Interaction Design, User Interfaces, and the Future of Model-Driven Design and Development

Prof. Larry Constantine, IDSA, ACM Fellow
Department of Mathematics and Engineering
University of Madeira
Considerations and techniques are proposed that reduce the complexity of programs by dividing them into functional modules. This can make it possible to evolve complex systems from simple, independent, reusable modules. Debugging and modifying programs, reconfiguring I/O devices, and managing large programming projects can all be greatly simplified. And, as the modular library grows, increasingly sophisticated programs can be implemented using less and less new code.

**Structured design**

by W. F. Stavens, G. J. Myers, and L. L. Constantine

Structured design is a set of proposed general program design considerations and techniques for making coding, debugging, and modification easier, faster, and less expensive by reducing complexity. The major ideas are the result of nearly ten years of research at Control Data Corporation by Mr. Constantine. His results are presented here, but the authors do not intend to present the theory and derivation of the results in this paper. These ideas have been called compositional design by Mr. Myers. The authors believe these program design techniques are compatible with, and enhance, the documentation techniques of NOS and the coding techniques of structured programming.

These cost-saving techniques always need to be balanced with other constraints on the system. But the ability to produce simple, changeable programs will become increasingly important as the cost of the programmer's time continues to rise.

**General considerations of structured design**

Simplicity is the primary measurement recommended for evaluating alternative designs relative to reduced debugging and modification time. Simplicity can be enhanced by dividing the system into separate pieces in such a way that pieces can be considered implemented, fixed, and changed with minimal consideration of effect on the other pieces of the system. Observability (the ability to easily perceive how and why actions occur) is another...
No Conspiracy, But No Accident

• The place and role of users, usability, and user interfaces in software development.

• The place and role of programmers in software development.
Do Users and the Context of Use Matter?

- Inside the cockpit of the Airbus 320...
- Video games anyone?

“Congratulations to our pilot, newly promoted to Captain!”
Why Design for Use

• A product can be a success without being useful.
• But usefulness and usability pay.
  ✓ Simpler documentation, lower support costs.
  ✓ Faster learning, lower training costs.
  ✓ Fewer errors, better data integrity.
  ✓ Improved efficiency, dependability.
  ✓ Increased customer and end-user satisfaction.

Design for use is an alternative to competing by commodity pricing or through feature bloat.
In the Beginning

Deus ex machina...

Technology at the center. User-friendly design. UI and usability as after-the-fact decoration.
Advances in Computing

- IBM 1401
- Autocoder
- Progress
  - machine language
  - assembly language
  - high-level languages
  - 3GLs
  - 4GLs
  - application generators
  - executable models
1986. What has Changed, Ed?

- Excelerator (Index Technology)

Same old, same old. But now it’s object-oriented programming. And CASE.

Just pretty pictures. Good marketing. The real stuff is database and code generation!
Why Models?

- Building models is easier than building the real thing.
- Models capture, carry, and organize understanding about a problem or possible solutions.
- Models permit exploration of problem and solution spaces.
- Models can be validated against objective criteria.
- Models can be tested and evaluated.

MODEL-DRIVEN PROCESSES:

- provide an “audit trail” of assumptions, of evolving understanding, and basis of solutions.
- facilitate tracing results back to requirements.
- enable reasoned derivation of successful solutions.
Some Holes in the Story
Where are the Users?

Even the gurus don’t get it!

In 21 classic, respected books —

<table>
<thead>
<tr>
<th></th>
<th>INDEX ENTRIES</th>
<th>TOTAL PAGES</th>
<th>% of PAGES</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UML and the unified process</strong></td>
<td>12.7</td>
<td>67</td>
<td>(2.1%)</td>
</tr>
<tr>
<td>(9 books, © 1997-2003; 3175pp)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Use cases</strong></td>
<td>18.7</td>
<td>47</td>
<td>(2.5%)</td>
</tr>
<tr>
<td>(6 books, © 1997-2003; 1913pp)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>XP and agile development</strong></td>
<td>2.5</td>
<td>8</td>
<td>(0.4%)</td>
</tr>
<tr>
<td>(6 books, © 2000-2002; 1868pp)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3 books have only 1 mention;
3 have nothing at all to say!

