The Guizmo Framework for Generating Final User Interfaces from Wireframes

Óscar Sánchez Ramón
University of Murcia
Faculty of Computer Science
Murcia, Spain
osanchez@um.es

Jesús García Molina
University of Murcia
Faculty of Computer Science
Murcia, Spain
jmolina@um.es

Nicolas Burny
Université catholique de Louvain
Louvain Research Institute in Management and Organizations
Louvain-la-Neuve, Belgium
nicolas.burny@uclouvain.be

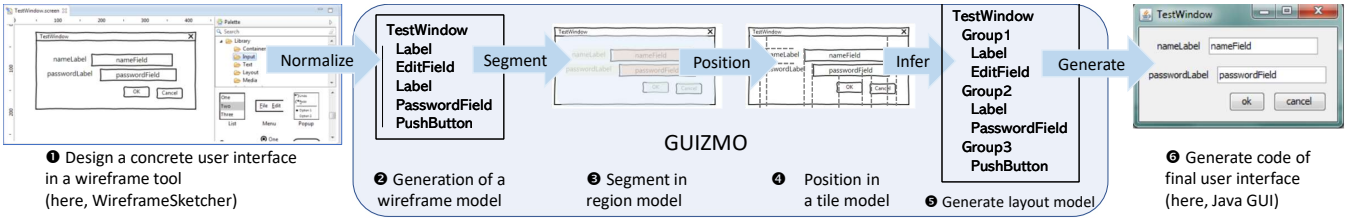

Figure 1: Overview of generation process with Guizmo.

ABSTRACT

We demonstrate Guizmo, a model-driven engineering framework aimed at generating final graphical user interfaces from wireframes according to the following process: (1) design a concrete user interface in a wireframe tool, (2) export/import the wireframe definition into Guizmo for generation of a wireframe model, (3) for segmenting the wireframe into regions by region detection, (4) for inferring positions and dimensions of individual interface elements, (5) in order to obtain a complete layout model that is used for generating the code of a final user interface by model-to-code generation.

CCS CONCEPTS

• **Software and its engineering** → *Graphical user interface languages*; **System modeling languages**; *Source code generation*; **Model-driven software engineering**; • **Computing methodologies** → Image segmentation; • **Human-centered computing** → *Interaction design process and methods.*

KEYWORDS

Model-based user interface design; model-driven engineering; model-to-model transformation; model-to-code generation; wireframes

ACM Reference Format:

Óscar Sánchez Ramón, Jesús García Molina, and Nicolas Burny. 2023. The Guizmo Framework for Generating Final User Interfaces from Wireframes. In *Companion of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '23 Companion), June 27–30, 2023, Swansea, United Kingdom.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3596454.3597189

EICS '23 Companion, June 27–30, 2023, Swansea, United Kingdom

ACM ISBN 979-8-4007-0206-8/23/06.
https://doi.org/10.1145/3596454.3597189

1 MOTIVATIONS

For a long time, significant efforts have been made to optimize the development life cycle of a graphical user interface (GUI), in particular by ensuring the best possible continuity between the artifacts produced in the preliminary stages of development and the final stage of coding [8] so as not to lose the effort produced during the design phase, which is known to be highly iterative, but also not to introduce a break. At the level of the concrete user interface, these artifacts include sketches [7], screenshots [9], mock-ups [8], textual descriptions [2], and wireframes [11] that are designed with different levels of fidelity [7] depending on the resources available in the project. A systematic literature review of strategies to automatically generate web sites identified three categories based on the dominant strategy used for automatic generation [10]: *examples based* [11], mock-up-driven [7]), and artificial intelligence (AI)-driven generation [8] (for example, Sketch2Code uses AI to convert hand-drawn wireframes of websites into working HTML code and MetaMorph does this from sketches [11]). Furthermore, much progress has been made in determining the layout of user interfaces using classical [5] and AI-based techniques [12].

