else if (Vari_C3_A1vel % 2 == 1) {
// se o resto da divisão do número por 2 for igual a um, então ele é impar
  contador_impar=contador_impar+1;
  // este é um contador, que mostra quantos números são pares e impares
}
```

Figura 18- Código comentado de P2 em JavaScript

```javascript
var par, impar, num, repeat_end; //Declara variáveis
par=0;
//Inicializa a variável par
impar=0;
//Inicializa a variável impar
repeat_end=5;
//Inicializa a variável repeat_end (fim do contador)

alert('Este programa mostra se o valor informado é par ou ímpar');
//imprime a mensagem
for(var count=0;count<repeat_end;count++){
//Conta de count ate repeat_end
num=prompt('informe um valor')
//pede o valor
if (num % 2 == 0){
// condição para par
alert(String(num)+String(' é par'));
//imprime a mensagem
par=par+1;
//contador para par
} else {
//senão
alert(String(num)+String(' é impar'));
//imprime mensagem
impar=impar+1
//contador para impar
}
}
alert(String('A quantidade de pares é ') + String(par));
//informa o numero de pares
alert(String('A quantidade de ímpares é ') + String(impar));
//informa o numero de impares
```

Figura 19 - Código em JavaScript de P3

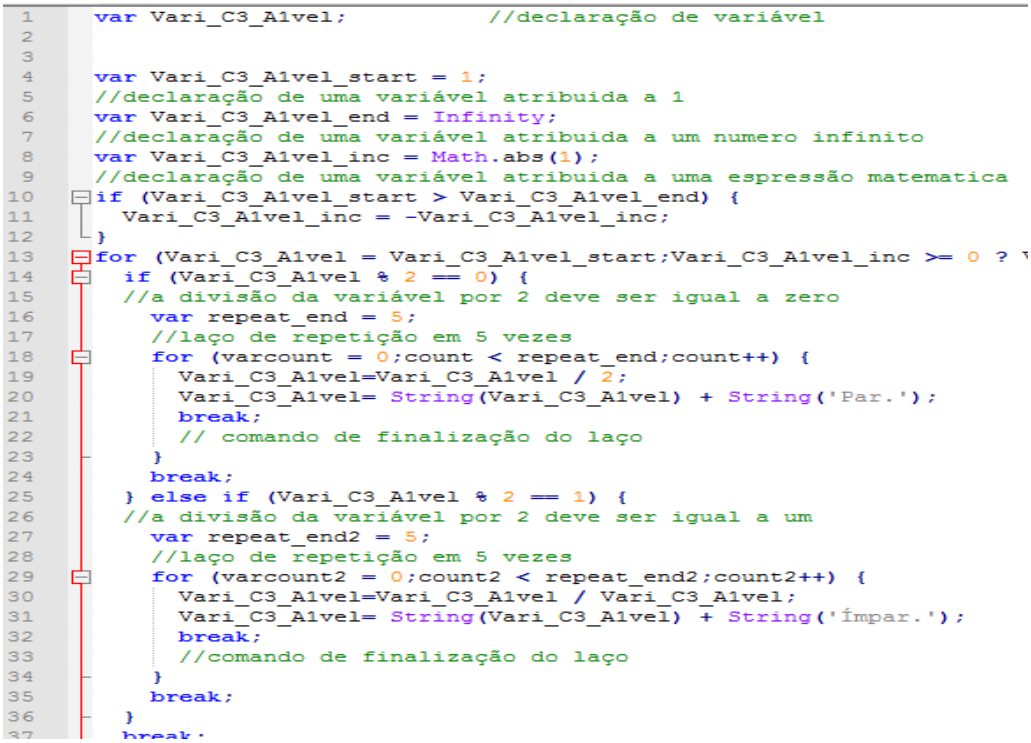
O participante P4 também conseguiu finalizar corretamente a tarefa proposta, e discorreu comentários em seu código onde podemos analisar a transferência de conhecimento entre a linguagem do Blokly e a linguagem JavaScript. Ele se referiu aos comandos como blocos, nas linhas 14 e 20 da Figura 20. Inclusive, usa termos específicos da faceta ‘Instruções’ exibida na Figura 11, como por exemplo ‘imprimir bloco’. Tal fato nos faz interpretar que ele está no processo de transição entre as duas linguagens, fazendo associações da lógica dos blocos com a dos comandos em JavaScript.

```
2   //O PoliFacets está fazendo um trabalho incrível ao conseguir transformar em linguagem javascript
3
4   //Essa ferramenta abre um leque de possibilidades criativa gigantesco, pois, é uma ferramenta a mais para programar
5   //facilitando ainda mais na hora da programação, conseguindo visualizar de uma outra forma os códigos digitados.
6
7   var numero, contpar, contimpar; // declaração de variável
8
9   //valores das vasriáveis
10  numero=0;
11  contpar=0;
12  contimpar=0;
13  var repeat_end = 5; // variavel da repetição
14  for (var count = 0;count < repeat_end;count++) { // bloco laço de repetição contando ate 5
15      numero=parseFloat(window.prompt('Digite um numero para verificar'));// usado para pedir um número ao usuário
16    if (numero % 2 == 0) {        // bloco de comandos(condições if...else)
17      window.alert('é par');//imprimir bloco
18      contpar=contpar + 1;
19    } else {
20      window.alert('é ímpar');  //imprimir bloco
21      contimpar=contimpar + 1;
22    }
23  }
24
25  window.alert('Numeros par:'); // imprimir
26  window.alert(contpar); //imprimir
27  window.alert('Numeros impar:'); // imprimir
28  window.alert(contimpar); //imprimir
```

Figura 20 - Código em JavaScript de P4

De todos os participantes apenas o P5 não conseguiu construir um programa funcional em nem no Blockly e nem do JavaScript. O programa dele é exibido na Figura 21. Ele usou uma série de comandos combinados sem lógica alguma, como a adição de três laços de repetição do tipo ‘for’, dois comandos condicionais do tipo ‘if’ e um método ‘math’ para obter o valor absoluto de um número. Porém, mesmo assim conseguiu compreender alguns comandos da linguagem JavaScript. Por exemplo, a declaração de variáveis na linha 1 e o laço da linha 17.

O caso do participante P5 reforça ainda mais o objetivo do nosso trabalho, claramente ele é um usuário que possui dificuldades com a lógica da linguagem de programação, exemplo do que acontece com muitas pessoas e isso nos fez buscar mais formas de ajudar esse tipo de usuários. Para que elas possam fazer a transferência de conhecimento de uma LP para outra. Ainda sim, P5 conseguiu identificar alguns comandos lógicos em JavaScript e isso já será um forte aliado em um momento futuro de aprendizado das regras sintáticas de LP textual.



