Navigation

- Home
- Sitemap

Recent site activity

- Publications edited by James Coplien
- scratch removed by James Coplien attachment removed by James Coplien edited by James Coplien attachment from James Coplien
- Publications edited by James Coplien attachment from James Coplien edited by Gertrud Bjørnvig attachment from Gertrud Bjørnvig
- FeetOfMaster.html edited by Gertrud Bjørnvig
- Home edited by Gertrud Bjørnvig
- View All

Architecture as Metaphor

James O. Coplien

Bell Laboratories

cope@bell-labs.com

Martine Devos

EDS

martine.devos@eds.com

ABSTRACT

The field of architecture--building architecture--has long provided a metaphor for the work of software designers and programmers. It is instructive to look at software perceptions both of elements of the metaphor itself and the fidelity of the metaphor. Grounded theory development provides a technique for studying the practice of architecture as a phenomenon, drawing out concepts both in the construction and software fields. Carrying grounded theory techniques to their natural conclusion results in a preliminary theory of architecture as a profession, articulated as a story line around the central category of constraints.

Keywords: Grounded theory, architect, construction, constraints, story line, economics, change, process

1. INTRODUCTION

Most of the software development community uses a longstanding metaphor for a key technical project lead, the role responsible for the overall system structure and vision. Sometimes defined as the "keeper of the flame," the role of *architect* draws on parallels with the role of the same name in the building profession. The term had strong tones of metaphor from its earliest use. It is in universal vogue across many software development cultures, where it usually evokesat least in hopeful termsthe romantic image of prestigious architects in the likes of Vitruvius or Alberti.

The "Architecture as Metaphor" workshop at OT '99 was designed to investigate how the software community perceives the similarities and differences between software architects and building architects. It is perhaps important to emphasize that the goal of the exercise was *not* to do a studious comparison of the two professionsthough we of course would like to base the exercise on as well-founded impressions of the building profession as possiblebut to investigate how software architects view themselves relative to their *perceptions* of the building profession.

1.1 Prior Art

Jerry Weinberg relates that Fred Brooks asked him in the early 1960s if the term "architect" was suitable for systems people. There was concern about whether the analogy was appropriate or not, but after some discussion to flesh out the analogy, the term came into wider use following Brooks' lead. [Weinberg1999]

It wasn't until the early 1960s that several disciplines (urban design, graphic and interior design, industrial design, and engineering among them) recognized the common elements of the phenomenon we now call design. We appeal to that common understanding as a backdrop for software to take building design as a metaphor [Darkel1979].

Of all architects, Christopher Alexander seems to have been most successful at capturing the imagination of a broad cross-section of the software development community. The tie between software and Alexander goes back a reference made by Naur at the first conference on Software Engineering:

> software designers are in a similar position to architects and civil engineers, particularly those concerned with the design of large heterogeneous constructions, such as towns and industrial plants. It therefore seems natural that we should turn to these subjects for ideas about how to attack the design problem. As one single example of such a source of ideas I would like to mention: Christopher Alexander: Notes on the Synthesis of Form. [NATO1968]

Naur's prescient suggestion has played out as contemporary software development, particularly the object-oriented development community, has taken Alexander's vision of architecture as a metaphor for software development.

In other contemporary literature, Dahlbom [Dahlbom1992] argues that though software development isn't physical, we nonetheless interact with it using the same faculties as for material constructions. The architectural metaphor figures strongly in Dahlbom's arguments.

But there is an important "man bites dog" aspect to the relationship between the two disciplines as well. L. Bruce Archer writes:

> The most fundamental challenge to conventional ideas on design, however, has been the growing advocacy of systematic methods of problem solving, borrowed from computer techniques and management theory, for the assessment of design problems and the development of design solutions. [Archer1965]

So as early as 1965, computer science was already enough on its feet as a discipline to have influence on the design techniques in building construction. By the 1980s, this autonomy in computer scienceperhaps combined with a need to develop an identityhad taken the software field away from the mainstream design movement [Mitchell1988]. But by this time there was also a growing move back to artisanship and craftsmanship that brings us full circle to Naur's original premonition. Jones notes that software had moved more to local fit and craftsmanship than to a "bird's eye view" of design [Jones1988].

