Designing Learning Methods: Programming with Visual and Textual Language in Python

Gonçalo Castro Henriques[1], Pedro Maciel Xavier[2], Victor de Luca Silva[3], Luca Rédua Bispo[1]

[1] Universidade Federal do Rio de Janeiro, LAMO-PROURB
gch@fau.ufrj.br
luca.bispo@fau.ufrj.br
[2] Universidade Federal do Rio de Janeiro, Engenharia Computação de Informação
pedromxavier@poli.ufrj.br
[3] Universidade Federal do Rio de Janeiro, Engenharia de Controle e Automação
vicdlsns@poli.ufrj.br

Abstract. At the fourth industrial revolution, programming is gaining relevance, and it promises to be a fundamental teaching subject as math, science, languages or the arts. Architects project more than buildings; they have developed innovative methods and are among the pioneers developing visual programming. However, after more than 10 years of use visual programming in architecture, despite its fast learning curve, it presents limitations to address complex problems. To overcome them, we propose associating the advantages of visual with textual languages in Python. The article reports the process to implement the discipline "*Computation for Architecture in Python*" at FAU-UFRJ. The methodology comprises the translation and adaptation of generic programming disciplines, and exercises, for architecture. The results are encouraging and demonstrate that students value learning programming. However, despite the participants' satisfaction with the discipline, they report difficulties in programming fundamentals, such as lists, loops and recursion.

Keywords: Computation, textual programming, visual programming, python, learning

1 Introdução

Projetar é antecipar o futuro, antevendo possibilidades. A capacidade de projetar depende de faculdades mentais, de aptidões específicas, sendo influenciada pelas técnicas e processos de que dispomos e que nos são ensinadas. A disciplina de projeto abarca uma gama ampla de conhecimentos,

de diferentes áreas, sendo constantemente renovada. Segundo Solá-Morales (2002), a Arquitetura, evolui por territorialização e re-territorialização de conhecimento externo. Com a revolução digital, e mais recentemente com a 4ª revolução industrial, a informação adquiriu uma amplitude e escala sem precedentes, aumentando a complexidade dos nossos desafios (Toffler 1984, Mitchell 2008, Schwab 2017 e Carpo 2018). Esta evolução tecnológica acelerada, recomenda projetar novos métodos.

Nesta realidade, da 4ª revolução industrial, o ensino de programação é considerado fundamental, no início da graduação, nas ciências exatas, na matemática e nas engenharias, embora a sua utilidade se deva futuramente estender a todas as áreas de conhecimento (Blumenfeld, 1988). Tal não acontece em geral na Arquitetura, e no Brasil em particular. Num estudo analisando 617 disciplinas de Arquitetura que usam informática, Natumi (2013), conclui que a informática é ensinada em Projeto assistido por computador 63%, Multimídia 17,5%, Tecnologia Informação 13%, Geoprocessamento 6,5%. O ensino de programação textual é residual, correspondendo a apenas 3,8% destas disciplinas. Pesquisamos sobre a existência de disciplinas de introdução à computação na UFRJ. Em resumo, a maioria das disciplinas obrigatórias da grade curricular pertence aos grupos das Ciências Exatas e da Terra, Engenharias ou Ciências Biológicas (Dados SIGA UFRJ). A maioria destas disciplinas tem 45 horas práticas e 15 teóricas, e tem como matriz as disciplinas do Departamento de Ciência da Computação (DCC-UFRJ). Estas disciplinas, começam também a ser requisitadas em outros cursos como direito na FGV.

Assistimos nos últimos 10 anos, a um interesse crescente dos arquitetos pela programação visual, para suportar processos algorítmicos de projeto. Para este interesse, contribuíram aplicativos de programação visual desenvolvidos por arquitetos como Rutten (Grasshopper 2008). Atualmente a maioria dos arquitetos depende, de programadores externos para adaptar as suas ferramentas de trabalho, o que limita a sua atuação em projeto, num mundo progressivamente codificado.

