Design, The “Straw” Missing From the Bricks of IS Curricula

Les Waguespack, Ph.D.
Professor & Chairperson of Computer Information Systems
Bentley University
Waltham Massachusetts

ISECON - Nashville, TN - 2010-10-28
"Ye shall no more give the people straw to make brick, as heretofore: let them go and gather straw for themselves"

Exodus
With the evolution of IS curricula over the last decade, are we teaching students to “make systems” without teaching them “Design?”
Design: Applying quality Principles in the process of creating artifacts
FROM PROGRAMS TO SYSTEMS

(1938 - 1958)

- Single-process computing (esp. Analog)
- I/O multiplexing
- Reshaping Rqmts to fit Computer Abilities
- Multiprocessing required effecting: coordination, prioritization, dependency, planning
- Need for System efficiency > Programs
From Possible to Efficient

(1956 - 1968)

Advancing developers from skill to craft

Design quality through repetition

Assembler, FORTRAN, COBOL, ...

Discipline enforced quality

Modularity, transparency, traceability, ...

Developer Education focused on reinforcing these concepts

Computer Science and SW Engineering expanded on clarity and reliability . . .
From Scientific to Commercial

(1950 - 1965)

- Shifting / Expanding Computer application from privilege: Governmental, Financial; To commerce: sales, marketing
- Computing education branched out
- Business, office systems, workstations
- Computing became "discipline-based"
- "Using" not "Developing" computing
- Design shifted to the Off-the-Shelf vendor
- Computing education not focused on system!
From Architect to “Interior Decorator”

(1980 - 2010)

- Business development is “form based”
- Developers insulated from design
- Business developers don’t learn the “art”
- But can the “interior decorator” design the “corporate structure?”
- Without access to “carpentry” and “structural engineering”
Design: Applying quality Principles in the process of creating artifacts
To paraphrase Brooks:

The difference between good designs and great ones does not lie in design methods but in the mindset of a great designer; producing structures that are faster, smaller, simpler, cleaner and less expensive – a difference approaching an order of magnitude.

The essentials are plan, in the mind, and later execution.

Design and Implementation are distinct
What do we teach?

CC2001 (CS2001) Computer Science Curriculum Volume

IS2002 Information Systems Curriculum Volume

SE2004 Software Engineering Curriculum Volume

CE2004 Computer Engineering Curriculum Volume

IT2006 Information Technology Curriculum Volume

Other curriculum volumes as needed for emerging disciplines

CC2005 The Overview Volume on Undergraduate Degree Programs in Computing
Where have we been?

Pre-1990s:
- EE+
- CE
  Hardware
- CS
  Software
- IS
  Business

Post-1990s:
- EE
  Hardware
- CE
- CS
  Software
- SE
- IT
- IS
  Organizational needs
What do we cover?

Among the 39 Knowledge Areas of computing identified in CC2005:

- Only 7 reference design as a specific professional competency
- The area definitions do not distinguish design from implementation
- Only SE addresses design principles or quality
- IS2010 has no learning units on design distinct from a specific technology
Teaching Implementation Suffices for Design?

- Implementation taught across several courses might suffice for teaching design.
- Few programs have 2 or 3 courses in systems development technology.
- Few have more than 1 in any technology.
- IS2010 no longer lists implementation (application development) as a core requirement!
Talking about vs Building Systems

- Deemphasizing Design narrows the learning experience
- Building is More than “writing code”
- Modeling is a key
  - Requirements, information, processes, transactions, applications, networks, systems
- Design is fundamental to problem-solving
- Design is a foundation and justification of systems and is essential to understanding them!
Employment opportunities

Graduates of an “about-IS” program are challenged to distinguish themselves from general business graduates.

Employers find it hard to distinguish.

Omitting design leads colleges of business to diffuse “about-IS” throughout and further obscure the recruitment potential of IS programs.
**What Portends Our Future**

- **In IS, Design fuses technological opportunity with business opportunity often reshaping or reinventing both**

- **Absent Design, computing is a contraption off-the-shelf as-is, surrendering quality and innovation to appliance manufacturers**

- **Design underpins understanding requirements, formulating models of software, business, business process**

- **Surrendering Design for IS curricula portends the demise of IS as a discipline**
What are our Options?

- Re-Energize design education in our programs
- Creatively renew conceptual pedagogy around design and implementation
- Recruit industry partners to corroborate the importance of design in their recruiting
- Explore greater partnerships in institutions with CS, SE, and CE programs
- Find ways for arts & Science electives to reinforce design across curricula
How is it that one system is more effective, appealing, satisfying and/or more beautiful than another to its stakeholder community? This question drove Christopher Alexander’s fifty-year quest to explain great physical architecture and give birth to pattern languages for building that underpins much of modern systems engineering.

How is it that so many individual stakeholders consistently recognize the same quality, the same beauty in a system? This question led George Lakoff to research the role of conceptual metaphor in human understanding.

What is essential to stakeholders’ satisfaction with systems? Fred Brooks addressed this question in No Silver Bullet: Essence and Accidents of Software Engineering.

This monograph fuses these diverse streams of thought in proposing Thriving Systems Theory by translating Alexander’s properties of physical design quality into the abstract domain of information systems and modeling. Metaphor-Driven Modeling incorporates the theory while examining its impact throughout the system life cycle: modeling, design and deployment. The result is holistic and innovative, a perspective on system quality invaluable to students, practitioners and researchers of software and systems engineering.

Les Waguespack is a computer science Ph.D., professor and chairperson of computer information systems at Bentley University, USA. Dr. Waguespack’s experience as programmer, software engineer, software architect, database architect, project manager and systems consultant underpins 35 years of teaching and research, the last 20+ years teaching object-oriented modeling and systems engineering to undergraduates, graduate students and practicing professionals.