1.2 The Workshop

The OT conference provided a forum for a workshop to explore the question of Architecture as Metaphor. We used two techniques: group theatre, and a simplified version of grounded theory suitable for group interaction. The workshop comprised an eclectic audience, a majority of whom claimed to be practising software architects.

Grounded theory is based on asking questions about the phenomenon in question. We primed the activity by posing the following questions in the call for participation:

1. How do architects relate to cultural norms and fashion?
2. What is the relationship of architects to "true beauty" that transcends fashion?
3. What do architects produce? Do architects also implement? What is the role of as-builts?
4. How do architects deal with customization?
5. Where do architects work? What does "on site" mean?
6. What is the esteem of architects in their respective professions for different cultures? What are the correlators for esteem or value for the role of architect?
7. When do architects start? When are they done?
8. Who do architects listen to? Who do they talk to? In the typical case? In the ideal case?

We introduced the workshop as an improvised play for two players, one a software architect, and the other a building architect. The setting was to be a public park bench that favored a chance meeting between the two architects. There were many close links from the attendees to the architecture profession: some worked with architects on the building of their houses; others had family members who

are architects. Jim Coplien started in the role of software architect, and Tony Heap started in the role of building architect.

The audience was asked to observe the play attentively and to note salient *concepts* that arose in the dialogue. Each concept was captured on a card and thrown on the floor. In grounded theory, this is called *open coding*. At the end of the play, we collected the cards and organized them into *categories*. The intent was to look for linkages between the categories (which is *axial coding* in grounded theory), to further flesh out each category, and to work towards a theory of architecture (which is done in a grounded theory exercise called *selective coding*). We fell short of a full theory, but we were able to converge on seven preliminary categories.

During the play, audience members could inject themselves into the play as an actor playing a role of their choice. We had cameo appearances by software developers, by bricklayers (including one "Extreme Builder" who extolled the woes of Brick Ownership), by people having houses built, by Christopher Alexander, by a dog who was curious about the building architect and who did a nasty deed to the software architect (a dog curiously reminiscent of the OT '98 chair). The actors for the architecture roles changed out a lotthe building architect more so than the software architect, surprisingly enough.

The crowd had rather a self-deprecating tone to it, as though the architects had come to do penance or to take part in a pilgrimage of sorts that would enlighten the way to improvement. Part of this might be wrapped up in British culture (self-deprecating humour is a characteristic element of British advertising and cultural criticism) and part of it might owe to the frustration in the profession.

The crowd took strong ownership for the metaphor, which combined with the slightly self-deprecating tone, made for an open setting where few inhibitions took root. There was open but thoughtful communication both in the role-play and in the ensuing analysis exercise.

2. CATEGORIES

Grounded theory starts by identifying *concepts*, and groups them into *categories* that characterize important *phenomena* of the topic under study. These are the categories that the workshop converged on:

1. Change: How do architects deal with change? This was actually one of the last categories proposed, but it turned out to be one of the most insightful.
2. Role Interaction: Who do architects interact with? This was the first category that came out of the axial coding exercise.
3. Structure: What is the role of the architect in shaping structure? This subsumes the originally proposed category "Materials".
4. Economics: What economic processes does the architect participate in? This was originally proposed as "Funding" and then broadened.
5. Constraints: What are the fundamental constraints that limit the architect in each profession?
6. Process: This subsumed the proposed categories Planning, Division of Labor, and Work Practice.
7. Professionalism: This subsumed the proposed categories Responsibility and Liability.

Each group searched through the cards produced during open coding to find concepts relevant to their categories. Some concepts made their way into several categories. Each group produced a flip-chart summary of their results. The following sections present these results in a largely verbatim form.

2.1 Change: How do architects deal with change?

Table 1 summarizes the group's findings. It is one of the most thought-provoking summaries of the workshop.