2 AIMS AND GOALS OF GUIZMO

GUIZMO (GUIs by Models) consists of a software environment aimed at supporting the migration of legacy interactive systems, particularly applications created with Rapid Application Development Environments (RAD), based on the principles of Model-Driven Engineering (MDE). MDE has been proven effective for generating a single user interface [11] (*e.g.*, from business processes [13]) or many interfaces [2], for reverse engineering of user interfaces (*e.g.*, by transformations [4]), and for automatic analysis [3]. When facing the modernization of GUIs of applications developed with these applications, developers must deal with two non-trivial issues: (1) the GUI layout is implicitly provided by the position of the GUI elements [8] while taking advantage of current features of GUI technologies often requires an explicit, high-level layout

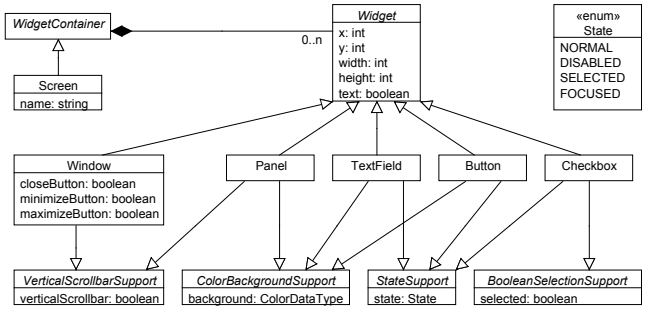

Figure 2: Metamodel of GUI elements in WireframeSketcher.

model [12]; (2) we must deal with event handling code that typically mixes concerns such as GUI and business logic. Tackling a manual migration of the GUI of a legacy system, *i.e.*, re-programming the GUI, is time-consuming and costly for businesses [4]. Guizmo generates final GUIs from wireframes as follows (Fig. 1): (1) design a concrete GUI in a wireframe tool – for the time being, we selected the WireFrameSketcher software because it natively offers an Eclipse plug-in that (2) exports/imports the wireframe definition into Guizmo for generation of a wireframe model by model injection via Gra2MoL [6], (3) segment the wireframe into regions by region identification and explicit containment (*i.e.*, every GUI element is represented by a region, that is a rectangular area defined by its coordinates), (4) infer positions and dimensions of individual GUI elements, (5) obtain a complete layout model that is used for generating the source code of a final GUI. WireframeSketcher was selected as a wireframing tool that helps designers quickly create wireframes, mockups and prototypes for desktop, web and mobile applications. It can be executed as a desktop application as well as a plug-in for any Eclipse IDE. WireframeSketcher outputs an Ecore model conforming to a predefined metamodel that is available in the distribution. The Guizmo pattern matching engine matches layout patterns to create a tree of GUI elements augmented by relationships between them. Four GUI layout patterns are used:

- *horizontal/vertical flow*: select the nodes connected by one outgoing edge with the xInterval/yInterval equals to Before or Meets, two relationships between from the 2D Allen interval algebra [1], which is captured in the tile model (Fig. 3).
- *gridLayout*: searches recursively for 2×2 node-subgraphs connected among them to form a rectangular grid of $n \times m$ nodes. The nodes inside the grid cannot contain edges that point to nodes outside the grid, only the border nodes of the grid are allowed to have connections to the nodes outside.
- *BorderLayout*: analyses the graph looking for subgraphs containing some of the following areas: North, South, East, West, and Center. For example, a left-aligned part (East) and a right-aligned part (West) of a subgraph should match.

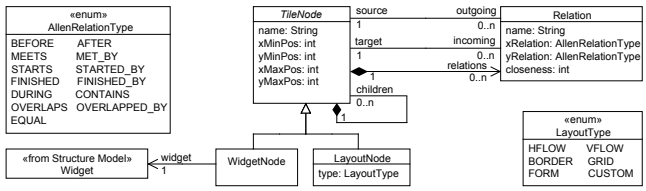

Figure 3: Metamodel of the Tile model with Allen algebra.