2 Linguagens programação visual e textual

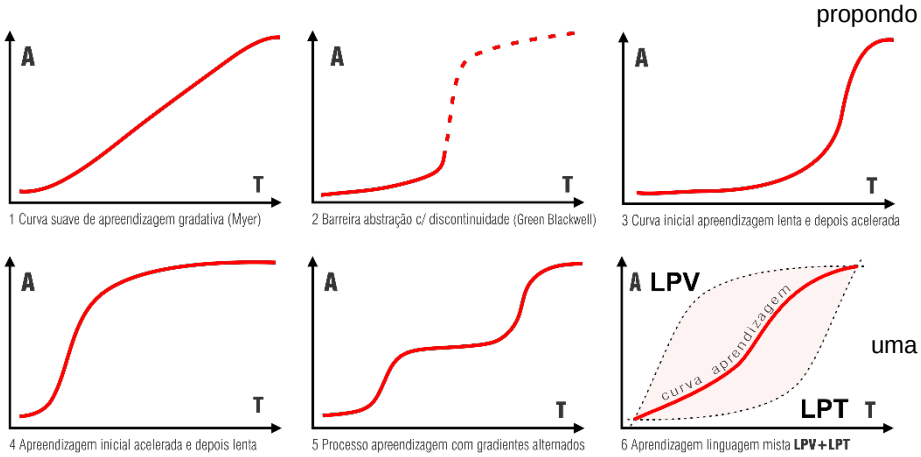
Para projetarmos e comunicarmos as nossas ideias recorremos a linguagens codificadas (Flusser 2004). A comunicação pode ser realizada utilizando linguagem oral, escrita, através de símbolos gráficos, de texto, numéricos, fórmulas matemáticas, diagramas, imagens, hologramas, entre outros. Uma linguagem de programação tem uma especificidade, precisa ser entendida por um computador, para poder ser processada (Leitão; Santos, 2011). O

desenvolvimento de um projeto em arquitetura, tem um paralelo com a programação, no sentido que também é necessário comunicar o projeto, para que resulte num edifício contruído. Tradicionalmente a execução passa por vários intervenientes, mas pelo menos em teoria, o processo generativo, construtivo e de montagem poderá ser automatizado, o que aproxima mais ainda o projeto da programação. Segundo Leitão (2012), a programação em arquitetura permite: a) automatizar as tarefas repetitivas, sendo esta uma das justificações mais frequentes para sua utilização (Blackwell, 2006), b) desenvolver formas geométricas e funcionalidades que não existem nas ferramentas padrão dos softwares disponíveis, c) introduzir variações de projeto utilizando métodos probabilísticos, como a aleatoriedade associada a regras de variação de sistemas generativos, d) verificar e melhorar o desempenho de soluções. Com a crescente necessidade de programação em arquitetura, ganha relevância a linguagem dessa programação. Para entender a sua origem e processo, apresentamos uma reflexão sobre a utilização de linguagens de programação em arquitetura.

As linguagens de programação textual, já existem há algum tempo, embora tenha sido aparecimento das linguagens visuais que despertou mais interesse em projetistas, especialmente os que não aprenderam a programar. As primeiras aplicações de programação em arquitetura, começaram nos 80, com o AutoLISP uma linguagem textual desenvolvida para o AutoCad. Essa linguagem no AutoCad vem sendo substituída por outras linguagens como Visual Basic, VB.Net, e mais recentemente pelo Python. Estas linguagens no AutoCad atuavam como macros, para automatizar a execução de tarefas, com scripts em linguagem textual. A partir de 2000, alguns arquitetos, não satisfeitos com a linguagem textual, desenvolveram uma alternativa mais adaptada ao pensamento visual do arquiteto. Em 2003, Robert Aish desenvolveu, com o Smart Geometry Group, aquele que é considerado o primeiro programa para arquitetura utilizando programação visual, o generative components (Martin et al, 2006), que viria a ser lançado comercialmente em 2008. Em 2007, o MIT desenvolveu aquela que é considerada a linguagem de programação visual mais popular: o SCRATCH, que foi desenvolvido para ensinar programação a crianças a partir dos 8 anos. Em paralelo arquitetos, como David Rutten desenvolveram alternativas para programação visual em arquitetura, como o Explicit History (2007, sucedendo à programação textual RhinoScript), que deu origem ao programa grasshopper em 2008. A programação visual, com a sua interface visual mais intuitiva, atraiu um número sem precedentes de interessados. Apesar do sucesso das linguagens visuais, estas apresentam um grande desenvolvimento inicial e depois tem um aprendizado mais lento. Esta afirmação, é confirmada pela experiência prática