```
var Vari_C3_A1vel;           //declaração de variável


var Vari_C3_A1vel_start = 1;
//declaração de uma variável atribuida a 1
var Vari_C3_A1vel_end = Infinity;
//declaração de uma variável atribuida a um numero infinito
var Vari_C3_A1vel_inc = Math.abs(1);
//declaração de uma variável atribuida a uma espressão matematica
if (Vari_C3_A1vel_start > Vari_C3_A1vel_end) {
  Vari_C3_A1vel_inc = -Vari_C3_A1vel_inc;
}
for (Vari_C3_A1vel = Vari_C3_A1vel_start;Vari_C3_A1vel_inc >= 0 ? 
  if (Vari_C3_A1vel % 2 == 0) {
  //a divisão da variável por 2 deve ser igual a zero
    var repeat_end = 5;
    //laço de repetição em 5 vezes
    for (varcount = 0;count < repeat_end;count++) {
      Vari_C3_A1vel=Vari_C3_A1vel / 2;
      Vari_C3_A1vel= String(Vari_C3_A1vel) + String('Par.');
      break;
      // comando de finalização do laço
    }
    break;
  } else if (Vari_C3_A1vel % 2 == 1) {
  //a divisão da variável por 2 deve ser igual a um
    var repeat_end2 = 5;
    //laço de repetição em 5 vezes
    for (varcount2 = 0;count2 < repeat_end2;count2++) {
      Vari_C3_A1vel=Vari_C3_A1vel / Vari_C3_A1vel;
      Vari_C3_A1vel= String(Vari_C3_A1vel) + String('Ímpar.');
      break;
      //comando de finalização do laço
    }
    break;
  }
  break;
```

Figura 21 - Código JavaScript de P5

4 POLIFACETS-BLOCKLY II

Como podemos ver no PoliFacets-Blockly I, a nossa ferramenta de apoio a transferência de conhecimento da lógica de programação por meio de uma LP visual para uma LP textual, com o auxílio do modelo PoliFacets, os dados que conseguimos coletar no primeiro teste com usuários, nos indicaram resultados positivos para colaboração da transição entre estas duas linguagens. De modo que levamos em consideração todas as sugestões que os usuários fizeram durante a avaliação dos pontos positivos e negativos para a continuação e melhoria do nosso trabalho no desenvolvimento do PoliFacets-Blockly II.

Esses dados nos fizeram atentar para melhorias e para a criação de novas facetas de significados de programas para colaboração de transição entre a LP visual para uma LP textual. Os participantes do primeiro teste escolheram as facetas ‘Instruções’ e ‘Suporte’ como as melhores e que estas foram as que mais ajudaram para a transferência de conhecimento. Com destaque para a faceta ‘Instruções’, os participantes tiveram uma melhor compreensão semântica por meio de uma linguagem, podendo assim fazer a transferência de conhecimento para a LP textual de forma mais suave.

Outra observação que destacamos para melhoria, foi a escolha do treemaps para representação gráfica da quantidade de bloco utilizados. Os participantes tiveram dificuldades em entender a sua importância e também criticaram a forma como esses dados são representados, tendo dificuldades para a leitura. Realmente não é comum encontrarmos esse tipo de representação gráfica.

Mais uma observação muito importante que fizemos foi a importância de termos mais de uma LP textual a ser fornecida com a ajuda das facetas, para que o aprendiz possa escolher entre mais de uma linguagem de programação e assim fazer uma melhor transferência de conhecimento da forma que ele sentir maior facilidade.

4.1 DESENVOLVIMENTO POLIFACETS-BLOCKLY II

Foram feitas melhorias no nosso protótipo e foram criadas as facetas ‘Requisitos’ e a ‘Código Python’ e fizemos também mudanças na representação gráfica da quantidade de blocos da ‘Tags’ e junto a isso o seu nome, agora chamada por ‘Chart’. Por causa dos destaque aos pontos positivos durante a avaliação dos participantes do PoliFacets-Blockly I à faceta ‘Instruções’, como a faceta que melhor ajudou na compreensão do código do programa por meio de uma linguagem natural, criamos a faceta ‘Requisito’.

Como já dito anteriormente a faceta ‘Requisito’ busca especificar o programa feito pelos aprendizes, de modo que eles possam compreender ainda mais o seu código. Ela também faz um hiperlink para a faceta que a originou, a ‘Instruções’. Queremos com esta faceta que o usuário tenha uma visão mais completa do que está sendo produzido, junto as funcionalidades do seu código e assim ajudar na transferência de conhecimento de uma LP visual para uma LP textual.

Também foi implementada a faceta ‘Código Python’ para o PoliFacets-Blockly II, a escolha da linguagem Python se deu pela sua popularidade no meio acadêmico como a melhor linguagem para introdução ao ensino de programação nas principais universidades pelo mundo como em Carnegie Mellon, MIT, Standford e muitas outras. Informação retirada do site da organização mundial de TI ‘Association for Computing Machinery’(Guo, 2014). Veja na Figura 22 como ficou o ranking das principais linguagens.

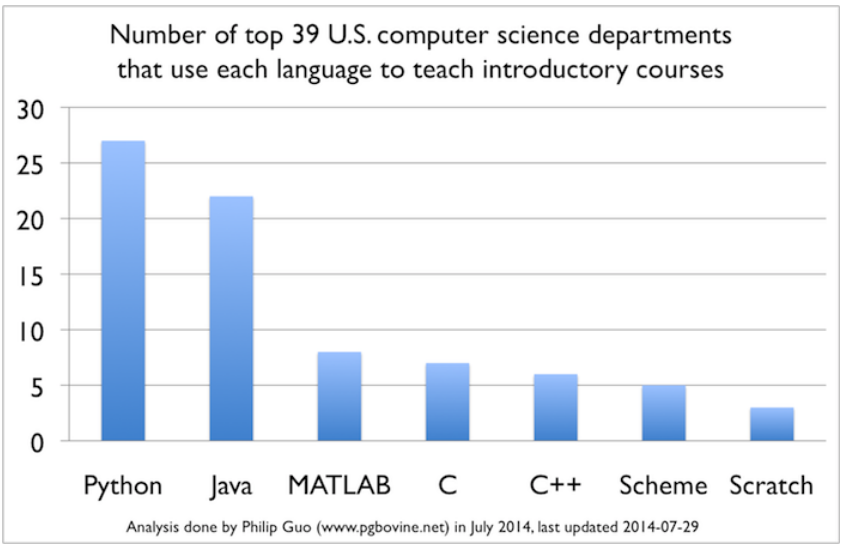


Figura 22 - Linguagens mais utilizadas para a introdução a programação nas principais universidades do mundo (Guo, 2014)

Escolhemos a linguagem Python de forma estratégica para que o aprendiz tenha acesso a uma linguagem de programação que ele possa fazer a transferência de conhecimento da LP visual para LP textual mais suavemente.

E por último, melhoramos a representação gráfica da quantidade de blocos da faceta ‘Tags’, agora não mais o modelo treemaps, pois este foi muito criticado no Teste 1. Agora para representação gráfica foi escolhido o modelo de barras, pois este é muito mais popular e usado por qualquer estudante de qualquer área de conhecimento. Preferimos também a mudança de nome para ‘Chart’, pois esta tem como principal foco visual o destaque na informação numérica da quantidade de vezes que cada bloco foi utilizado.

No Quadro 2 - Classificação das facetas do PoliFacets-Blockly II mostramos as facetas utilizadas na segunda fase do protótipo e sua classificação de acordo com o modelo PoliFacets. Durante a experiência do aprendiz, novamente, há uma ordem proposital na posição delas para que ele possa ir subindo de nível para o aprendizado de programação. A ordem das facetas ficou da seguinte forma 1) Instruções; 2) Suporte; 3) Requisito; 4) Chart e 5) Código JS e 6) Código Python. Onde as novas facetas (Requisito, Chart e Código Python) serão melhor detalhadas a seguir.

Tipos de Faceta	**Facetas do PoliFacets-Blockly II**
Facetas de Paráfrase-Reformulação	• Instruções; • Requisito; • Código JS; • Código Python.
Facetas de Paráfrase-Reescrita	• Suporte; • Chart.