Table 1: Dimensions of Change Category

Building	Software
Provide Vision	

Aesthetics Look Use of Materials (?Leads)	Technology Strategy (?Interfere with Developers ?)
Prototypes	
Scale models Presented to Customer	Can become the product Presented to internal people
Iteration	
Only in design, little during construction (? Moving walls/furniture) Environment changes small	Throughout design, build, test Layers º rates of change Environment change frequent?
Fashion	
Prone to	Prone to
Formality	
Yes	Less so
Mistakes	
Accountable Architect sued Visible Difficult to correct	Not architect's fault? Less accountable Opaque Perceived as easy to correct Hideable?
Regulations	
Legal Framework de jure	Disclaimers Internal

	De facto
People / Roles	
Low turnover (recruit externally) Direct Authority	High turnover (promote from within) (< developers) Indirect Authority
Driver	
Architecture making most of possibilities	Architect is policeman
Time scale	

Longer Time Scale	¬ Discipline ®	Shorter Time scale

2.2 Role Interaction: With whom do architects interact?

The Role Interaction group bifurcated the world into a client community and project worker community:

The role interaction group produced a diagram depicting architect communication in a project. They viewed the architect as often being the single point of contact to the client and as the interpreters of both the contract and architectural vision to the workers. They negotiate with the client and "reflect", perhaps artistically, on the client needs. The interaction at this level is much the same for the software and building architectan observation that would be borne out in the process category.

A main finding of another workshop I attended (Andy Moorley's "Architects are from Mars, Developers are from Venus") is that architects spend a lot of time carrying information from person to personbeing information brokers. This facet of the architectural role is corroborated in at least one other study [Grinter1999]. Cher Devey notes that architects may play both of the potentially conflicting roles of information brokers and negotiators.

2.3 Structure: What is the role of the architect in shaping structure?

Perhaps the most noteworthy thing about this chart (Table 2) is that building architects are credited with influence over all aspects of structure, whereas software architects are accorded no influence (Ø), questionable influence (?), or an even stronger version of Ø (X). The place of aesthetics and fashion are questioned for software, but software architects are given some credit for dealing with architectural style.

Table 2: Dimensions of Structure Category

Building	**Software**	**Consideration**
3	**Ø**	Shape structure to requirements

Ø	**3**	What not how
3	**3**	Infrastructure
3	**?**	Prototypes / models
3	**?**	Number of types of elements
3	**?**	Aesthetics
3	**?**	Fashion
3	**3**	Style
3	**X**	Context
3	**Ø**	Standards and Regulations
3	**3**	Fitness for Purpose
3	**3**	Forces Between Elements
3	**X**	Visibility
3	**Ø**	Geometry and Space

2.4 Economics

The results of this category appear in Table 3.

Table 3: Dimensions of Economics Category

Building	Software	Consideration
3	**3**	Costing / Estimate
3	**3**	Proposal/Tender
X	**3**	Money - Time to

		Market
3		Large Lump Development
?	**3**	Money follows "exciting work", fashion
3	**3**	Maintenance / whole life cost
3	**X**	Aesthetics
3	**X**	Materials / work costs and visibility

I showed this to someone with accounting knowledge (Sandra Coplien) who immediately recognized all the facets of this chart as key considerations of management cost accounting. She noted we might seek data in the following areas to enrich the category's dimensionalisation:

1. Cost/benefit analysis (replacement versus using what one has, which in software would relate to reuse)
2. Variable versus fixed cost analysis (which is another way of looking at large lump development)
3. Direct labor, direct materials : breaking down the analysis into parts
4. How long does it take to recap (whole life cost)?
5. Depreciation and amortization

2.5 Constraints

Table 4 depicts the dimensionality of the Constraint category.

Table 4: Dimensions of Constraint Category

Building	**Software**	**Kind of Constraint**
3		Indemnity
3	**3**	Stress / Strain
	3	Number of types of components
3	**3**	Costings / Estimates
Client	**Practitioner**	Fashion

2.6 Process

Rather than doing a dimensional analysis, the process team chose to do a simple enumeration of similarities and differences, as in Table 5. The central theme seems to be the relative maturity of the disciplines.

