User Interface Optimization using Genetic Programming with an Application to Landing Pages

Paulo Salem

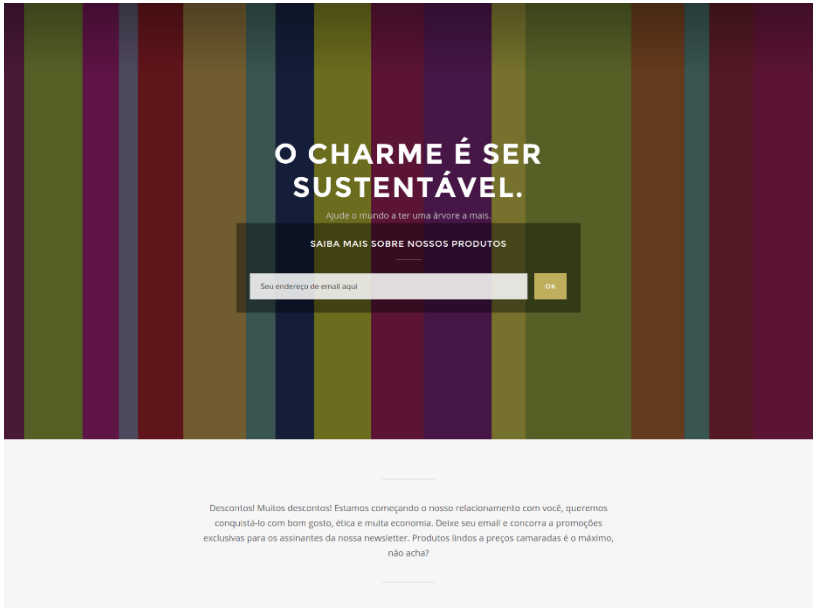
The 9th ACM SIGCHI Symposium on Engineering Interactive Computing Systems

June 26-29, 2017 - Lisbon, Portugal

DELL EMC

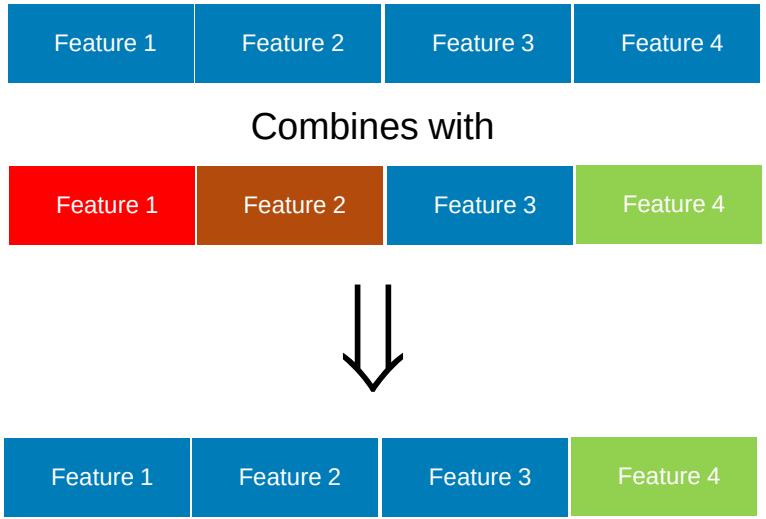
The Problem

- Given some basic **UI elements**, **generate** UIs based on a **specification** that are **optimal** w.r.t. some **performance metric**.
- Elements: texts, images, colors, components, etc.
- Working example: landing pages
  - Users are directed to a special page
  - Users are motivated to take some action (e.g., subscribe to a newsletter)
  - Acquiring users is **costly**, therefore we want to **maximize conversion rates**
  - $conversion\ rate = \frac{actions\ taken}{total\ users}$