Quadro 2 - Classificação das facetas do PoliFacets-Blockly II

4.1.1 FACETA ‘REQUISITO’

Esta faceta se originou da faceta ‘Instruções’, percebemos no primeiro teste com usuários que os usuários iniciantes se saíram melhor com uma linguagem mais próxima da comum. Então a faceta ‘Requisito’ busca especificar o programa produzido pelos usuários como em uma espécie de documentação de especificação de requisito, muito comum à engenharia de software, para uma compreensão ainda melhor do programa produzido. Ela vem após a faceta ‘Suporte’ para o usuário verificar a funcionalidade do seu programa e ela também também faz um hiperlink com a faceta ‘Suporte’. Observe na Figura 23 como ficou mais detalhada a funcionalidade do programa ao aprendiz por meio da faceta Requisito.

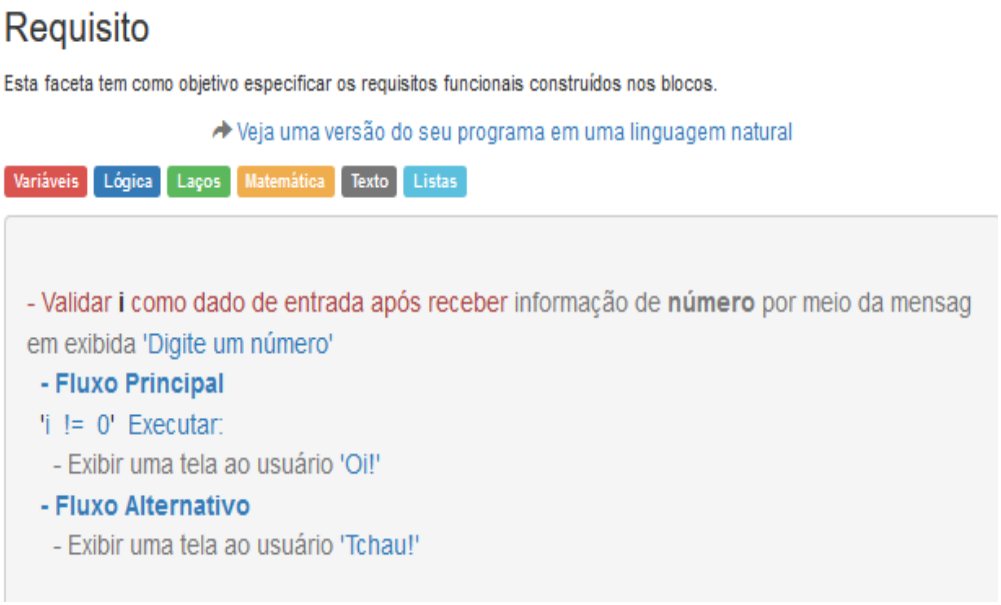


Figura 23 - Exemplo da faceta Requisito

4.1.2 FACETA 'CHART'

Esta faceta foi desenvolvida para substituir a faceta Tags por questões de design, pois no primeiro teste com usuários a faceta Tags foi muito criticada pelo seu design e espaço que ela utiliza para fazer a representação dessa quantidade. Então a faceta Chart surgiu como uma opção para melhorar essa representação de forma que o usuário consiga enxergar as informações de forma mais clara e direta. Falaremos melhor sobre essa mudança, junto a questão dos testes e resultados com usuários nos próximos capítulos. Veja como ficou essa mudança de representação gráfica na Figura 24.

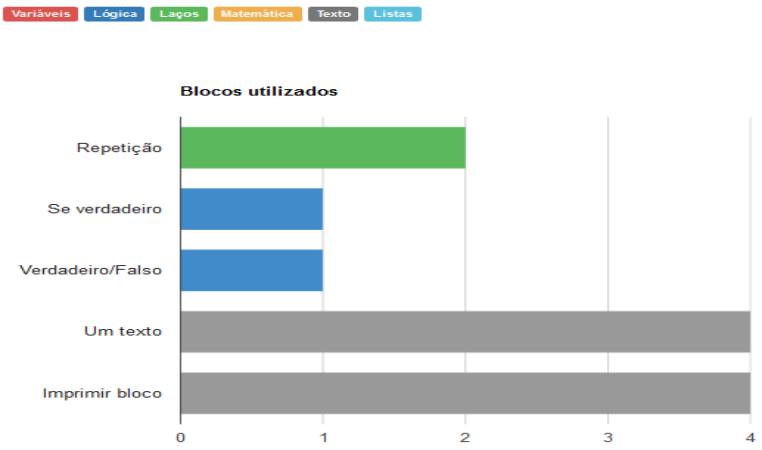


Figura 24 - Exemplo da faceta Chart

4.1.3 FACETA 'CÓDIGO PYTHON'

Esta última faceta representa uma versão do programa na linguagem Python. Ela também atenta para a questão da referência das cores para identificação dos comandos com os

blocos. Com destaque nos comandos específicos da linguagem que não fazem parte do Blockly, à exemplo a declaração de variáveis. Ela surge como uma opção a mais de LP textual ao usuário, para ele poder fazer uma escolha de qual linguagem o aprendiz consegue fazer uma transferência de conhecimento da LP visual para a LP textual mais facilmente. Não limitando assim à apenas uma única linguagem. Veja na Figura 25 como ficou a faceta 'Código Python' e atente para a questão das cores em referência aos blocos equivalentes e também ao complemento do código Python.

Figura 25 - Exemplo da faceta Código Python

4.2 TESTE COM USUÁRIOS – PARTE II

Para uma avaliação qualitativa do PoliFacets-Blockly II, foi realizado um teste maior em relação ao primeiro, este segundo teste foi feito com 20 usuários, dividido em três equipes. Sendo estudantes de diversas áreas de conhecimento como computação, administração, ciências biológicas, letras e comunicação. Eles foram divididos em dois grupos, o primeiro grupo com nove participantes sem experiência em programação e o segundo grupo com onze participantes com algum nível de experiência de programação (classificados entre iniciantes, médio e avançados), sendo a maioria iniciantes deste último grupo.

Os participantes do primeiro grupo foram identificados de P1 a P9, e do segundo grupo de P10 a P20. Foi feito um exemplo antes deles iniciarem a atividade para que eles entendessem melhor o objetivo e pudessem utilizar a ferramenta melhor. Os testes foram conduzidos com os participantes de cada grupo juntos e foi dado a eles cerca de uma hora para execução das tarefas. Para visualizar o roteiro do teste, consulte o Apêndice – Roteiro do teste 2.

4.3 RESULTADOS – PARTE II

Para o estudo dos resultados do segundo teste utilizamos os mesmos métodos exploratórios de avaliação do primeiro teste com usuários. Buscamos compreender e interpretar as opiniões, expectativas de cada participante com base nas respostas provenientes do questionário aplicado e também na observação dos testes.

Demos mais foco também em qual LP textual os usuários conseguiram fazer uma melhor transferência de conhecimento, no caso entre JavaScript e Python. Junto as evidências que mostrassem compreensão dos usuários com os conceitos da LP´s textual enquanto programavam na LP visual Blockly. Um dos pontos positivos foi que seis dos nove participantes sem experiência alguma com programação conseguiram construir um programa funcional nessas na LP visual e conseguiram fazer a transferência de conhecimento para uma das duas LP textual. Por meio da nossa análise foi possível dividir em três categorias de significados, são eles: compreensão das facetas e suas ligações, reconhecimento de colaboração das facetas na compreensão do programa em JavaScript e Python, e análise dos programas nas LP´s textual.