no ensino em programação visual na FAU-UFRJ (Henriques 2016), mas procuramos completar esta intuição com uma revisão de literatura sobre processos de aprendizagem, para melhorar o ensino de programação em arquitetura. Recorremos a uma revisão sistemática da aplicação de linguagens de programação visual (LPV) e linguagens de programação textuais (LPT), (Noone e Mooney 2018). A revisão destes autores avaliou a introdução à programação em diferentes áreas de conhecimento, e faixas etárias. A pesquisa mostrou as vantagens pedagógicas do ensino de programação visual, e também algumas das suas limitações, para abordar problemas mais complexos, para o qual concluem com o estudo ser essencial utilizar a programação textual.

Figura 1 Curvas de aprendizagem no tempo (Adaptado de Aish e Hanna 2017), propondo uma curva esperada de aprendizagem com programação mista (LPV+LPT).



Na figura anterior apresentamos diferentes curvas de aprendizagem. Embora existam alguns estudos (Aish 2017), este tipo de informação é ainda difícil de precisar estatisticamente. No entanto, estas curvas de aprendizagem, ajudam a compreender o processo de aprendizagem no tempo. Os primeiros gráficos da figura são adaptados (Aish e Hanna 2017), sendo que o último gráfico proposto por nós, procura antever como poderá ser o processo de aprendizagem de uma linguagem de programação mista. A Curva de programação visual é inicialmente rápida, mas depois gradualmente mais lenta (Leitão 2012, Landim 2019, Villares 2019), aproximando-se de uma curva logarítmica. Pelo contrário a linguagem textual tem um início de aprendizagem mais lento e gradual, mas que com a prática aumenta exponencialmente (Aish 2017). Ambas as linguagens apresentam dificuldades no seu desenvolvimento, e com o intuito de melhorar a aprendizagem pensamos no uso de programação mista para resolver problemas mais complexos

recorrendo à modelagem paramétrica com programação visual e textual. É esperado que esta combinação permita controlar melhor o fluxo de dados em ciclos, estruturas condicionais e os processos recursivos. Escolhemos a linguagem Python, por ser uma linguagem ágil, de notação simplificada, e por ser a linguagem mais utilizada (Guo 2019), e estar embutida na maioria dos softwares de Arquitetura (Villares 2019).

3 Linguagens programação visual e textual

Esta pesquisa, procurou alternativas para ensinar programação em arquitetura de acordo com as suas especificidades. A hipótese colocada, é de que é possível desenvolver uma disciplina para ensinar a programar utilizando uma linguagem mista, visual e textual. Este artigo pretende partilhar essa metodologia e os resultados obtidos, que confirmam esta hipótese.

4 Metodologia

4.1 Estrutura geral

A pesquisa começou por identificar um problema, apresentando um projeto de pesquisa sobre o tema, propondo uma disciplina. A disciplina foi preparada em 3 etapas: estudo e aprendizagem da linguagem, aplicação piloto interna e implementação de disciplina experimental.

Identificação de problema. Com base na experiência no ensino de algoritmos visuais e modelagem na disciplina de MDA, foram identificadas dificuldades em projeto. Essas dificuldades, levaram a pensar numa disciplina de programação, que inclua linguagem textual, em Setembro 2019. Foi iniciado um estudo para analisar as disciplinas de programação existentes noutros cursos, pensando como as adaptar para arquitetura. Foram estudadas as linguagens existentes, onde prevalece o Python. Foi submetido um projeto de pesquisa à agência de fomento FAPERJ em Dezembro. A proposta assegurou 2 bolsistas com prática em Python, selecionados em Março, quando iniciamos o projeto. A disciplina foi preparada em três semestres: estudo da linguagem (Abril-Julho 2020), aplicação interna (Setembro-Fevereiro 2021), e implementação da disciplina (Março-Junho 2021).