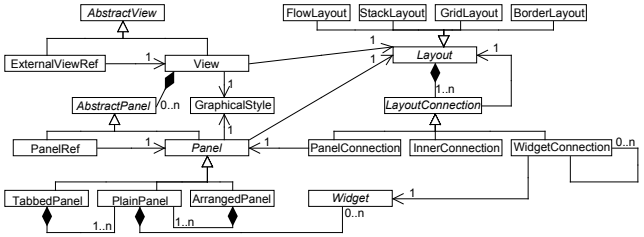

Figure 4: Metamodel of Concrete User Interface for M2M.

The process of matching the layout patterns of the sequence is performed according to the closeness levels. The algorithm defines a current closeness level which is used to limit the relations which the patterns are matched against, so only the relations with a closeness level equal to or lower than the current level are candidates to be matched. Therefore, at first, the current level is the lowest level, so the layouts in the sequence are applied to the relations with the lowest level. If there are no matches, then the current level is increased and the sequence is applied to the relations marked with the lowest level and the next one. If there are no matches, the and so forth. Note that this makes a partition of the graph in connected components, so each connected component is a subgraph of the original graph where all edges have a closeness level equal to or less than the limit. When the sequence has been tried with all the closeness levels and there have been no matches, the algorithm stops since no solution can be found by applying such a sequence. As a result of this step, a Layout Model which represents the design of the views in terms of composite layouts is achieved.

3 GUIZMO WALKTHROUGH

Model-to-model transformations are exploited to ensure stages 2 and 3 (from wireframe model to region model, then from region model to layout model), while stage 5 is ensured by model-to-code generation. The layout inference component (LAyout gueSSER) of the Guizmo framework has been improved to support the generation of final UIs from wireframes, especially the generation of Java Swing GUIs for the ZK environment with transition between them, from wireframes created with WireframeSketcher tool. While Guizmo has been crafted to remain as independent as possible from the input model (here, the wireframe model obtained from WireFrameSketcher) and the target programming language (here, the Java Swing/Zx environment), it still requires redefining the transformations for each input/output desired. Although these transformations are one-to-one mappings, they will change depending on the wireframe tool or sketching input. For example, any WireFrameSketcher element as specified in the corresponding metamodel (Fig. 2) can be mapped to one and only one GUI element belonging to the Concrete User Interface as specified in the metamodel (Fig. 4).

OPEN SCIENCE

The GitHub repository is at https://github.com/osanchezUM/guizmo.

ACKNOWLEDGMENTS

The authors of this paper are very grateful to the anonymous EICS reviewers and the associate chair for their insightful and constructive comments on earlier versions of this manuscript.

REFERENCES

[1] James F. Allen. 1983. Maintaining Knowledge about Temporal Intervals. *Commun. ACM* 26, 11 (nov 1983), 832–843. https://doi.org/10.1145/182.358434

[2] Nathalie Aquino, Jean Vanderdonckt, Nelly Condori-Fernández, Óscar Dieste, and Óscar Pastor. 2010. Usability Evaluation of Multi-Device/Platform User Interfaces Generated by Model-Driven Engineering. In *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (Bolzano-Bozen, Italy) *(ESEM '10)*. Association for Computing Machinery, New York, NY, USA, Article 30, 10 pages. https://doi.org/10.1145/1852786.1852826

[3] Abdo Beirekdar, Jean Vanderdonckt, and Monique Noirhomme-Fraiture. 2002. A Framework and a Language for Usability Automatic Evaluation of Web Sites by Static Analysis of HTML Source Code. In *Computer-Aided Design of User Interfaces III, Proceedings of the Fourth International Conference on Computer-Aided Design of User Interfaces, May, 15-17, 2002, Valenciennes, France*, Christophe Kolski and Jean Vanderdonckt (Eds.). Kluwer, 337–348.

