# Abstract Program Visualization for Model-View-Update User Interfaces

**Geoffrey Litt**
MIT CSAIL
glitt@mit.edu

## ABSTRACT
This is the abstract

### Author Keywords
Program visualization, program understanding, debugging

### CCS Concepts
•**Human-centered computing** → **Human computer interaction (HCI);** *Haptic devices;* User studies; Please use the 2012 Classifiers and see this link to embed them in the text: https://dl.acm.org/ccs/ccs_flat.cfm

## INTRODUCTION
Lots of recent program vis work is low level. Tied directly to the source code, visualizing state at individual lines, individual variables. Makes sense for novices: small programs, need help understanding small details.

OTOH: many tasks require higher level of understanding system behavior. Eg, initial explanation of a codebase. How do we make a "whiteboard drawing" live? Some challenges:

- relevant state isn't necessarily individual variable values: need to select certain relevant attributes to visualize. Corollary: can't be as automated as low level visualization.
- Too much work on "targeted debugging", not enough on generalized understanding of a system. More specifically: if we don't know what change you might want to make to a codebase, how can we maximally equip you to be ready to make an arbitrary change?
  - Naur Programming as theory building
- combines both static + dynamic aspects: code modules, abstracted state
- Lots of non-numeric data

Abstract visualization is an interesting direction. Can be superspecific, eg algorithm animation [1, 15] or class and thread vis of [12, 14].



*CHI'20,* April 25–30, 2020, Honolulu, HI, USA



ISBN 978-1-4503-6708-0/20/04…$15.00

DOI: https://doi.org/10.1145/3313831.XXXXXXX

But a major challenge is making it easy enough for programmer to actually make it worth it [13]

This is an initial exploration. Some solution characteristics:

- redux apps [2, 3]: 1) broader than Vega, narrower than general programs, 2) imposes a worldview of abstract state and simplified event stream. THIS IS THE HEART OF THE WORK
  - In this work: focus on TodoMVC specifically, but in theory should generalize to Redux programs. (Future work: how to efficiently generate a visualization like this one)
- "Guided tour": introducing you to how the application works.

## RELATED WORK
- low level program vis
  - Learnable [16]
  - omnicode [5]
  - Projection boxes [7]
  - Theia [11]
  - Theseus [8]
- Whyline, targeted interrogation [6]
- In-situ: nice taxonomy of visualizations (but still limited to Vega, narrow domain) [4]
- Myers taxonomy [10]
- Redux / Elm
  - redux dev tools Tree viewer
- http://cs.brown.edu/~spr/research/bloom/jvlexec.pdf
- Steve Reiss overview
- Brad Myers incense
- Girba's Mondrian: a toolkit for programmers building vis [9]

### Methods
Only some initial experiments.

Redux dev tools monitor D3 + react

Show the mockups Developed 3-4 sparkline style visualizations

- line graph
- Categorical graph
- Collection dots view?
- String change view?

### Results

- get feedback from 1-2 people?
- Tried using it myself

### Discussion

### Future work

### References


[1] Marc H. Brown and Robert Sedgewick. "A System for Algorithm Animation". In: *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '84. New York, NY, USA: Association for Computing Machinery, Jan. 1, 1984, pp. 177–186. ISBN: 978-0-89791-138-2. DOI: 10.1145/800031.808596. URL: http://doi.org/10.1145/800031.808596 (visited on 05/11/2020).

[2] Evan Czaplicki. *The Elm Architecture · An Introduction to Elm*. URL: https://guide.elm-lang.org/architecture/ (visited on 05/11/2020).

[3] Simon Fowler. "Model-View-Update-Communicate: Session Types Meet the Elm Architecture". In: (Jan. 13, 2020). arXiv: 1910.11108 [cs]. URL: http://arxiv.org/abs/1910.11108 (visited on 05/11/2020).

[4] Jane Hoffswell, Arvind Satyanarayan, and Jeffrey Heer. "Augmenting Code with In Situ Visualizations to Aid Program Understanding". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI '18. Montreal QC, Canada: Association for Computing Machinery, Apr. 21, 2018, pp. 1–12. ISBN: 978-1-4503-5620-6. DOI: 10.1145/3173574.3174106. URL: http://doi.org/10.1145/3173574.3174106 (visited on 05/11/2020).