No semestre de estudo, a equipa foi constituída pelo coordenador e pelos bolsistas de engenharia de controle e automação, e engenharia de Computação e Informação, ambos autores deste artigo. O coordenador introduziu o grupo no contexto de programação em arquitetura, no tipo de problemas e necessidades da área. Houve troca de ideias sobre os exercícios

e metodologias. Após a introdução analisamos e sistematizamos informação sobre as disciplinas de programação existentes, quanto à sua estrutura, conteúdo e metodologia de ensino. Estabelecemos uma dinâmica em que os bolsistas partilharam semanalmente o seu conhecimento sobre a matéria com o coordenador, através de reuniões expositivas, seguidas de reuniões com exercícios de cada matéria. Nestes encontros semanais remotos, abordamos os temas: 1- Funções, variáveis, tipos dados 2- Listas e operações com listas e tuplas 3- Estruturas condicionais 4- Loops e matrizes 5- Teoria recursão. Esta dinâmica permitiu testar a linguagem Python e antever possibilidades de integração com a programação visual e com projeto.

O semestre de teste, contou com um novo bolsista, estudante de arquitetura, também autor do artigo, para prosseguir a adaptação para a Arquitetura. Neste período testamos mais intensivamente a linguagem textual em Python, dentro do módulo de programação Python do Grasshopper e a sua convergência com os componentes visuais do grasshopper. Esta articulação pretendeu associar as qualidades das duas linguagens. Tomamos a importante decisão de utilizar a linguagem textual do Python para chamar os métodos do grasshopper. Na maioria dos cursos atuais de Python em Grasshopper, é utilizada a linguagem textual RhinoScript, o que obriga a estudar outra linguagem. Ao chamar diretamente, dentro do módulo Python, os métodos dos componentes do grasshopper, utilizamos a experiência dos utilizadores de grasshopper. A maior dificuldade, foi não haver manuais de utilização para este método, como acontece com o RhinoScript, que tem manuais desenvolvidos pelos seus programadores.

4.2 Implementação experimental

Para implementar a matéria “Computação para a arquitetura em Python 2020.2”, definimos o perfil dos participantes na disciplina, a estrutura das aulas e o seu conteúdo.

Em experimentos anteriores de programação aplicada em projeto, domina a ideia que a linguagem textual é mais apropriada para alunos com conhecimentos avançados (Celani e Vaz, 2012), procuramos ensinar programação nas etapas iniciais de projeto. Esta escolha segue os exemplos dos cursos de engenharia, que promovem a literacia digital, e o ensino de estruturas lógicas. Para incentivar a aplicação de linguagem textual e visual, procuramos também alunos com experiência em programação visual. Na chamada da FAU-UFRJ houve 35 interessados. Selecionamos 15 valorizando: alunos iniciantes, com perfil e interesse em computação, alunos do início do curso com experiência em programação visual, e alunos avançados, experientes em programação visual. Curiosamente a maioria dos candidatos procurou a disciplina, para o trabalho final do curso. Entendemos que esse

perfil não se adapta a esta experiência, dado não terem experiência prévia em programação, e não haver tempo para amadurecer esse conhecimento, e os aplicar num projeto complexo. A turma contou com alunos de quase todos os semestres, um doutorado, e um Professor, ambos do LAMO-Prourb.

Estrutura aulas e conteúdo. Para implementar a matéria, utilizamos uma eletiva já existente, Computação Gráfica Aplicada à Arquitetura FAR-606, na FAU-UFRJ, com 20 aulas, 3 horas por semana, num total de 60 horas. Verificamos que no ensino de Python em outros cursos, é frequente haver duas aulas por semana, uma prática e outra teórica. Para ultrapassar esta dificuldade dividimos as aulas, numa parte teórica e outra prática com exercícios. De acordo com a matéria definimos aulas teóricas (com exercícios no final), alternadas com aulas práticas, com exercícios mais desenvolvidos de projeto. Nas aulas apresentamos exercícios para casa, para facilitar a absorção de conteúdos. Para complementar definimos um dia, para tirar dúvidas com os monitores, o que segundo os alunos foi providencial.