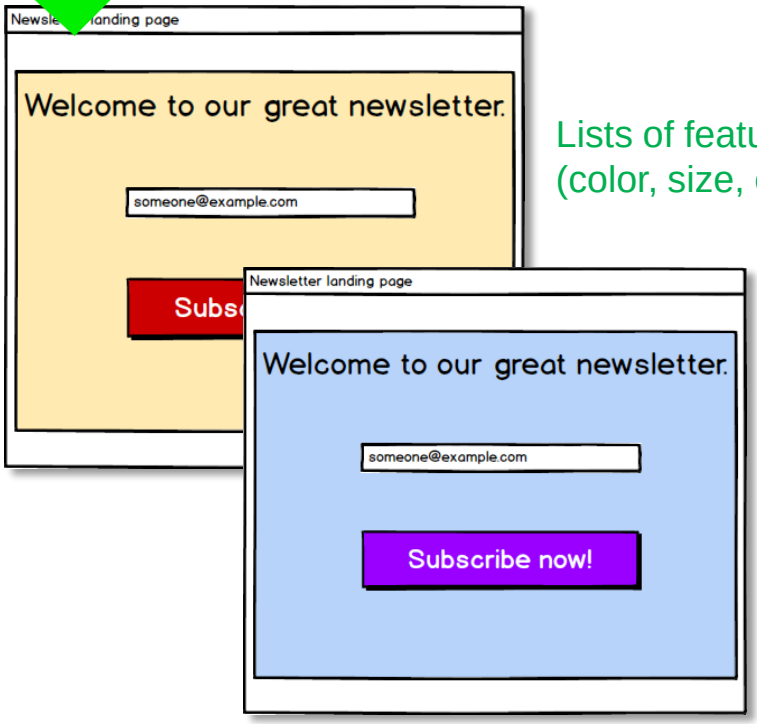
Existing Solutions using Genetic Algorithms

- UIs are **individuals** in a **population**
- Each UI's **genome** is a sequence of features
  - Color of components
  - Images
  - Texts
- By recombining and mutating features, new UIs **evolve**
- Users can **select** their prefered UIs
- Main problem: not general, cannot express arbitrary UIs



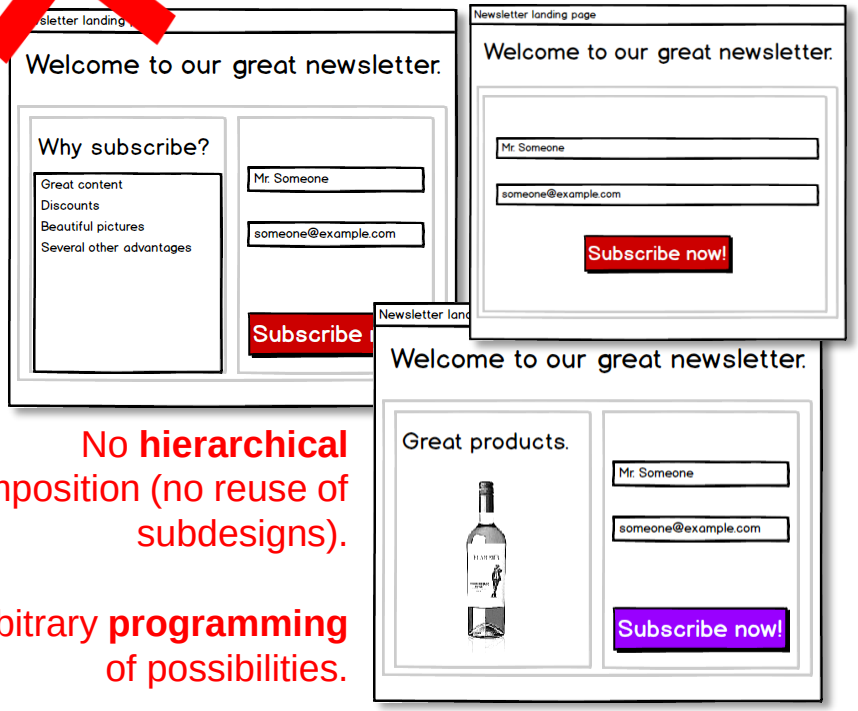
Limitations of Existing Solutions

Can express superficial variations.



Lists of features (color, size, etc.)