4.3.1 COMPREENSÃO DAS FACETAS E SUAS LIGAÇÕES

Por meio das respostas do questionário de avaliação pelos participantes, podemos considerar que a ideia geral das facetas foi bem compreendida. Observe como P1, definiu o que é uma faceta e quais os objetivos delas no Blockly, P1 as definiu da seguinte forma: “*as facetas funcionam como abas com conteúdos para instruções, dicas e referencias sobre a construção dos blocos*”. Notamos que P1 compreendeu as facetas corretamente e que elas o ajudaram na construção lógica do seu programa. O participante P9 ao responder a mesma questão diz que “*uma aba que ajuda a elaboração da resolução do problema base acompanha o dinamismo em tempo real dada a adição dos blocos*”. P9 conseguiu entender perfeitamente o significado das facetas e usou a expressão ”*dinamismo em tempo real*” para a

documentação ativa destas, mostrando que este participante entendeu muito bem o funcionamento das facetas.

Os participantes P7 e P8 classificaram as facetas, respectivamente como "*uma faceta é um aspecto que se observa em algo. Os objetivos das facetas do blockly é facilitar o aprendizado e uso de programas de programação*" e "*são abas que explicam ou demonstram determinada ação no momento de programar*" estes participantes também conseguiram compreender o comportamento e os objetivos das facetas. Observe que P7 cita que a "*faceta é um aspecto que se observa em algo*", este participante tentou expressar que cada faceta ajuda o usuário a compreender o mesmo objeto de estudo por meio de abordagens diferentes de acordo com cada categoria delas, provocando assim uma reflexão mais abrangente sobre o programa.

Já no grupo de participantes com alguma experiência em programação, o participante P16 classificou o que é uma faceta como "*facetas são abas que dão suporte a uma determinada atividade (no caso, à programação), disponibilizando conteúdo útil para a realização eficaz da mesma. Os objetivos das facetas do blockly foram: o de dar uma experiência interativa e limpa ao usuário e disponibilizar conteúdo útil que pode ser consultado a qualquer momento dando assim um suporte extra a ferramenta apresentada*". Este participante analisou corretamente as informações das facetas.

Ao perguntarmos da experiência de explorar as facetas do Blockly, o participante P8 avaliou como "*foi uma boa experiência, as facetas me ajudaram a entender como estava criando meu programa*", este participante sou fazer uso das informações geradas nas facetas para entender o seu programa, nessa linha o participante P5 respondeu a mesma questão "*ajudam a esclarecer a lógica da programação*" mostrando que elas o ajudaram a compreender a lógica de programação. Já o participante P6 atenta para a necessidade de um tutorial para ajudar a mexer na ferramenta Blockly junto a exemplos "*acredito que além das facetas, seria importante um tutorial ou mesmo que houvesse uma faceta com um exemplo semelhante ao solicitado para quando tivéssemos alguma dúvida compararíamos o que estamos fazendo com o exemplo*", talvez a sugestão de se ter um botão com uma Wiki ou uma vídeo-aula, possa ser inserido e que isto pode vim a ajudar outros aprendizes que vierem a utilizar o nosso protótipo.

No grupo dos participantes com alguma experiência em programação, o P19 opinou "*as facetas ajudaram a compreender cada bloco sem a necessidade de um professor ou tutor*" um opinião muito positiva, pois é exatamente esta a intenção que as facetas buscam passar aos usuários. O participante P16 descreveu a sua experiência como "*foi uma experiência boa,*

interessante e muito construtiva, pois além de tudo ser disponibilizado ao lado do código que é desenvolvido todas as informações referentes ao mesmo, todas as facetas (bem como os blocos) são acompanhados de explicações simples e muita intuitividade".

O participante P17 e P18 citaram como pontos positivos, respectivamente, "*informações que ajudam a pessoa se guiar pelo sistema; facilitam o uso tanto para quem já tem experiência com programação, quanto para leigos*" e "*facilita a otimização de códigos e ajuda a compreender o processo de programação*" notamos que o participante entendeu o objetivo da faceta 'Chart' ao falar sobre a 'otimização de códigos', pois esta busca justamente atentar ao usuário sobre essa necessidade.

Sobre a faceta 'Chart', outro participante que compreendeu perfeitamente a sua importância foi o P16, ele diz que "*é uma faceta que ajuda a ter uma visão geral sobre o código. Pode ser utilizada inclusive para otimizá-lo*", nessa mesma linha de ideia outro participante que também compreendeu a sua importância foi o participante P15 "*é boa, visto que, é uma forma do programador avaliar seus códigos e gerar métodos de otimização e correção, além da visualização ser bem amigável. Não creio que precisem melhoras aqui*" fazendo nos ter convicção de que a mudança do treemap para o gráfico em barras foi uma ótima ideia.

Dentro do grupo dos participantes com pouca experiência em programação o participante P6 respondeu sobre o que achou sobre a faceta 'Chart' "*nao entendi sua utilidade*", assim como ele os participantes P3, P4 e P5 não entenderam a sua funcionalidade. Pois, esta faceta requer um pouco mais de conhecimento de programação para entender a sua complexidade. Em contra partida escreveu "*percebi que ela é como o "rol" na matemática. Achei muito bacana, principalmente se eu fizer um programa muito grande*". Mostrando que mesmo este assimilou bem a sua importância em casos de problemas mais complexos como o de programas muito grandes, como ele mesmo abordou.

Na questão sobre o que os participantes acharam da faceta 'Instruções', esta foi muito elogiada nos dois grupos. O participante P9 respondeu que "*a faceta de instruções expõe de forma clara o código*" ele quis dizer que foi possível ter uma melhor compreensão dos comandos em uma linguagem natural. O participante P16 comentou "*achei muito interessante a ideia de linguagem natural, isso ajuda muito mesmo aqueles que estão iniciando no mundo da programação. Nesta faceta creio que nada precise ser melhorado.*" É exatamente essa intensão da faceta 'Instruções' ajudar o aprendiz de programação a compreender a lógica para uma futura transferência de conhecimento para uma linguagem de programação real.

Alguns participantes sugeriram algumas melhoras a serem feitas nesta faceta como foi o caso do P1 que disse "p*oderia descrever melhor as categorias, por exemplo para uma pessoa leiga em programação pode não conhecer a definição de variável, ou laço*". Não cabe a faceta Instruções este tipo de informação, porém a ideia de se ter informações a cerca das categorias para um melhor entendimentos às pessoas leigas é uma boa sugestão.

Ao perguntarmos a opinião sobre a faceta 'Suporte', o participante P9 respondeu "e*la é importante, principalmente para aqueles que estão aprendendo a programar*" é exatamente esta intenção que nós queremos passar ao aprendiz, para que este possa aprender mais sobre os comandos de programação. No grupo dos participantes com experiência em programação, o participante P15 respondeu acerca desta faceta como "a *escolha do link é muito boa, visto que aquele é uma das principais fontes para programadores, fora que lá eles acompanham as atualizações, além disso é muito interessante a associação entre as operações utilizadas e os links disponíveis. Não creio que tenha algo que deve ser melhorado*" podemos notar que este participante explorou bem as informações passadas por ela e que a escolha pelo MDN como fonte foi uma ótima escolha.