A preparação e adequação dos exercícios ao campo da arquitetura, foi fundamental. A maioria dos exercícios de Python das engenharias, eram exercícios numéricos, com poucos exercícios que recorressem ao pensamento visual, ou que abordassem problemas geométricos relacionados com arquitetura. No período preparatório desenvolvemos, para cada matéria, um conjunto de exercícios, com problemas de projeto para estimular a programação textual e visual. Durante a preparação ao resolvermos os exercícios, fomos melhorando a colocação dos problemas, com soluções mais simples. Desenvolvemos alguns exercícios complexos, que optamos por não incluir, por requerem mais tempo. A adaptação, incluiu a escolha de aprofundar determinadas matérias, em detrimento de outras. Por exemplo, aprofundamos o ensino de estruturas de dados lineares, como listas, na preparação e no ensino da disciplina, enquanto optamos por não ensinar algumas matérias analisadas sobre dicionários e conjuntos. Demos menos atenção a algoritmos de ordenação, que é um tópico fundamental nos cursos de Computação e Engenharias. Por outro lado, demos mais importância a laços de repetição e estruturas condicionais, temas recorrentes nas aulas. Devido à implementação da disciplina ter ocorrido num semestre remoto em vez de 16 aulas planejadas, tivemos que nos adaptar a 12 semanas, o que consideramos escasso.

Relativamente ao conteúdo, a experiência do ensino da disciplina MDA Modelagem Aplicada à Arquitetura, com programação visual, desde 2015, recomenda ensinar matérias com aplicação prática. Assim foi necessário coordenar matéria, com exercícios para casa e com 2 trabalhos práticos da disciplina. A escolha de problemas, que tenham aplicação no universo de projeto é importante para potenciar futuras aplicações.

4.3 Trabalhos propostos na disciplina

1-Algoritmo de Fachada combinatória. Neste exercício pretendemos estabelecer regras, para encontrar uma família de soluções de projeto, o que seria trabalhoso com processos convencionais. A aplicação visou encontrar múltiplas soluções de fachada combinando diferentes tipos. Assim pedimos um algoritmo que crie uma matriz de N x M elementos, com módulos de 3 x 3 metros, e 4 tipos de fachada [A, B, C, D]. Como entrada do algoritmo deve ser selecionada a percentagem de cada tipo, e calculado o número de cada tipo, para a matriz escolhida. Após definidas as quantidades deve ser gerada uma frase com os tipos [A, A, A, A, B, B, C, D, D]. O algoritmo deve depois calcular combinações aleatórias destas letras [D, A, D, A, C, A, B, A, B]. Finalmente deve ter uma sub-rotina para associar as letras com um conjunto de formas. Estimulamos o uso de componentes de programação visual, como sliders para as entradas, para definir as matrizes, número de cada tipo, e para semente de aleatoriedade. O algoritmo principal deveria ser escrito textualmente em Python. O algoritmo faz a gestão de listas, associando letras com formas.

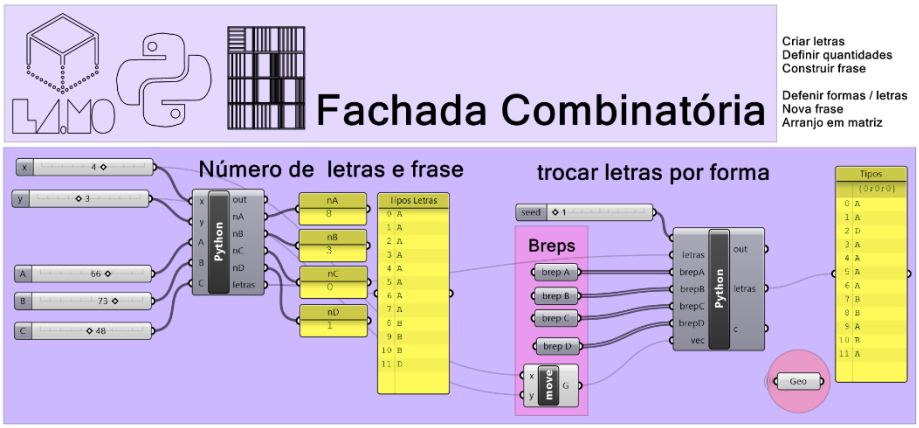


Figura 2 Fachada combinatória, módulos Python e respetivos outputs. Fonte: autores.

