



Software, Architecture, and Participatory Design

Stephen Rank, Carl O’Coill, Cornelia Boldyreff, and Mark Doughty

`srank@lincoln.ac.uk`

University of Lincoln



Questions

- Alexander's work in architecture (*The Timeless Way Of Building, A Pattern Language, etc.*) are widely-referred to in the software engineering literature, but not in architecture
 - Why is this so?
- Participation of the “end-user” in software engineering and in architecture are becoming increasingly important
 - What can software engineering and architecture learn from each other?



Design Patterns



Alexander and Architecture



From *The Timeless Way of Building*:

- “There is one timeless way of building. It is thousands of years old, and the same today as it has always been.

...

And there is no other way in which a building or a town which lives can possible be made.” (pp 7–8)

- “[The timeless way is] a process, which lies deep within us: and only needs to be released.” (p 14)



Reactions: Software Engineers



From Gamma *et al*, *Design Patterns*:

- “Even though Alexander was talking about patterns in buildings and towns, what he says is true about object-oriented design patterns.” (p 2)
- “[Alexander’s] work has inspired us time and again.” (p 356)
- “[Alexander’s] descriptions of how patterns generate designs implies that a pattern language can make the design process deterministic and repeatable.” (p 356)



Reactions: Architecture



W S Saunders, “Book Reviews: A Pattern Language”,
Harvard Design Magazine 16, Winter/Spring 2002:

- “ [Alexander] has little ‘cultural capital’... belief in timeless and universal human needs is considered naïve.”
- “He is—in part—a wild anarchist, but *A Pattern Language* is overwhelmingly authoritarian... its ambitions are totalitarian... profoundly paternalistic”
- “His book is based on observation, but it is really observation without methodology”



But...



From the same article:

- “*A Pattern Language* is imaginative, lively, spontaneous, and abundant, overflowing with quickly sketched, informed intuitions. Alexander keeps his eyes on the prize of particular daily experiences; this gives the book a pervasive warmth and humanity”
- “among the few books about architecture that won’t and shouldn’t go away”
- “bounty of delightful details and insights”
- “deserve[s], despite all its serious problems, to be treated as a ‘classic.’”



Reactions: (DIY) Architecture



From (erm) Amazon's reviews of *A Pattern Language*:

- “Overall, a must have for any planner, architect, or home-improver!”
- “Every home project, from designing a new house through to putting up a simple shelf, will take on richer and deeper meaning.”
- “Read this book if you're designing [a] house, working with an architect, looking for a new house, or contributing to your city's planning commission.”
- “Many times I have stopped in mid hammerswing and climbed down a ladder to consult this bible.”



Reuse and Patterns in Architecture

- Patterns are not considered architecture
- Usually considered ‘features’, small-scale reusable details (such as cornices)
- Reuse of larger-scale patterns is considered gauche; large identikit housing estates, for example, are not ‘good architecture’ (at the æsthetic level)
- Appearance

What's the difference?

- Why are architects and software engineers in such disagreement?
- Suggestion: Structuralism in social theory is out-of-date and considered unhelpful, and Alexander has no coherent theoretical foundation. On the other hand, the a structuralist view of software is part of our current view; components in a software system have deterministic behaviour and are not prone to revolt.
- Changing power structures in society belie the “one timeless way”; *power* structures in software do not often change.

What can we learn?



- Software engineering has made much use of a technique that is, at best, considered with ambivalence in its original field
- There is no direct analogy between software and buildings: there is complexity of different kinds in each field
- Exploring the areas where the analogies break down can bring out some useful ideas





Participatory Design




What is Participation?

- Involve the user community in the design and development of urban areas/buildings/software
- Most often used in public works such as urban regeneration
- Encourage 'ownership' by the community
- An approach to social development; aims to increase social cohesion

PD in Architecture



From Henry Sanoff, *Community Participation Methods in Design and Planning*, 2000

- “[Participation] is commonly associated with the idea of involving local people in social development.”
 - “integrates traditional top-down approaches with bottom-up, resident-driven initiatives to create a network of partnerships.”
 - “This collaborative involvement builds social capital.”
 - “Mediation [in conflict resolution] is a participatory process... primary responsibility for the resolution of a dispute rests on the parties themselves.”
- 

e.g: Planning for RealTM



From New Economics Foundation, *Participation Works*, 1998

- “Using the ‘Planning for Real’ process, a large 3D model of the neighbourhood is made and used by the people who live there to *show* their needs in a non-confrontational way.”
- “At the ‘Planning for Real’ exercise lots of illustrated suggestion cards are available, covering community facilities, crime and safety, the local environment, health, housing, leisure, traffic and transport, work, training and the local economy. Blank cards are also available for people to make their own suggestions.”



PD in Software



- Most often used in HCI
- *e.g.*, using prototypes with real users to drive the development of the user interface
- However...
 - Usually (so far) only used in interface development, where it can be very successful
 - Use of homogeneous user groups does not address conflicting needs of different groups.
 - There is work in CSCW to address these conflicts, but not in very-large-scale projects (NHS?)



PD: Breaking Barriers

- Traditional architecture has been the preserve of “star architects”
- PD opposes this, but can be done superficially; cf Byker Wall: “This is clearly an ‘Erskine’ building, and not something designed collectively by the Byker residents. Yet an elaborate charade was gone through...” (G Towers, *Building Democracy*, 1995)
- Designers vs Users: “us” and “them” attitudes are seen as unhelpful in PD

Participation in Action?

- (Perceived as) expensive: “two years of community involvement on an East London Estate”; can we do this with software?
- More democratic than traditional methods, which is seen as essential in some contexts
- Demystifies design, allowing lay people to break through the barriers of professional jargon and notations



Conclusions



Conclusions



- Software and architecture have many differences...
 - The use of design patterns is almost completely different...
 - ...because software engineers and architects are concerned with different things
- ...and much to learn from each other!
 - conflict resolution in Participatory Design
 - computer-supported PD in urban regeneration



Comparisons



Software Architects	Physical Architects
Concerned with structure	Concerned with appearance as well
Aim to reuse	Aim to be original
Patterns considered valuable	Patterns not always considered helpful
End result's structure invisible to users	End result's appearance very important
Collaboration-in-the-small	Collaboration-in-the-large