* users, user interfaces, usability, GUI, end users,…
Models Matter

• Models should be suited to the design task.
• UML can be used to model user interface design and interface architecture, but…
• Some design modeling is better than none at all.
• But big, clumsy, complicated models tend to lead to…
Model-Driven Everything

• model-driven user interfaces?
• model-driven engineering
• executable models
• model-driven architecture
• model-driven development
• model-driven design
• models in analysis and design
Some processes can target alternate deployment: Web, desktop, smartphone,…
Some even attempt integrated “beautification.”

Pederiva et al., “The Beautification Process in Model-Driven Engineering of User Interfaces” Interact 2007
And the results?

- Real-world app generated by OlivaNova

Pederiva et al., “The Beautification Process in Model-Driven Engineering of User Interfaces” *Interact 2007*
Getting Better, Getting Worse

• To avoid “analysis paralysis,” agile development rejects thoughtful, thorough design. No BDUF!
• Users can be shortchanged in “customer-centric” focus on features, delivered value, customer satisfaction.

“GUI-intensive projects are problematical for XP (and probably for many [agile] approaches).”
—Ron Jeffries

“[Usability] is not a ‘weak point’ [of the agile methods], it is an absence.”
—Alistair Cockburn

• But, things may be getting better -
  – Jeff Patton, agile-usability@yahoogroups.com
  – Scott Ambler, Agile Modeling
  – Cutter Agile Management “The Usability Challenge”
Still, . . .

- User-centered design is no guarantee either.
- User experience design is often misguided because user performance is what is truly important.
- Industrial design does no better than software engineering.

“It probably won a prize!”
—Donald Norman

“Focus upon humans detracts from support for the activities themselves.”
—Donald Norman
If Only Things Were Simple

Management fantasy

- The new programmer –
- The new designer –

Everyone’s fantasy

- The new user –
Points of Departure

- Use cases promising but early results disappointing.

Keys
- ✓ simplification not elaboration
- ✓ abstraction not realism
- ✓ intentions not interaction

- Essential use cases and basic process (1993).
- Widely used and practiced: Siemens, McKesson, Nortel, Navy, Lockheed, Gartner, DaimlerChrysler, IBM,…
- Full spectrum of problems: e-commerce, automation, healthcare, finance, banking, insurance, education, embedded systems, CAD/CAM, office products, …
- Integrated with varied processes: OOSE (Ahlqvist, 1996), RUP/USDP, XP, agile modeling,…
Unified Modeling?

- For data modeling you want appropriate models by skilled data modelers.
- For process modeling you want appropriate models by skilled process modelers.
- For interaction modeling you want appropriate models by skilled interaction designers.
- Different models and different skills are needed for different parts of the modeling process!

* A. Chanda, E. Allen, Rice University
Much is Asked

• Focus shifted from users and user experience to human activity and user performance.
• Model-driven inquiry grounded in systematic models of human activity. **Human Activity Modeling**
• Model-based process for user interface and interaction design. **Usage-centered Design**
• Models tailored to the needs of interaction design and designers.
• Models based in powerful abstractions and separation of concerns.
• Tools supporting flexible, seamless problem solving with all models interconnected.

"With dynamic extensions!"

**Canonical Abstract Prototypes**

* Nóbrega, University of Madeira
Human Activity Modeling

• Based in activity theory, models human activity:
  “loosely ordered collection of flexibly adaptive actions having distinct but disparate goals contributing to a shared or common purpose.”

• Simple notation, small number of concepts systematizing activity theory, linked with UML.

• Captures and organizes aspects of activity context most salient and relevant to interaction design in support of real user requirements.

“Business process modeling done right!”
- Nuno Nunes
Anatomy of an Abstraction

abstract UI components
with standardized user-centered semantics

CAPs and dCAPs: formalized abstract wireframe schematics.

content, function, layout and organization

Supports separation of concerns.
And they are “executable”!*
Prescription for Progress

• Models to capture, hold, and organize understanding of problems and solutions.
• Focus on user performance, human activity.
• Tools to support, not take over human activity.
• Fix and finish UML (notations suited for human activity and user interaction modeling).
• Use people for what people are best at: creative exploration, analysis, invention, and design!

OR...

“Hello. Are you listening OMG?”

Thank you.