Cannot express intrincate UI structural possibilities.



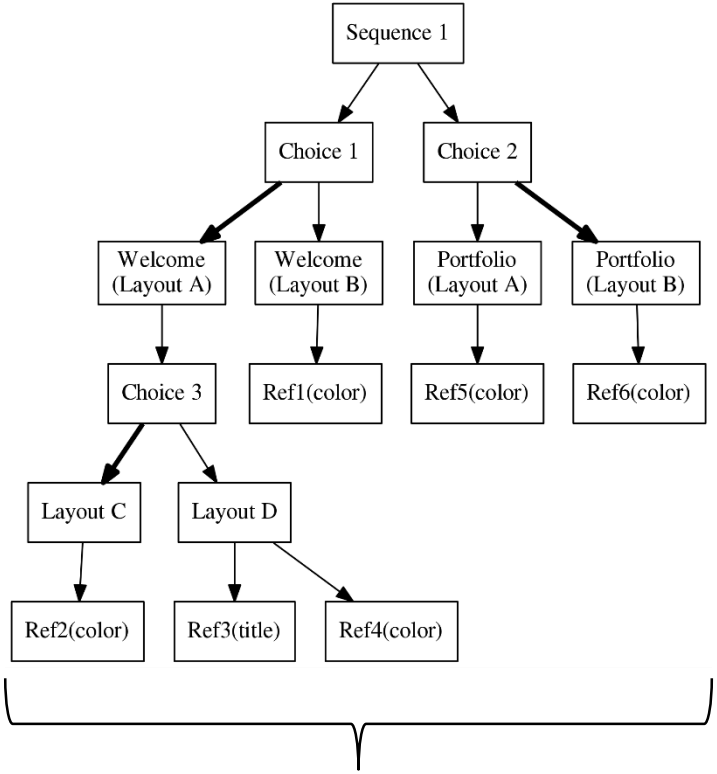
No **hierarchical** composition (no reuse of subdesigns).

No arbitrary **programming** of possibilities.

The Proposed Approach

Genetic Programming

- Programmatic specifications
  - Program expressions instead of features.
- Structured representations
  - Sections, subsections, components, subcomponents, etc.
  - Mimics the structure of the UIs themselves.
- Non-deterministic choices
- Fitness determined by the actions of many concurrent users (i.e., “crowdsourcing”)
- ⇒ **A general way to specify UIs suitable to evolutionary optimization**



possibilities can grow exponentially

UI Specification

Content elements

“This is some headline”
“This is a better headline”
“Some other headline”
Etc.

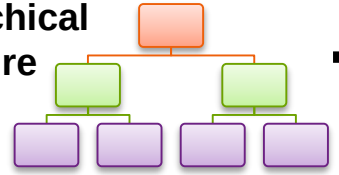
+

Components

Buttons
Input boxes
HTML blocks
Etc.

+

Hierarchical structure



+

User actions

Form submission
Clicks
Page scroll
Etc.

Programmatic UI Specification

- Specification given programmatically
- Clear definition of hierarchies and non-deterministic choices
- Domain-Specific Language in Scala

```
Include(LeadGenBlock, variation = 1,
    Sequence(
       Choice(
          Define("opacity", "0.2"),
          Define("opacity", "0.4")
        ),
        ParameterValue("title" ->
                       Reference("title")),
        ParameterValue("tagline" ->
                       Reference("tagline")),
        // (...)
    ))
```

```
<multi-value name="title">
    <text>Recycled furniture and home accessories</text>
    <text>Lovely and recycled furniture</text>
    <text>Taste, recycled</text>
    <text>Learn more about recycling with taste</text>
</multi-value>
```

Can also load content elements from XML:

- Text variations
- Image URLs variations
- Arbitrary HTML variations

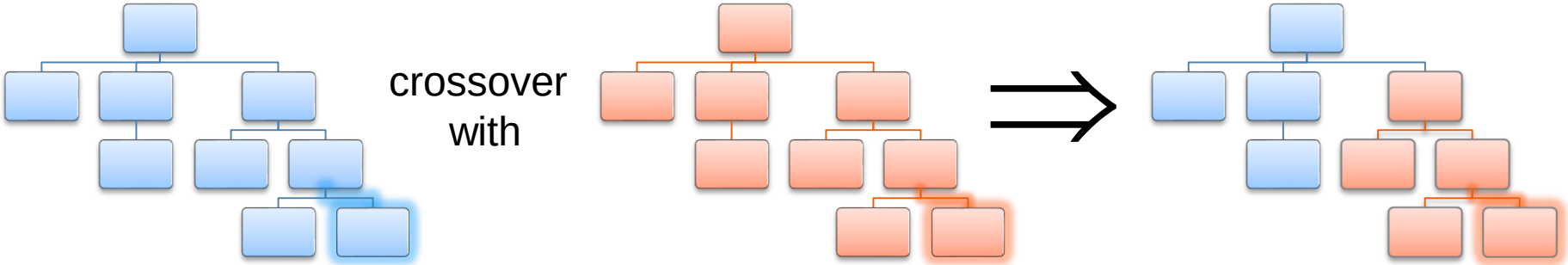
Programmatic UI Specification

Program statement	Meaning
`Sequence(a, b)`	The sequential composition of two program statements.
`Choice(a, b)`	The non-deterministic choice among two program statements.
`Define(name, value)`	Defines a variable named *name* with content *value*.
`Reference(name)`	A reference to the value of the variable named *name*.
`Include(blockName, variation, init)`	The rendering of a template named *blockName*.
`ParameterValue(name -> reference)`	Defines that a parameter named *name* has the value specified by *reference*. This must be used within an *Include*'s *init*.
`Div, B, Span, etc`	Primitive rendering operations, which allow the writing of the actual HTML that will be displayed later to the user.