[5] Hyeonsu Kang and Philip J. Guo. "Omnicode: A Novice-Oriented Live Programming Environment with Always-On Run-Time Value Visualizations". In: *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. UIST '17: The 30th Annual ACM Symposium on User Interface Software and Technology. Québec City QC Canada: ACM, Oct. 20, 2017, pp. 737–745. ISBN: 978-1-4503-4981-9. DOI: 10.1145/3126594.3126632. URL: https://dl.acm.org/doi/10.1145/3126594.3126632 (visited on 05/07/2020).

[6] Andrew J. Ko and Brad A. Myers. "Designing the Whyline: A Debugging Interface for Asking Questions about Program Behavior". In: *Proceedings of the 2004 Conference on Human Factors in Computing Systems - CHI '04*. The 2004 Conference. Vienna, Austria: ACM Press, 2004, pp. 151–158. ISBN: 978-1-58113-702-6. DOI: 10.1145/985692.985712. URL: http://portal.acm.org/citation.cfm?doid=985692.985712 (visited on 05/07/2020).

[7] Sorin Lerner. "Projection Boxes: On-the-Fly Reconfigurable Visualization for Live Programming". In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, 2020. DOI: 10.1145/3313831.3376494.

[8] Tom Lieber, Joel R. Brandt, and Rob C. Miller. "Addressing Misconceptions about Code with Always-on Programming Visualizations". In: *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems - CHI '14*. The 32nd Annual ACM Conference. Toronto, Ontario, Canada: ACM Press, 2014, pp. 2481–2490. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557409. URL: http://dl.acm.org/citation.cfm?doid=2556288.2557409 (visited on 05/11/2020).

[9] Michael Meyer, Tudor Gîrba, and Mircea Lungu. "Mondrian: An Agile Information Visualization Framework". In: *Proceedings of the 2006 ACM Symposium on Software Visualization - SoftVis '06*. The 2006 ACM Symposium. Brighton, United Kingdom: ACM Press, 2006, p. 135. ISBN: 978-1-59593-464-2. DOI: 10.1145/1148493.1148513. URL: http://portal.acm.org/citation.cfm?doid=1148493.1148513 (visited on 05/11/2020).

[10] Brad A. Myers. "Taxonomies of Visual Programming and Program Visualization". In: *Journal of Visual Languages & Computing* 1.1 (Mar. 1990), pp. 97–123. ISSN: 1045926X. DOI: 10.1016/S1045-926X(05)80036-9. URL: https://linkinghub.elsevier.com/retrieve/pii/S1045926X05800369 (visited on 04/28/2020).

[11] Josh Pollock et al. "Theia: Automatically Generating Correct Program State Visualizations". In: *Proceedings of the 2019 ACM SIGPLAN Symposium on SPLASH-E - SPLASH-E 2019*. The 2019 ACM SIGPLAN Symposium. Athens, Greece: ACM Press, 2019, pp. 46–56. ISBN: 978-1-4503-6989-3. DOI: 10.1145/3358711.3361625. URL: http://dl.acm.org/citation.cfm?doid=3358711.3361625 (visited on 01/28/2020).

[12] Steven P. Reiss. "JIVE: Visualizing Java in Action Demonstration Description". In: *Proceedings of the 25th International Conference on Software Engineering*. ICSE '03. Portland, Oregon: IEEE Computer Society, May 3, 2003, pp. 820–821. ISBN: 978-0-7695-1877-0.

[13] Steven P. Reiss. "Visual Representations of Executing Programs". In: *J. Vis. Lang. Comput.* (2007). DOI: 10.1016/j.jvlc.2007.01.003.

[14] Steven P. Reiss and Manos Renieris. "Jove: Java as It Happens". In: *Proceedings of the 2005 ACM Symposium on Software Visualization - SoftVis '05*. The 2005 ACM Symposium. St. Louis, Missouri: ACM Press, 2005, p. 115. ISBN: 978-1-59593-073-6. DOI: 10.1145/1056018.1056034. URL: http://portal.acm.org/citation.cfm?doid=1056018.1056034 (visited on 05/11/2020).

[15] John T. Stasko. "Tango: A Framework and System for Algorithm Animation". In: *Computer* 23.9 (Sept. 1, 1990), pp. 27–39. ISSN: 0018-9162. DOI: 10.1109/2.58216. URL: http://doi.org/10.1109/2.58216 (visited on 05/11/2020).

[16] Bret Victor. *Learnable Programming*. URL: http://worrydream.com/LearnableProgramming/ (visited on 04/28/2020).