[4] L. Bouillon, Q. Limbourg, J. Vanderdonckt, and B. Michotte. 2005. Reverse engineering of Web pages based on derivations and transformations. In *Proceedings of the Third Latin American Web Congress* (Buenos Aires, Argentina) *(LA-WEB 2005)*. 11 pp.–. https://doi.org/10.1109/LAWEB.2005.29

[5] Ahmet Selman Bozkir and Ebru Akcapinar Sezer. 2018. Layout-based computation of web page similarity ranks. *International Journal of Human-Computer Studies* 110 (2018), 95–114. https://doi.org/10.1016/j.ijhcs.2017.10.008

[6] Javier Luis Cánovas Izquierdo and Jesús García Molina. 2009. A Domain Specific Language for Extracting Models in Software Modernization. In *Model Driven Architecture - Foundations and Applications*, Richard F. Paige, Alan Hartman, and Arend Rensink (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 82–97.

[7] Adrien Coyette, Sascha Schimke, Jean Vanderdonckt, and Claus Vielhauer. 2007. Trainable Sketch Recognizer for Graphical User Interface Design. In *Human-Computer Interaction - INTERACT 2007, 11th IFIP TC 13 International Conference, Rio de Janeiro, Brazil, September 10-14, 2007, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 4662)*, Maria Cecília Calani Baranauskas, Philippe A. Palanque, Julio Abascal, and Simone Diniz Junqueira Barbosa (Eds.). Springer, 124–135. https://doi.org/10.1007/978-3-540-74796-3_14

[8] Harsh Dave, Sarah Sonje, Jaswantsingh Pardeshi, Sheetal Chaudhari, and Pooja Raundale. 2021. A survey on Artificial Intelligence based techniques to convert User Interface design mock-ups to code. In *Proceedings of International Conference on Artificial Intelligence and Smart Systems* (Coimbatore, India) *(ICAIS '21)*. 28–33. https://doi.org/10.1109/ICAIS50930.2021.9395994

[9] Abner Augusto Lima de Oliveira and Cidcley Teixeira de Souza. 2017. Paper Prototyping in a Model-Driven Process for Android Application Simulation Support. In *Proceedings of the XXXI Brazilian Symposium on Software Engineering* (Fortaleza, CE, Brazil) *(SBES '17)*. Association for Computing Machinery, New York, NY, USA, 267–272. https://doi.org/10.1145/3131151.3131192

[10] Thisaranie Kaluarachchi and Manjusri Wickramasinghe. 2023. A systematic literature review on automatic website generation. *Journal of Computer Languages* 75 (2023), 101202. https://doi.org/10.1016/j.cola.2023.101202

[11] Vinoth Pandian Sermuga Pandian, Sarah Suleri, Christian Beecks, and Matthias Jarke. 2020. MetaMorph: AI Assistance to Transform Lo-Fi Sketches to Higher Fidelities. In *Proceedings of the 32nd Australian Conference on Human-Computer-Interaction* (Sydney, NSW, Australia) *(OzCHI '20)*, Naseem Ahmadpour, Tuck Wah Leong, Bernd Ploderer, Callum Parker, Sarah Webber, Diego Muñoz, Lian Loke, and Martin Tomitsch (Eds.). ACM, 403–412. https://doi.org/10.1145/3441000.3441030

[12] Irfan Prazina, Šeila Bećirović, Emir Cogo, and Vensada Okanović. 2023. Methods for Automatic Web Page Layout Testing and Analysis: A Review. *IEEE Access* 11 (2023), 13948–13964. https://doi.org/10.1109/ACCESS.2023.3242549

[13] Kênia Soares Sousa, Hildeberto Mendonça Filho, Jean Vanderdonckt, Els Rogier, and Joannes Vandermeulen. 2008. User interface derivation from business processes: a model-driven approach for organizational engineering. In *Proceedings of the ACM Symposium on Applied Computing* (Fortaleza, Ceara, Brazil) *(SAC '08)*, Roger L. Wainwright and Hisham Haddad (Eds.). ACM, 553–560. https://doi.org/10.1145/1363686.1363821