No primeiro teste tanto a faceta 'Instruções' quanto a 'Suporte' foram muito elogiadas pelo seu conteúdo buscar ajudar aprendizes com pouca ou sem nenhuma experiência em programação e podemos confirmar neste segundo teste as suas aceitações. Nesta mesma linha de raciocínio colocamos a faceta 'Requisito' para especificar de forma mais detalhada o programa criado pelos usuários. Nesta linha de raciocínio o participante P7 a definiu como "*Autoexplicativa*", o participante P9 opinou "*a faceta de requisito expõe objetivamente o que quer se mostrar*" e nesta mesma linha de raciocínio o participante P14 e P16 responderam respectivamente "e*la oferece uma função muito útil pra quem deseja documentar seu código e apresentar para outras pessoas*" e "*o interessante aqui é que dá viabilidade para documentar o código que é desenvolvido, assim, será muito mais simples disponibilizar o material para pessoas estudarem o trabalho além de construir a contribuição com o código*". É exatamente o que esta faceta busca, especificar o programa de forma que ele possa ser lido de forma muito mais suave e detalhada, ajudando assim em uma maior compreensão.

Por último nesta categoria os participantes quanto as facetas de LP´s textual 'Código JS' e 'Código Python'. O participante P8 respondeu sobre a faceta 'Código JS' "a*chei esclarecedora, pode melhorar na linguagem que é em inglês e nem todos entendem a língua*" podemos perceber que apesar da sua sugestão de ser escrito em português que não é possível, ele demonstrou uma certa compreensão do que havia sido analisado por ele dentro das facetas

ao definir a faceta como "*esclarecedora*". Nesta mesma linha de raciocínio o participante P7 definiu a faceta 'Código Python' como "*bem prática*".

4.3.2 RECONHECIMENTO DE COLABORAÇÃO DAS FACETAS NA COMPREENSÃO DO PROGRAMA EM JAVASCRIPT E PYTHON

Esta categoria aborda em como as facetas puderam ajudar na compreensão dos programas em JavaScript e Python P2 disse que as facetas foram "*boas para ajudar a entender a relação entre os blocos e os comandos em JS e Python*" desta forma notamos alguma transferência de conhecimento da LP visual para as LP´s textual, pois ele ainda cita a relação dos blocos e comandos nas linguagens em Código JS e Python, o participante P2 conseguiu reconhecer perfeitamente e obteve sucesso na transferência de conhecimento.

O participante P1 também conseguiu reconhecer esta colaboração "a*lem das cores, a correlação fica no entendimento do funcionamento do comando*" ele atentou para a referência das cores dos comandos nos códigos com as cores dos blocos e obteve uma melhor compreensão utilizando as facetas para o entendimento dos comandos. Outro participante que teve esse mesmo resultado foi o P6 que disse "*melhor visualização e entendimento de como esses códigos funcionam*" ou seja, este participante conseguiu enxergar e compreender melhor as LP´s textual por meio da ajuda das facetas. O participante P8 também conseguiu um bom resultado ao dizer que "e*las me ajudaram a compreender melhor como desenvolver os comandos*".

O participante P16 reconheceu a colaboração das facetas ao falar da relação das cores na faceta 'Código JS' "*essa faceta foi uma das melhores, visto que ele monta o código para o usuário, além disso, ela ajuda muito no entendimento dos blocos e da ferramenta como um todo, além de ter um ótimo apelo visual (identificar as linhas com as cores dos blocos)*" e o participante P18 ao falar da faceta 'Instruções' também identifica a importância dela na compreensão dos programas em LP´s textual dizendo "*achei muito interessante a ideia de linguagem natural, isso ajuda muito mesmo aqueles que estão iniciando no mundo da programação*".

O participante P15 coloca muito bem o reconhecimento das facetas na compreensão da relação dos blocos com os códigos em JavaScript e Python ao falar que as facetas o ajudam a "*ter certeza de que o que está sendo desenvolvido nos blocos está de acordo com que eu escreveria em uma IDE comum*". Ou seja, este participante que possui conhecimento de programação atentou para o caminho "inverso" para se certificar se as informações nas facetas

estão de acordo com o que era feito nos blocos. Reconhecendo que estas podem sim ajudar na compreensão das LP´s textuais.

4.3.3 ANÁLISE DOS PROGRAMAS EM JAVASCRIPT E PYTHON

Ao final do teste foi pedido aos participantes que lessem como ficou o código dos programas que eles criaram. Escolhessem qual linguagem eles conseguiram compreender melhor o código, fazendo transferência de conhecimento da LP visual para as LP´s textuais das facetas ‘Código JS’ e ‘Código Python’. Após isso, tentassem reescrever os códigos do programa que eles construíram de forma que eles consultassem o mínimo possível as facetas ‘Código JS’ e ‘Código Python’. Ao grupo de iniciantes foi permitida sempre a consulta, já que eles estavam tendo contato pela primeira vez com a sintaxe e a semântica destas linguagens.

E por fim pedimos para serem feitos comentários nos comandos em que eles conseguiram reconhecer a relação com os blocos do Blockly. Esta estratégia foi adotada para podermos analisar a linguagem escolhida com base no perfil de cada participante e, principalmente, pudéssemos analisar a ocorrência da transferência de conhecimento gerada no teste por meio das facetas.

Todos os participantes conseguiram compreender a necessidade de declarar as variáveis, mesmo esta não sendo necessária no Blockly. Nós colocamos em destaque nas nossas facetas de LP textuais e também percebemos que a maioria dos participantes compreenderam a relação do comando ‘For’ com o bloco ‘Repetir’.

```
1   Nome = None
2   Frutas_preferidas = None #definir variável
3
4   def text_prompt(msg): #definição para comandos de textos
5     try:
6       return raw_input(msg)
7     except NameError:
8       return input(msg)
9
10
11  for count in range(int(5)): #Repetir até
12    Nome = text_prompt('Informe o nome do cliente') #Tela de texto
13    Frutas_preferidas = [text_prompt('fruta 1'), text_prompt('fruta 2'), text_prompt('fruta 3')]
14    #Variável onde salva a fruta preferida do cliente
```

Figura 26 - Código em Python de P5

Observe na Figura 26–Linha 4 que o participante P5 compreendeu que era necessário definir certos comandos na linguagem em Python, que não se fazem necessários no Blockly. Confirmando a nossa ideia de que os participantes entenderam a parte de “complementos” presente nas facetas de código das nossas LP´s textuais. Outro ponto interessante foi que este participante compreendeu o comando “for” como o bloco de repetição.

Outro participante sem experiência que conseguiu reescrever o seu código, foi o participante P7. Observe na Figura 27 que este não utilizou o comando de repetição e escreveu 5 vezes o mesmo comando que pedia na questão. Porém ele soube utilizar perfeitamente as instruções de Definir variável, pedir um texto ao usuário e imprimir texto. Concluímos que P7 compreendeu bem a lógica de programação e que isso já será um forte aliado para o aprendizado de comandos de laço, lógica e outros mais a frente. Outro ponto importante a se destacar seja que por ser um número consideravelmente baixo de repetição, ele não sentiu a necessidade de utilizar ou de buscar aprender este método, ou seja, talvez se fosse um problema com o número de repetições mais elevado, poderia ser que P7 atentasse para esta parte.

```
1  var cliente_1,cliente_2,cliente_3,cliente_4,cliente_5; //Declaração de Variáveis para salvar a Fruta Preferida dos Clientes.
2
3
4  cliente_1=window.prompt('Qual a fruta preferida do cliente 1'); //Telas de Pergunta ao Usuário.
5  cliente_2=window.prompt('Qual a fruta preferida do cliente 2');
6  cliente_3=window.prompt('Qual a fruta preferida do cliente 3');
7  cliente_4=window.prompt('Qual a fruta preferida do cliente 4');
8  cliente_5=window.prompt('Qual a fruta preferida do cliente 5');
9  window.alert(String('A fruta preferida do cliente 1 é ') + String(cliente_1)); //Telas com as Frutas Preferidas de cada um dos Clientes.