É necessário desenvolver a volumetria de cada tipo de módulo da fachada de acordo com princípios de composição arquitetónica, como quantidade e tipo de luz, para criar efeitos de ritmo, dinamismo, variação, profundidade, etc. Estas qualidades, não estão no algoritmo, que para apresentar boas soluções, precisa de tipos bons, e que formem um bom conjunto desenvolvendo a articulação entre o todo e a parte, e vice-versa. A coordenação todo-parte em Arquitetura é conseguida tradicionalmente por sistemas de proporções, como por exemplo a regra de ouro, mas que nem sempre são declarados explicitamente. A seleção das melhores soluções combinatórias de projeto também depende de critérios de projeto. Neste caso o algoritmo amplia a geração de soluções, mas para que estas tenham significado, devem radicar em valores de projeto, que estão para além da aplicação eficiente do algoritmo.

Ou seja, são decisões de projeto que dão significado ao uso da programação em Arquitetura.

2- Algoritmo desempenho da fachada combinatória. Devido ao semestre ser mais curto, propusemos um segundo exercício partindo do primeiro. Pedimos um algoritmo para selecionar soluções de fachada, de acordo com critérios de desempenho. Indicamos como referência o edifício do Instituto de Resseguros do Brasil, dos Irmãos Roberto (Rio Janeiro 1941). Este edifício possui elementos de sombreamento que o adaptam melhor à luz solar. As fachadas utilizam elementos, com diferentes graus de fechamento, para controlar a luz interior e evitar aquecimento excessivo. Na sua concepção foi utilizada tecnologia moderna, para calcular a luz, embora a aplicação estivesse limitada pelos sistemas construtivos padronizados e a repetição modular. Este segundo exercício pretendeu desenvolver soluções que cumpram o desempenho, mas permitindo uma maior variação nos módulos.

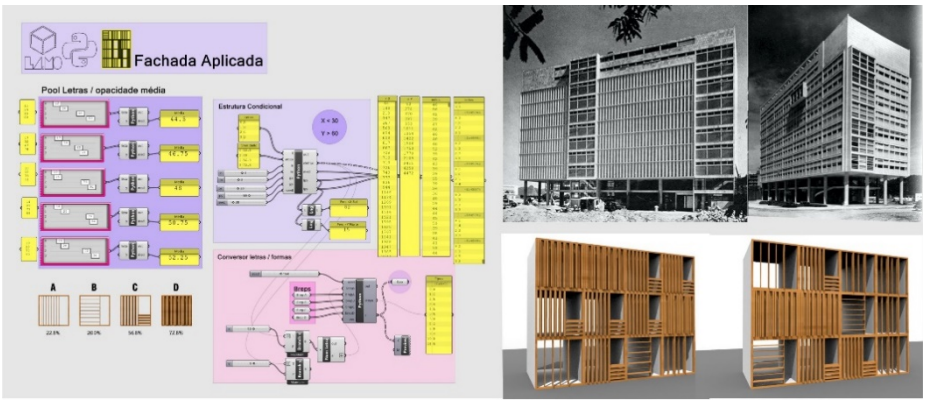
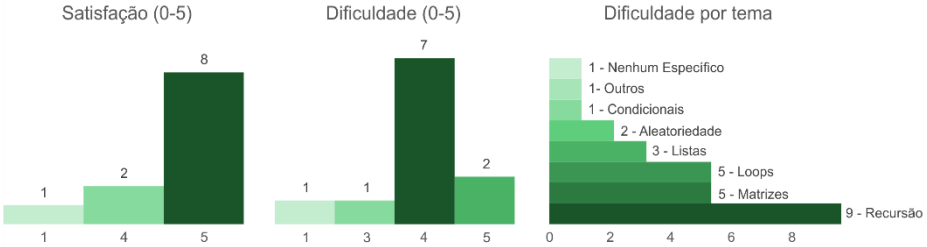


Figura 3 Fachada aplicada: código misto, referências MMM Roberto (Arquivo NPD, domínio público), e aplicação, fonte autores.