Table 5: Similarities and Differences for the Process Category

Similarities	**Differences**
Constraints Satisfaction	Specialization
Overall Process -- but activities are different	Standard Tools and Techniques
Management	Underlying Theoretical Basis
Modeling & Realization	Modeling & Realization
	Maturity

2.7 Professionalism

Table 6 shows the distillation of the Professionalism category, which is self-explanatory.

Table 6: Dimensions of Professionalism Category

Concept	**Building Architecture**	**Software Architecture**
Specialization	Lots in architecture	Less in software
Showing people your previous work	Essential Architects are responsible for the user-facing aspect of the building (they delegate internals)	Resumés (snigger) The software architect is rarely concerned with the GUI
Regulations and Standards	Both have 'em, but see legal below	
Attribution	Only Sometimes For groundbreaking	Only Sometimes Cf. 'about' screens + Easter Eggs

3. AXIAL ANALYSIS CONSIDERATIONS

Axial analysis so-called because it takes place along the axis of the categories is the due consideration of relationships between the categories. We try to link categories together at the level of their properties and dimensions. Here, we visit each category in turn and examine linkages to other categories.

3.1 Change

The Change category highlights the central concerns of change management that fall into the architect's domain. We also see change considerations in the Economics category, particularly with respect to large lump development. This is a central concern in the architecture work of Christopher Alexander, whose influence on software may be as profound as his influence on architecture of the built world. Alexander argues that front-loaded mortgage funding models are antithetical to piecemeal growth. Interest expense could be used to improve the houses instead of feeding bank investments. Piecemeal growth attacks this and other problems: it allows houses to grow with increasing understanding in needs. And a builder cannot achieve a true sense of accommodation and comfort in an edifice that's master planned: only in one whose growth is tailored to the needs of the edifice over time. The Change team mentioned the issue of iteration as well, noting iteration is a staple of software but that little happens in construction.

The Economics team found that both software and builders deal with maintenance and whole life cost. However, software models tend to be back-loaded: Boehm's oft-quoted figure notes that 80% of software expense goes into maintenance. Houses are more front-loadedanother change concept that surfaces in the Economics category. [Brand1994] notes that the greatest buildings are so adaptable, at a deep level, that they can serve many purposes over a long lifetime. They may change not only function, but form as well, over that lifetime: the original form does not reflect all the functions the building can support.

3.2 Role Interaction

The visibility of the architect was a recurring theme in many categories. Visibility is perhaps best suited to inclusion in the Professionalism category, which tied visibility to credibility. The wry conclusion is that building architects are into "show", while software architects are invisible. This is consistent with Alexander's view that building architects often consider art for its own sake:

> There is a strange dichotomy between the present architecture and planning professions. On the one hand, the architects are in the habit of creating completely mad idealistic utopias. These utopias often have little meaning, they are unlikely to be implemented; often no one in his right mind would want to implement them. They are personal dreams, not anchored in reality. Archigrams city on legs is an extreme example. [Alexander1969]

Visibility is an important dimension of role interaction. The Professionalism category captures the concept that the building architects are visible leaders, while that is less so in software, particularly outside the domain of interest.

This ties into the Change category with respect to the visibility of prototypesa factor in both disciplines. Visibility correlates strongly to the dimension of *focus*, which is externally directed in architecture, internally focused in software until the prototype itself evolves into product. Change also noted that mistakes are visible, actionable, and difficult to correct in building architecture; less so in software architecture, and the errors are perceived as easy to correct. Furthermore, there was speculation that it may be easier to hide the errors in software than in construction. These, too relate to professionalism, which suggests that professionalism and role interaction are closely related categories linked by many dimensions.

3.3 Structure

The use of materialsand, in particular, the breadth of materials inventorywas a recurring theme. The Structure team itself noted that the number and type of elements is a stock concern in building, but it's questionable whether this is so in software. On the other hand, the Constraints team didn't view the component inventory to be a constraint in building, but felt it was a constraint in software. The Change team noted that while builders deal with materials, software architects deal with technologywhich is certainly a consideration in building as well. Materials and work costs were also noted in the Economics category, and balancing quality and cost also appeared in the Constraints category.