10 window.alert(String('A fruta preferida do cliente 2 é ') + String(cliente_2));
11 window.alert(String('A fruta preferida do cliente 3 é ') + String(cliente_3));
12 window.alert(String('A fruta preferida do cliente 4 é ') + String(cliente_4));
13 window.alert(String('A fruta preferida do cliente 5 é ') + String(cliente_5));
```

Figura 27 - Código em JavaScript de P7

Um participante dentro do grupo dos aprendizes, o P3, conseguiu concluir a tarefa de reescrever o seu código, porém sem funcionar. Observe na Figura 28-linhas 4, 5 e 7 onde estava o erro. Notamos que este participante conseguiu compreender a lógica e entender a relação dos blocos e comandos dentro do código JavaScript, apesar de ter compreendido o comando de repetição, este não compreendeu que a repetição terminaria ao se satisfazer a condição definida no código e colocou o comando de “break” relacionado ao bloco “encerra o laço”. Talvez se este participante tivesse atentado à faceta ‘Suporte’ no texto de ajuda do bloco de repetição, ele encontraria a seguinte informação “repete instruções até que a condição especificada seja falsa.” e que poderia lhe ajudar a fazer esta transferência de conhecimento.

```
1  var Nome,Fruta; //Variáveis do Bloco Definir Variáveis.
2
3
4  var repeat_end = 5; //Bloco de repetição.
5  for (varcount = 0;count < repeat_end;count++) { //Repetirá a tela "Qual seu nome?"
6    Nome=window.prompt('Qual seu nome?');
7    break; //Comando para parar de repetir a mensagem.
8  }
9  Fruta=window.prompt('Qual a sua fruta preferida?'); //Tela com a pergunta "Qual a sua fruta preferida?"
10 window.alert('Obrigado!'); //Tela com "Obrigado" para fechar o programa.
```

Figura 28 - Código em JavaScript de P3

Já o participante P9, que se destacou nos testes, atentou para essa questão da condição dentro do comando de repetição. Note que ele destacou isso nos comentários do seu código e inclusive utilizou o mesmo texto presente na faceta ‘Suporte’. Observe na Figura 29 -linha 6. Mostrando que este soube utilizar a faceta corretamente e fez a transferência de conhecimento ao código corretamente.

```
1  var numero_de_insercoes,Nome_do_Cliente,Fruta_Preferida,Plano_B; //declaração de variáveis através do bloco "Definir variável"
2
3
4  numero_de_insercoes=window.prompt('Número de Clientes'); //variável receber número de clientes
5  var repeat_end = numero_de_insercoes;
6  for (varcount = 0;count < repeat_end;count++) { //bloco "Repetição" repete instruções até que a condição especifica seja falsa.
7    Nome_do_Cliente=0;
8    Fruta_Preferida=window.prompt('Inser Nome do Cliente'); //tela pedindo informação ao usuário.
9    Plano_B=window.prompt('Inserir uma segunda Alternativa'); //tela pedindo informação ao usuário.
10   window.alert(String(String('Nome do cliente') + String(Nome_do_Cliente)) + String(String(String('Fruta Preferida') + String(F
11 } //imprimir texto.
12 window.alert('Obrigado!'); //imprimir texto.
```

Figura 29 - Código em JavaScript de P9

O participante P9 foi além do que pedíamos na questão. Dando a opção de lista de frutas preferidas dos clientes, linhas 8 e 9, sem definir um número fixo de clientes, linhas 4 e 5, e gerando no final esta lista de informações linha 10. Este usuário nunca tinha tido contato algum com qualquer tipo de linguagem de programação e mostrou que as facetas o ajudaram na transferência de conhecimento com êxito. No grupo dos participantes com experiência em programação, o P16 conseguiu realizar a questão sem problemas e alertou para algo dentro da faceta ‘Código Python’ que não havíamos levado em consideração. Ele alertou na linha 5, da Figura 30 que não é obrigatório definir os comandos de texto na parte de cima do código, como os comandos “def text_prompt”, “return raw_imput”. Desta forma podemos dificultar a compreensão de usuários aprendizes, sendo que temos a opção de fazer o mesmo código de forma mais fácil.

```
1  nome = None #Declaração de variáveis
2  peso = None
3
4  '''def text_prompt(msg): #definindo comandos de texto
5    try: #não é obrigatório definir desta forma.
6      return raw_input(msg)
7    except NameError:
8      return input(msg)'''
9
10
11 nome = raw_input('') #tela para informação do nome
12 print(nome)
13 peso = raw_input('') #tela para informação do peso
14 if peso < 10: #Bloco de lógica 'Se verdadeiro' para condição peso menor que 10
15   print('Bagagem Permitida!')
16 else: #Bloco de Lógica "Se falso" a condição peso maior que 10
17   print('Peso ultrapassado!')
```

Figura 30 - Código em Python de P16

Podemos nos certificar mais dessa suspeita por meio do programa de P6, este participante teve algumas dificuldades com a LP visual e não conseguiu ter êxito na criação de seu programa. Porém ele conseguiu reescrever o seu código em Python e comentou o seu entendimento da colaboração dos blocos para o código na linhas 1 e 10 da Figura 31. Destacasse para as linhas 3, 4 e 5 que não houveram qualquer comentário para nós analisarmos a transferência de conhecimento deste aprendiz nesta parte do código.

```
1  fruta = None #Definir variavel
2
3  def text_prompt(msg):
4    try:
5      return raw_input(msg)
6    except NameError:
7      return input(msg)
8
9
10 for count in range(int(5)): #bloco Repetição
11   fruta = [text_prompt(fruta), None, None]
```

Figura 31 - Código em Python de P6

Vale ressaltar, ainda assim, que P6 compreendeu parte do código. Se este tivesse tido mais tempo para utilizar as facetas como ‘Suporte’ ou a ‘Requisitos’ ele poderia ter corrigido o seu programa. Apesar dos erros de sintaxe, ele compreendeu a semântica e a lógica de alguns comandos em código Python e isto já será uma grande ajuda para a continuação do aprendizado de programação deste usuário.

5 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Este trabalho oferece uma ferramenta de apoio a transferência de conhecimento de lógica de programação de uma linguagem visual Blockly para duas linguagens textuais JavaScript e Python. Por meio do modelo PoliFacets foram desenvolvidas facetas de significados de programas com características que os dados coletados neste estudo indicam colaborar positivamente na transição da linguagem visual para as textuais. Os participantes também puderam se conscientizar sobre a importância do aprendizado da leitura, escrita e lógica da linguagem de programação.

A busca por melhorar a construção um espaço, onde usuários possam programar de uma forma menos complexa e posteriormente fazer a transferência de conhecimento para uma linguagem de programação textual, é de grande utilidade. Pois, as dificuldades enfrentadas no ensino de programação são muitas a serem superadas, em grande parte, este conhecimento é necessário em várias disciplinas do curso da faculdade de computação e a falta de compreensão desmotiva muitos alunos ao prosseguimento dos estudos.

Mas, não somente para pessoas da área de computação, como também para pessoas de outras áreas como administração, biológicas e comunicação, que participaram do teste e são exemplos dos novos tipos de usuários, os end users, que sentem cada vez mais necessidade de expressar suas ideias por meio de programas e aplicativos. Então por meio desta pesquisa com aprendizes de programação pôde-se reforçar ainda mais esta ideia, de como o uso da ferramenta desenvolvida e outras similares, tanto para iniciantes quanto para professores ministrarem o curso, são eficientes para o processo de aprendizado.