A proposta foi desenvolver um algoritmo ágil, em pouco tempo. Para isso, deveriam calcular a opacidade de cada tipo [A, B, C, D], através da percentagem de superfície de cada módulo, adequando a sua opacidade para influenciar o desenho global. Pretendeu-se gerar combinações de módulos aleatórias, com uma estrutura condicional para verificar o cumprimento de requisitos, como variedade de combinações, com um valor médio máximo e mínimo de opacidade. A fachada Norte deve ter opacidade superior a 60%, e a fachada sul inferior a 30%. Ou seja, o algoritmo deve combinar e identificar soluções que cumpram. Este desenvolvimento obrigada a um desenvolvimento coordenado dos módulos e testes estatísticos para que cumpram o objetivo global. Valorizamos mais do que a solução o desenvolvimento de um processo expedito, de baixa resolução, para as etapas iniciais de projeto. Uma versão mais aprofundada deverá considerar a localização geográfica, hora do dia, e posição solar, calculando a radiação solar e iluminância.

5 Resultados

Avaliamos a satisfação dos alunos através de um questionário, preenchido por 11 participantes. No questionário constam a avaliação numérica, assim como respostas escritas sobre o curso. A satisfação média foi de 4,5 pontos numa escala de 1 a 5, com desvio de 0,93, indicando que a grande maioria ficou satisfeita com o curso. A dificuldade média, na mesma escala foi de 3,8 pontos, com desvio de 1,08, o que mostra que o curso foi complexo. Dos participantes nenhum tinha apreendido apenas programação escrita, enquanto metade tinha experiencia em programação visual, 27,3% tinha experimentado uma abordagem mista, e 18,2% não tivera programação.



Avaliando a dificuldade por conteúdo, os mais selecionados foram recursão (9 votos) acompanhado por loops e matrizes (ambos com 5 votos); por outro lado, a parte inicial da matéria, que envolve funções, variáveis e dados, não foi votada. Houve um voto para nenhum conteúdo específico e para “outros” dizendo haver uma dificuldade geral para introduzir novos conceitos. Todos alunos afirmaram ver potencial na aplicação do conhecimento adquirido na disciplina para os mais variados fins na arquitetura, e 90,9% também veem potencial de aplicação fora da arquitetura.

Em resumo, os alunos reconhecem o valor do aprendizado lógico de programação e viram aplicação na automatização de tarefas, organização, planejamento e processamento de dados, alguns inclusive foram mais além e observaram a possibilidade de integrar em sistemas complexos, como o georeferenciamento. Vários sintetizaram bem as principais aplicações do conhecimento passado na disciplina, havendo inclusive relatos de uso prático dos conceitos da matéria no estágio ou em trabalhos de outras disciplinas, sendo que alguns procuraram estender o conhecimento, após a conclusão da matéria. Um dos pontos fortes mais citados foi a maior liberdade e possibilidade de controle sobre a ferramenta nas escolhas de projeto. De modo geral, os alunos sentiram a experiência como uma boa introdução ao assunto. Alguns elementos facilitam a receção, como a linguagem da computação sendo trazida para elementos cotidianos e a pluralidade da turma. No entanto, nem todos que tiveram essa perceção. Os prazos curtos foram uma dificuldade, a matéria necessita prática para se desenvolver. Houve relatos das aulas serem longas, o que pode levar a pensar em alternativas. Houve também certa dificuldade em lidar com temas e exercícios mais abstratos; e de aplicar

conceitos com outras finalidades. As ideias da lógica de programação são importantes para garantir generalidade e um maior leque de possibilidades não só para arquitetos, mas para todos que desejem projetar usando programação.

6 Discussão

Entre as melhorias sugeridas, concluímos que será interessante desenvolver material de apoio para a disciplina. Uma ideia será uma página com tutoriais, exercícios passados e possíveis soluções. O material de apoio, além de auxiliar os alunos cursando a disciplina, pode ser um meio para aumentar o alcance e difusão do conhecimento, a um maior número de alunos. Um material bem elaborado e aberto pode atrair mais alunos e professores para estudar e lecionar o tema, além de ajudar a preparação da disciplina.

Refletimos também sobre a lógica de adaptar o ensino de programação à arquitetura. Na utilização de programação em projeto, a informação contextual ganha relevância acrescida, para dar significado ao problema. Ou seja, o significado de um problema computacional, está fora do próprio problema. Então não se trata apenas de tornar uma lógica operacional, mas sim de encontrar problemas em que a aplicação faça sentido, só com essa contextualização, podemos conseguir resultados com real significado na área.