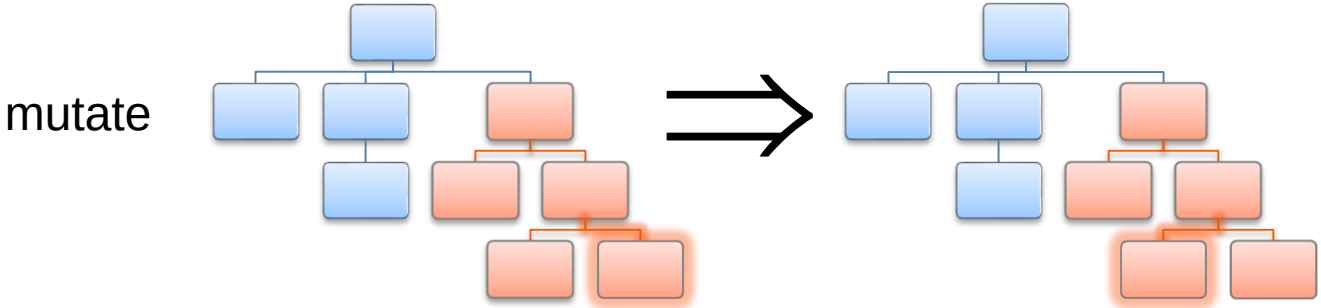
DELL EMC

UI Recombination

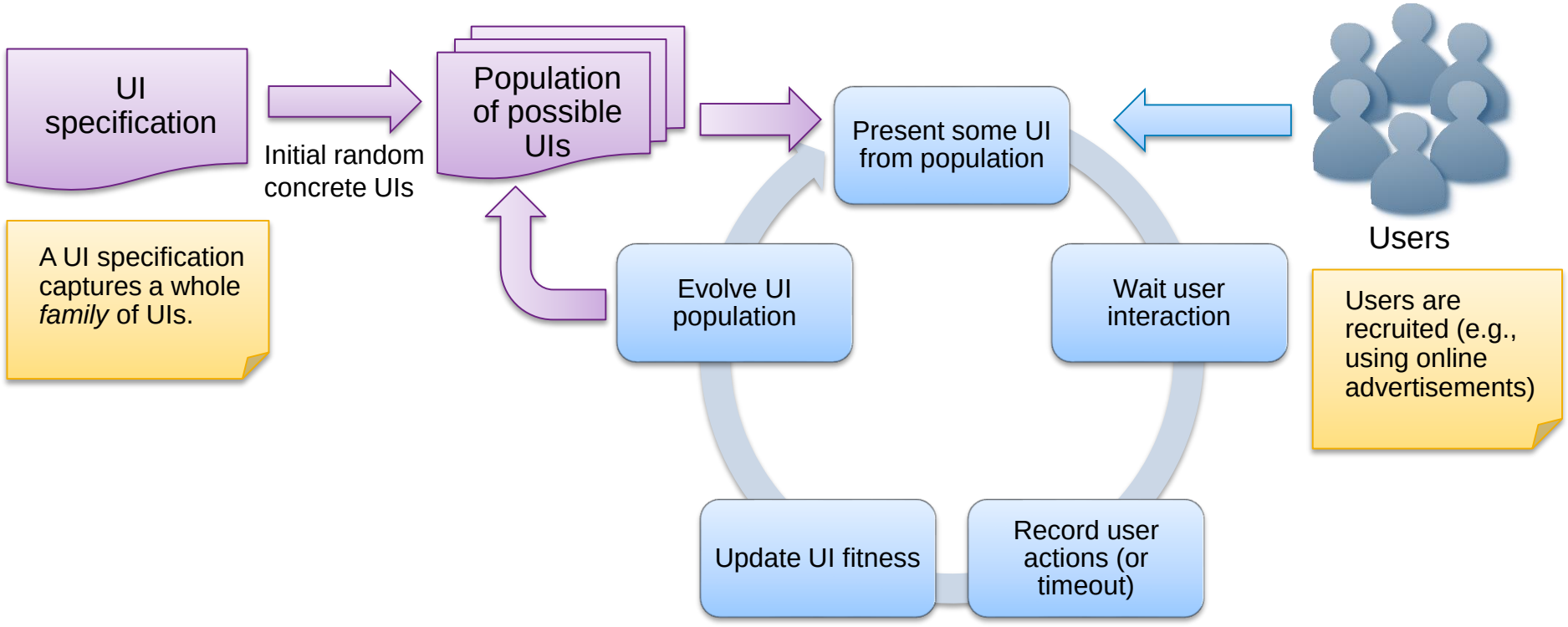
Crossover



Mutation

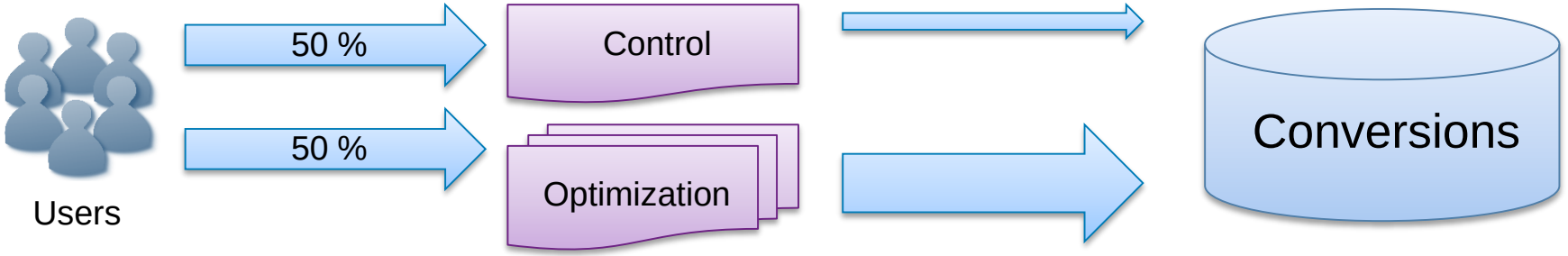


Complete Process

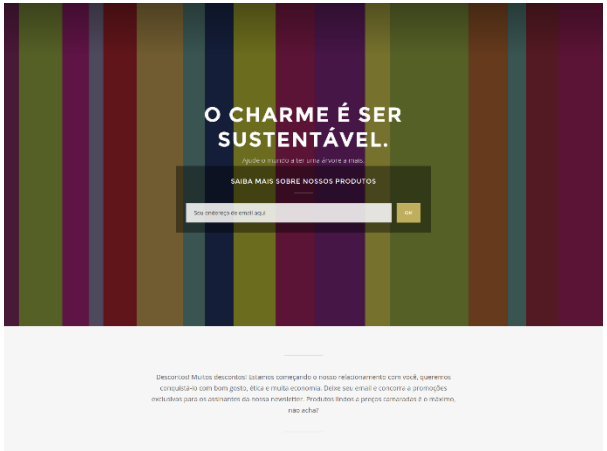


Evaluation

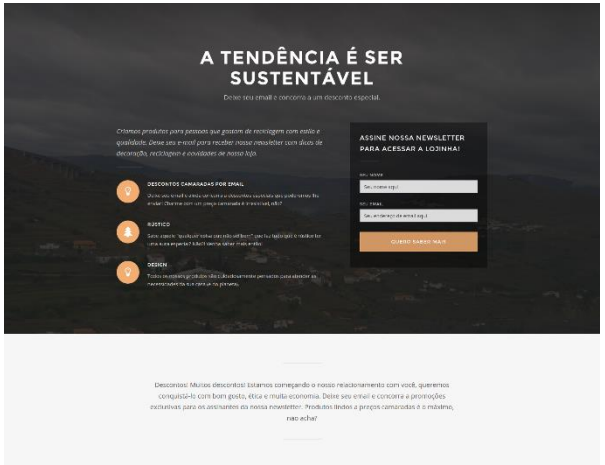
Evaluation



Control page



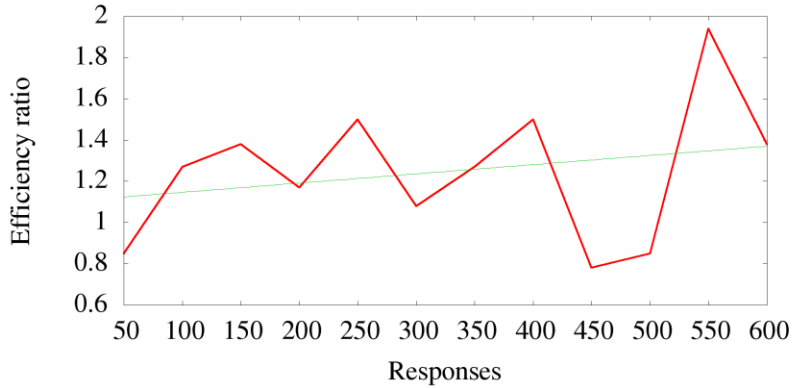
Random initial page



Best page found

Evaluation

- ~ 6^{10} possible concrete UIs
  - Combinatorial explosion
- **5,462 users** were recruited for experiment
- **20%** performance increase
  - Conversion rate (optimized) = 12%
  - Conversion rate (control) = 9.9%
- Non-numerical insights:
  - 8 out of 10 best UIs mentions discounts or monetary savings
  - However, it is not always clear *why* performance improves
- Seasonal effects (i.e., weekends seem to change user behavior).



Efficiency ratio > 1.0 ⇒ optimization performs better than control UI

Conclusion

Future Work

- More experiments, in different domains
- More independent controls, by professional designers
- More programming primitives for UI specification
  - Constraints of various kinds
- More basic UI elements to be composed
- Variations of crossover and mutation operators

Summary

- Designers must think in terms of **UI families** instead of unique UIs.
- **Programmatic**: allows for the specification of arbitrary UI families.
- Foundation for general UI optimization based on Genetic Programming and similar techniques.
- Example uses Web pages, but the approach is general to any UI which:
  - Can be built out of simpler elements
  - Can be instrumented in order to record user actions
  - Has a clear performance metric to be optimized
- Aims at **making designers more productive**, not at substituting them.


DELL EMC