A análise da nossa pesquisa, em cima do PoliFacets-Blockly, ficou caracterizada basicamente em dois ciclos, cada um com as seguintes etapas: (I) Planejamento de melhora; (II) Ação para implantação da melhoria; (III) Monitoramento e descrição dos efeitos da ação; (IV) Avaliação dos resultados da ação. Este método foi utilizado para o desenvolvimento do PoliFacets-Blockly I, seu impacto e nossas conclusões com base nas experiências relatadas dos usuários para posteriormente o desenvolvimento do PoliFacets-Blockly II.

No primeiro ciclo o planejamento de melhora entende-se como a elaboração da melhor forma de comunicação entre o design e o usuário para a criação das facetas utilizadas, com base na Engenharia Semiótica, o nosso principal foco de estudo foi a ontologia de metacomunicação. A ação para implantação da melhoria se deu com base nos resultados obtidos em trabalhos anteriores de ferramentas que utilizaram o modelo PoliFacets, como o AgentSheets e o PoliFacets Iv-Prog, que nortearam a direção para implantação do modelo na

ferramenta Blockly do Google. O monitoramento e descrição dos efeitos da ação das facetas foram analisadas em cima do primeiro teste com usuários para posterior avaliação dos resultados que serviram como base para o desenvolvimento do segundo ciclo da nossa pesquisa.

No primeiro teste com usuários, a maioria dos aprendizes não entendeu a necessidade de uma faceta que indicasse o número de vezes que determinado bloco foi utilizado o que nos levou a imaginar que um dos fatores sobre isso foi a questão do design implantado na faceta 'Tags', nesta utilizamos o modelo Treemaps para representação gráfica da quantidade. No segundo teste já com a mudança para a faceta 'Chart' onde a representação gráfica se dá por conta de um gráfico, os iniciantes em programação mesmo sem ter dificuldades com a compreensão do design, ainda sim, a maioria não compreendeu a importância da informação novamente. O que nos levou à compreensão que esta faceta necessitava de um problema de complexidade maior para o seu entendimento, já que o passado a eles durante os testes não revelavam a necessidade para o uso desta.

Outro ponto muito importante é que os participantes preferiram a faceta 'Código JS' a faceta 'Código Python' isso provavelmente se deve ao fato de que o PoliFacets-Blockly contém mais ajudas/suporte para a linguagem JavaScript, como o hiperlink do MDN na faceta 'Suporte' e pelo código em Python aparecer com definições de variáveis muito grande, deixando o código aparentemente mais difícil por estar maior do que o de JavaScript pro usuário. Então é importante equilibrar e corrigir esses pontos negativos para uma melhor compreensão dos aprendizes com o Python.

O feedback dos nossos testes servirão para o aprimoramento em trabalhos futuros para a faceta 'Chart', em como as informações dela podem ser melhor transmitidas e também outras sugestões para outras melhorias e criações de novas facetas como uma que explore mais o significado das categorias.

Cada uma das facetas apresentadas podem ser utilizadas de forma mais aprofundada, aplicação de melhorias para a faceta 'Chart', por exemplo, já que abordar questões mais complexas e evidenciar a importância dela aos aprendizes é um grande desafio e um dos principais propósitos do objeto documentado (Mota, 2013).

Explorar também a possibilidade de minimizar uma informação na faceta 'Instruções', 'Suporte' e 'Requisitos'. A avaliação da experiência dos usuários finais para a coleta de novos requisitos e aplicação de redesign. A criação de novas facetas para complemento, também para explorar novos espaços. Outro ponto importante seria uma faceta que explorasse as informações das categorias junto a uma breve aula de cada uma. Muitos

usuários iniciantes sentiram a necessidade de entender o que é uma categoria de forma que esta mensagem estivesse na tela paraajuda.

A expansão da documentação ativa do PoliFacets-Blockly pode ser alcançada por meio de uma faceta onde o usuário possa fazer o caminho inverso do proposto nesse trabalho. De uma linguagem de programação textual para uma linguagem de programação visual, ou seja, desta forma pode ser melhor abordada por professores ou até mesmo aos aprendizes que desejam testar os conhecimentos adquiridos até então. Na atual versão do nosso protótipo não tivemos essa faceta, por causa do seu nível de dificuldade e tempo necessário para implementação. Porém, seria explorar um passo importante, por meio de uma ratificação de equivalência, para certificar a transferência de conhecimento proposta na ferramenta.

Uma pesquisa interessante em longo prazo seria como criar facetas com diferentes tipos de diagrama e fluxograma. Um estudo com a combinação de regras de diagramas e equivalência dos blocos para os tipos de representação dentro destas seria uma forma totalmente diferente das facetas criadas até agora e com um nível de complexidade para implementá-las que requer muito tempo de investimento. Junto a uma nova forma de abordar ao aprendiz o problema e ajudando em uma melhor compreensão deste.

Outro aspecto importante para pesquisar seria a exploração das características colaborativas no uso das facetas e no design. Já que isso é algo de grande importância para a compreensão e é proporcionado nas salas de aula aos alunos. Ou seja, uma faceta com a possibilidade de colaboração entre aprendizes junto a possibilidade de questionamento de como isso poderia ajudar os iniciantes de programação e quais tipos de mudanças poderiam ocorrer no modelo para que isso ocorra.

REFERÊNCIAS

AURELIANO, V.; e TEDESCO, P. Avaliando o uso do Scratch como abordagem alternativa para o processo de ensino-aprendizagem de programação. In: XX Workshop sobre Educação em Computação, Curitiba, Paraná, 2012.

AßMANN, U. Architectural styles for active documents, Science of Computer Programming, Volume 56, Issues 1–2, April 2005, Pages 79-98, 2005.

BARATA, P.; CORREA, J. V. P.; MOTA, M. P. A Study on Knowledge Transfer Between Programming Languages by Programs Meanings Facets. XVI Simpósio Brasileiro Sobre Fatores Humanos em Sistemas Computacionais – IHC 2017. Joinville. Santa Catarina. Aceito para publicação.

BARATA, P..; MOTA, M. P. An Initial Study on Meanings Facets in IvProg programs. Proceedings of the 15th Brazilian Symposium on Human Factors in Computer Systems - IHC '16, 2016.

BARBOSA, S. D. J.; SILVA, B. S. D. A. Interação Humano-Computador. Rio de Janeiro: Elsevier, 2010.

BASTOS, J. Apoio à transferência de conhecimento de raciocínio computacional de linguagens de programação visuais para linguagens de programação textuais. PUC-Rio. Rio de Janeiro, 2015.

CODEORG. What Most Schools Don't Teach. YouTube. Disponível em: <https://www.youtube.com/watch?v=nKIu9yen5nc>. Acesso em: 14 de Setembro de 2017. Publicado em: 26 de Fevereiro de 2013.

DE SOUZA, C.S. The semiotic engineering of human-computer interaction. Cambridge, MA: The MIT Press, 2005.

DIAS, K. S.; SERRÃO, M. L. A Linguagem Scratch no Ensino de Programação: Um Relato de Experiência com Alunos Iniciantes do Curso de Licenciatura em Computação. XXXIV Congresso da Sociedade Brasileira de Computação - CSBC 2014. Belém. Pará, 2014.

FLORENZANO, C. Linguagem de programação é o novo idioma nas escolas. CBSI | SISTEMAS DE INFORMAÇÃO. Disponível em: <http://www.cbsi.net.br/2017/06/linguagem-de-programacao-e-o-novo-idioma-nas-escolas.html>. Acesso em: 14 de Setembro de 2017.