3.4 Professionalism

Liability was a recurring concept in many of the discussions and it found its way into many categories. Professionalism captured legal accountability for actions: present in building architecture, and much less so in software. But the Change category listed accountability for change as a liability: present in building architecture, but not the architect's fault in software. And Constraints ascribed indemnity to building but not to software. And in Process, building architects were seen as subject to pertinent legal frameworks, whereas software architects operated under disclaimers with only internal regulations.

Aesthetics was another recurring theme, which we can view as a category that extends professionalism. In the Change category, we found the focus on aesthetics in building contrasted with the focus on technology in software (though the building profession also considers material). Yet during change, both professions are concerned with fashion. In the Structure category, Aesthetics, Fashion, and Style were all addressed. Style is a concern in both professions, but Aesthetics and Fashion were deemed questionable for software. In Economics, Aesthetic considerations were found absent in software but present in building. And in the Constraints category, there is an interesting dimensionalization of Fashion, which appears to be focussed on the client in buildings, and on the practitioner in software. This is reminiscent of the Prototype concept dichotomy in the Change category.

As previously mentioned, professionalism and role interaction are closely linked in many dimensions.

3.5 Constraints

The Constraints category addresses issues in many of the other categories as a phenomenon that often cuts across the other categories. Stress/strain, component inventories and materials are issues of structure; costings are an economic consideration; fashion and standards relate to professionalism; skills shortages and specialization are issue of roles. The Constraint category is further elaborated in the discussion of the Core Category below.

3.6 Economics and Process

These categories seem to support the other five categories, and their attributes aren't often cited in other categories. This suggests that they may be secondary categories. In grounded theory we seek a central category from which we can derive a theory, and it appears that the central phenomenon of this study is only weakly related to economics and process.

3.7 Other concepts and "small" categories

There are other more detailed concepts that didn't surface as categories in the analysis such as stress, strain, inventories, visibility, aesthetics, and technology. These concepts will be presented in bold face in the story line of Section 5.

4. CONSTRAINTS: THE CORE CATEGORY

The goal of grounded theory is to develop a theory explaining some phenomenon. The phenomenon explained by the theory is labeled as the *core category*. The core category is broad enough to cover the range of material in the study, and it becomes the phenomenon around which the other categories are integrated. The core category should form the main "story line" of the core phenomenon.

Table 7: Core Category Analysis

	Change	Structure	Professionalism	Process	Constraints	Economics	Role Interaction
Change		H	M	H	M	L	L

Structure	H		M	M	M		
Professionalism				H			H
Process		M	H		M		H
Constraints		H	M			M	M
Economics		M			H		
Role Interaction					L		

The workshop exercise was too casual and incomplete to definitively point to a single core category. However, we can develop a tentative core category and theory to develop a story line. After the workshop, we did an informal coverage analysis between the categories as reflected in Table 7. The table shows high (H), medium (M) and low (L) coverage between pairs of categories (the rows are the "covering" categories, and the columns the "covered" categories). The Constraint category has the tightest coupling to the other categories. It lacks explicit coverage of Process and Change, but both Change and Process show links to Constraints. The Process category notes Constraints are a concern common to both building and software.

5. THE BASIC STORY LINE

The job of the architect is to balance **constraints** that arise from customers, **standards**, **economics**, and other people (such as project workers) involved in the **process**. These business constraints are in addition to the more obvious physical constraints of **Structure**of making sure everything fits together. Building architects are **qualified** to manage these constraints in **specialized areas** that involve scale and the type of building being built; this is true to a much smaller degree in software.

One of the main roles of the architect is to balance the constraints between **Clients** and **Project Workers**. The building architect is more **visible** in this process than is the software architect, partly because their focuses are different. The software architect is focussed on building a robust product in terms of the **technology** in vogue, while the building architect is more directly concerned with **aesthetics** and conceivably with **economics**. Building architects are more likely to shape the structure to **requirements** than are software architects; other considerations, like the domain and solution technology, provide the structural constraints for the software architect. Time constraints are much tighter on a human scale for software architects than for building architects.