Agradecimentos

A todos que apoiaram o projeto, nomeadamente a FAPERJ. O terceiro autor é bolsista FAPERJ-IT 2019.2 de Iniciação Tecnológica (260.029/2020), e o quarto bolsista FAPERJ-IC 2019.2 de Iniciação Científica (201.044/2020).

Referências

Aish, R., & Hanna, S. (2017). Comparative evaluation of parametric design systems for teaching design computation. *Design Studies*, *52*.

Blackwell, A. (2006). Psychological Issues in End-User Programming. In H. Lieberman, F. Paternò, & V. Wulf (Eds.), *End User Development. Human-Computer Interaction Series* (1st ed., Vol. 9, pp. 9–30). Springer Nature.

Blumenfeld, S. (1988). *New Illiterates and How You Can Keep Your Child from Becoming* (2nd ed.). Paradigm Co

Carpo, M. (2017). The Second Digital Turn. In *The Second Digital Turn*. The MIT Press.

Celani, G., & Eduardo Verzola Vaz, C. (2012). CAD scripting and visual programming languages for implementing computational design concepts: A comparison from a pedagogical point of view. *IJAC*. https://doi.org/10.1260/1478-0771.10.1.121

Coates, P. (2010). Programming Architecture. Routledge Taylor & Francis Group.

Flusser, V., & Cardoso, R. (2007). O Mundo Codificado. Por uma filosofia do design e da comunicação. In *Cosac Naify*. Cosac & Naify.

Guo, P. (2014). *Python is Now the Most Popular Introductory Teaching Language at Top U.S. Universities*. BLOG@CACM. http://cacm.acm.org/blogs/blog-cacm/176450

Henriques, G. C. (2016). Arquitetura algorítmica: Técnicas, processos e fundamentos. *ENANPARQ IV Encontro Da Associação Nacional de Pesquisa e Pós-Graduação Em Arquitetura e Urbanismo*, 1–19. https://doi.org/10.13140/RG.2.1.3479.3209

Henriques, G. C., Bueno, E., Lima, D. L. C., & Sardenberg, V. (2019). Generative Systems: Intertwining Physical, Digital and Biological Processes, a case study. In J. P. Sousa, G. C. Henriques, & J. P. Xavier (Eds.), *Architecture in the Age of the* 4th Industrial Revolution-37th eCAADe and 23rd SIGraDi Conference (pp. 15–34).

Landim, G. (2019). Programação para Arquitetura: linguagens visuais e textuais em Projeto Orientado ao Desempenho, Dissertação de Mestrado, IAB, S. Paulo.

Leitão, A., & Santos, L. (2011). Programming Languages for Generative design - Visual or Textual? *Respecting Fragile Places (29th ECAADe Conference Proceedings)*.

Leitão, A., Santos, L., & Lopes, J. (2012). Programming languages for generative design: A comparative study. *International Journal of Architectural Computing*.

Martin, A., Yau, A., Berglund, J., & Hardy, S. (2006). *Generative Components research, 2006 Beijing Biennale exhibition*. 25.

Natumi, Y. (2013). O ensino de informática aplicada nos cursos de graduação em arquitetura e urbanismo no Brasil, Dissertação de Mestrado [Universidade São Paulo Faculdade de Arquitetura e Urbanismo].

Noone, M., & Mooney, A. (2018). Visual and textual programming languages: a systematic review of the literature. *Journal of Computers in Education*.

Procópio, et all (2021). Parametria e o desejo de uma computação integrada em projeto. *PIXO – Revista de Arquitetura, Cidade e Contemporaneidade*, *5*(17).

Sola-Morales, I. de. (2002). *Territorios*. Gustavo Gili.

Terzidis, K. (2006). Algorithmic Architecture. In *Algorithmic Architecture.* Routledge.

Villares, A. B. do A. (2019). Taxonomia de Temas para Ensino de Programação em Contexto Visual. In *Dissertação Mestrado, Civil, Faculdade De Engenharia UNICAMP*, Universidade Estadual de Campinas.