GONDIM, H. W. A. S.; AMBRÓSIO, A. P. L.; COSTA, F. M. TaskBoard - Using XP to Implement Problem-Based Learning in an Introductory Programming Course. Lecture Notes in Business Information Processing Agile Processes in Software Engineering and Extreme Programming, p. 162–175, 2011.

GUO, P. Python is Now the Most Popular Introductory Teaching Language at Top U.S. Universities. ACM. Disponível em: <https://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-u-s-universities/fulltext>. Acesso em: 16 de Setembro de 2017. Publicado em 7 de Julho de 2014.

JAKOBSON, R. “Linguistics and poetics”. In: T. A. Sebeok (ed.), Style in Language. Cambridge, MA: Th e MIT Press, pp. 350–377, 1960.

KÖLLING, M. The Greenfoot Programming Environment. ACM Transactions on Computing Education, 2010.

LEITÃO, C. F.; SILVEIRA, M, S.; e SOUZA, C. S. Uma Introdução à Engenharia Semiótica: Conceitos e Métodos. Rio de Janeiro, PUC-Rio e Rio Grande do Sul, PUCRS.NRC. 2010.

PAPERT, S. Mindstorms: Children, computers, and powerful ideas. Basic Books, Inc., 1980.

MALAN, D. J.; LEITNER, H. H. Scratch for budding computer scientists. Proceedinds of the 38th SIGCSE technical symposium on Computer science education - SIGCSE '07, 2007.

MOTA, M. P.; MONTEIRO, I. T.; FERREIRA, J. J.; SLAVIERO, C.; SOUZA, C. S. D. On signifying the complexity of inter-agent relations in agentsheets games and simulations. Proceedings of the 31st ACM international conference on Design of communication - SIGDOC '13, 2013.

MOTA, M. P.; SOUZA, C. PoliFacets: um modelo de design da metacomunicação de documentos ativos para apoiar o ensino e aprendizado de programação. Rio de Janeiro. 202p. Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, 2014.

O ensino da linguagem computacional. Disponível em: <http://revistapontocom.org.br/entrevistas/o-ensino-da-linguagem-computacional>. Acesso em: 12 de Agosto de 2017.

PRADO, M.E.B.B. O uso do Computador na formação do professor: Um enfoque reflexivo da prática pedagógica. Coleção Informática para a Mudança na Educação. ProInfo/SEED/MEC. Brasília, DF, 1999.

PRETTO, N. D. E. L.; SILVEIRA S. A. Além das redes de colaboração: internet, diversidade cultural e tecnologias do poder. Salvador: EDUFBA, 2008.

RAMOS, J. L. Pensamento Computacional na Escola e Práticas de avaliação das Aprendizagens. Uma Revisão Sistemática da Literatura, Évora, Centro de Investigação em Educação e Psicologia da universidade de Évora. 2015.

RAMOS, J. L. Pensamento Computacional na Escola, no currículo e na aprendizagem. YouTube. Disponível em: <https://www.youtube.com/watch?v=P00TrGVtx2E>. Acesso em: 14 de Setembro de 2017. Publicado em: 25 de Outubro de 2014.

REPENNING, A., IOANNIDOU, A. Agent-Based End-User Development. Communications of the ACM. Vol. 47, 43-46, 2004.

WOLZ, U.; STONE, M.; PULIMOOD, S. M.; PEARSON, K. Computational thinking via interactive journalism in middle school. Proceedings of the 41st ACM technical symposium on Computer science education - SIGCSE'10, 239, 2010.

APÊNDICE – ROTEIRO DO TESTE 1

1. Apresentação do Blockly;
2. Resolvessem um problema por meio da construção de um programa simples;
3. Explicação que durante a tarefa seria gerada facetas, outras representações que poderiam ajudar na compreensão de conceitos e significados do programa deles também em uma LP textual chamada JavaScript;
4. Foi pedido que o participantes construíssem o mesmo programa na LP textual JavaScript utilizando como fonte de consulta a documentação gerada pelas facetas. E que comentassem os seus códigos;
5. Ao final, os participantes tiveram que responder a um questionário online, do google forms, de avaliação das facetas e a experiência de explorá-las.

As perguntas que fizeram parte, ao final do teste, no questionário online estão listadas a seguir.

1. O que é uma faceta e quais os objetivos das facetas do Blockly?
2. Explique de que formas as facetas afetaram positivamente ou negativamente a execução da tarefa usando javascript.
3. Como você pode descrever a experiência de explorar as facetas do Blockly?
4. Cite dois pontos positivos do uso das facetas do Blockly.
5. Cite dois pontos negativos do uso das facetas do Blockly.
6. Qual a função dos blocos que utilizou no seu programa no blockly?
7. Qual a relação entre os blocos que você utilizou no seu programa no Blockly e alguma categoria de bloco?
8. Qual suporte ou ajuda você consegue obter usando as facetas sobre os blocos que utilizou?
9. Qual(is) bloco(s) foram mais utilizados no Blockly?
10. O que é uma categoria de bloco?
11. Qual a relação entre os blocos que você utilizou no blockly e as instruções de código JavaScript?
12. Quais as diferenças que você notou entre as duas linguagens? Foi possível perceber isso por meio de alguma faceta? Comente as diferenças.
13. O que você achou da faceta Instruções? Existe algo que possa ser melhorado? Deixe sua opinião.
14. O que você achou da faceta Tags? Existe algo que possa ser melhorado? Deixe sua opinião.
15. O que você achou da faceta Suporte? Existe algo que possa ser melhorado? Deixe sua opinião.
16. O que você achou da faceta Código? Existe algo que possa ser melhorado? Deixe sua opinião.

APÊNDICE – ROTEIRO DO TESTE 2

1. Foi realizada uma apresentação do Blockly;
2. Foi solicitado que eles resolvessem um problema por meio da construção de um programa simples aos iniciantes de programação e um programa mediano para os usuários com experiência em alguma linguagem de programação;
    2.1. Questão do grupo de iniciantes em programação: Faça um programa pro fruteiro que cadastre cinco clientes que chegaram no estabelecimento com nome e fruta preferida de cada um deles.
    2.2. Questão do grupo de usuários com experiência em programação: Faça um programa para uma companhia aérea que informe o peso da bagagem e caso o peso seja acima de 10kg o cliente seja avisado sobre o excesso de peso.
3. Foi explicado que durante a tarefa seria gerada facetas, outras representações que poderiam ajudar na compreensão de conceitos e significados do programa deles também nas LP´s textual JavaScript e Python;
4. Foi pedido que o participantes escolhessem uma entre as duas LP´s textuais e construíssem o mesmo programa utilizando como fonte de consulta a documentação gerada pelas facetas;
5. Foi solicitado aos participantes que ao final respondessem a um questionário online, do google forms, de avaliação das facetas e a experiência de explorá-las.

Foram feitas perguntas similares ao questionário online do teste 1, porém com o acréscimo das perguntas:

1. Qual a categoria dos blocos que você mais utilizou no seu programa no blockly?
2. Você compreendeu a relação entre as categorias e os blocos por meio das suas cores?
3. O que você achou da faceta Requisito? Existe algo que possa ser melhorado? Deixe sua opinião.
4. O que você achou da faceta Chart? Existe algo que possa ser melhorado? Deixe sua opinião.
5. Qual linguagem você preferiu usar? (JavaScript ou Python).
6. O que determinou na sua escolha entre uma e a outra?
7. Caso a sua escolha tenha sido Código Python! O que você achou da faceta Python? Existe algo que possa ser melhorado? Deixe sua opinião.
8. Escolha somente duas facetas das quais você mais gostou.