Fashion constrains both construction and software, but the sources of the constraints are more internal in software (objects, databases, etc.) than in construction. This client focus in the building profession, contrasted with the builder focus in software, parallels the development of **models** (prototypes) used in both fields to explore constraints.

Constraints such as economics and process might be viewed as liabilities, but some constraints such as **standards** and **regulations** may serve both the architect and the client by scoping the design space. Some of the constraints come from theory itself; there are many more of these kinds of constraints in building architecture than in software architecture. Building architecture has more of an underlying **theoretical basis** to support processincluding the laws of physics.

To evaluate constraints, architects build **models**. For the software architect, the goal is usually to rally the programming team around the models. The building architect is more outwardly focused. This leaves the software architect less **visible** to the client than the building architect, particularly at the start of the project (when building architects are particularly visible). Yet the materials constraints are harder for the

building architect than for the software architect, and they more seriously constrain what one can and cannot do with Structure.

Unmet constraints have repercussionsbeing sued and **legal** frameworks for builders. Software architects are less **visible** and perhaps less **accountable.** Software architects also seem to have the opportunity to return and make more fixes around (just before and just after) deployment time, involving themselves as the bearers of "silver bullets" to fix system-level problems.

The architect may also be the source or conveyor of constraints. In software, the architect is often viewed as the "policeman" who "enforces" the architecture.

6. ADDING DENSITY TO THE THEORY

A good grounded theory covers all of the categories of interest. As suggested by the results of Table 7, the workshop exercise left gaps in the axial analyses and therefore in the ability of the theory to explain all the phenomena. "Architecture as metaphor" is a large topic, and we suspect many other gaps in the analysis. Grounded theory encourages additional lines of inquiry to fill in these gaps. We have already formulated questions to add density to the theory at an OT 2000 session.

7. CONCLUSIONS

There were several surprising conclusions from the workshop. From our perspective, the most surprising conclusion was that the software industry is further along in following the architecture principles of Christopher Alexander than building architects are. Among these practices is piecemeal growth: the fact that most software efforts are in fact an attempt to modify existing software. The economic investment follows the effort, which may suggest that software follows an economic model akin to that recommended by Alexander.

REFERENCES

[Alexander1969] Alexander, C. A. Major changes in environmental form required by social and psycho-logical demands. Ekistics, volume 48, 1969, pp. 78-85.

[Alexander1977] Alexander, C. A., et al. *A Pattern Language*. New York: Oxford University Press, 1977.

[Archer1965] Archer, L. Bruce. Systematic Method for Designers. In Nigel Cross, ed., Developments in Design Methodology. Chichester: Wiley, 1984.

[Brand1994] Brand, S. How Buildings Learn: What happens after they're built. New York: Viking (Penguin Group), ©1994.

[Dahlbom1992] Dahlbom, Bo. The Idea that Reality is Socially Constructed. In Floyd et al., eds., Software Development and Reality Construction. Berlin: Springer-Verlag, 1992, pp. 100-126.

[Darke1979] Darke, Jane. The primary generator and the design process. In Nigel Cross, ed., Developments in Design Methodology, p. 177. Chichester: Wiley, 1984.

[Grinter1999] Grinter, R. E. (1999) "Systems Architecture: Product Designing and Social Engineering." ACM WACC '99, San Francisco, California: February 20-22, 1999. 11-18.

[Jones1988] Jones, J. C. Softecnica. In John Thackara, ed., Design after Modernism. P. 219.

[Mitchell1988] Mitchell, John. The product as illusion. In John Thackara, ed., Design after Modernism. P. 211.

[NATO1968] Naur, P. and B. Randell, eds. Proceedings of NATO Software Engineering Conference, Garmisch, Germany.

[Strauss+1998] Strauss, A., and R. Corbin. Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. London: SAGE Publications, ©1998.

[Weinberg1999] Personal interview with Jerry Weinberg, 31 May, 1999.

Sign in|Recent Site Activity|Report Abuse|Print Page|Remove Access|Powered By **Google Sites**