The Application of Object-oriented Software Concepts in Architectural Pedagogy

Oren Lieberman

Keywords
object-oriented, aspect, subject-oriented, concern spaces, reusability, abstraction/compression, encapsulation, maintenance

Introduction

Architecture, a complex discipline that involves many people and things and the relationships amongst them, requires a pedagogical approach by which the student, even in her first year, must be able to think 'complexly' across many *subjects*. The object-oriented analysis and design software programming paradigm, which models complex 'realities', or 'models the way *people* understand and process reality'[1], holds promising concepts for architectural education.

It is not my intention to extract slavishly all possible concepts from object-orientation (OO) and accept them as a 'recipe' for educating the architect. Indeed, one of the reasons I find OO so elegant is that it provides a strategy, a non-prescriptive framework, with which both teachers and students can explore their own architectural investigations. It also provides the possibility of a common language, offering a structure in which, for example, certain standards can be measured within departments, or with which we can negotiate compatibility across different national credit systems to facilitate and encourage cross-cultural (border) exchange.

Architectural Education

In most institutions, architectural pedagogy consists of two strands: the design studio, and everything else. The latter consists of courses often taught by teachers who are academically trained but not necessarily architects, whereas professional architects usually teach the former. The assumption is that the subject matter taught in the 'other' courses will be synthesised by the student in her design work. But there is typically a lack of an explicit and supported across-the-board integration of courses in which students would learn something in history or theory or environmental sociology or even structures and would consider it as a constituent part or aspect of their studio project. This is partially due to the ways in which the separate strands are taught – fundamentally different teaching methodologies – but also to the fact that students are not being helped to think in an integrative manner. The different areas of architectural knowledge both within and across the strands need to be more clearly associated for the student, and towards this end, a commensurate model of description, and therefore of perception and thinking, needs to be introduced. Within the design stu-

dio, such a model should also encourage us to formulate design projects, which *decompose* along terms other than scale.[2] It should not only reflect the complexities of the praxis for which the student is being prepared, but also provide space for the intentional questioning and 'subversion' of some of praxis' circumscriptions.

Object orientation

Part of the appeal of looking to OO for this model lies in its emphasis on integration of interconnected entities, of *objects*. Martin and Odell define it thus:

… OO is a way of organizing our thoughts about the world … based on types of things–or object types–in our world. We can define the attributes of these object types, operations that are performed on these object types, rules based on these object types,… and so on. … Object orientation, then, provides an index of our knowledge.[3]

Integration not only occurs across various objects and systems, such as subjects and teaching methodologies in the different strands of an architectural education, but also between analysis and design. Because both analysis and design can be modelled using the same foundation of concepts, OO allows us to generate one from the other. The studio project has been taught as consisting of analyses along various contextual "dimensions," (be they historical, institutional, phenomenological, environmental, sociological, or economical) and design work as synthesis. The latter is usually restricted to the projections of ideas about buildings through scaled drawings and representational models, renderings of spaces, or animations/walkthroughs of spatial sequences in time. But just as the two strands noted above can remain disjointed, a break between the analysis part and the design part of the studio project can occur whereby the *what* and *how* of the analysis are not understood to be already a part of the design. This break leaves students again staring at the white page and due to time constraints, projects end up lacking both the breadth and depth possible. OO's integration is relevant not only in associating the two strands of architectural education with each other, but also the different parts within each strand.

Many aspects in both project management and building construction are being integrated using an object-oriented approach. It is seen as increasing overall efficiency (and therefore as cost cutting), supporting adaptability over a project's life and allowing easier exchange amongst various partners and experts[4]. In education, object-based approaches to teaching of CAD are being investigated as well:

[it is necessary] that cutting edge advances in Computer Aided Design & Construction (CIDC) [sic] and Automatic Data Collection (ADC) technologies are rapidly assimilated and embedded in curriculum development to prepare undergraduates for a new age of data generation, management and integration. Beyond representing emerging and continually shifting frontiers for hardware and software, current developments in IT are beginning to challenge the cultural and pragmatic traditions, which have shaped built environment professional discipline boundaries throughout the twentieth century.[5]

However, OO concepts are not currently being investigated as a model for understanding and creating pedagogy. Within the limited scope of this paper, I will therefore introduce a 'global' rather than a 'local' level. By the former, I am referring to the structuring of an entire architectural curriculum (from history to drawing to theory to structures to design to aesthetics etc.) so that each strand, and each subject, has an explicit *operational* relationship to another, i.e., so that each can be explicitly "mapped" onto the other. By local I mean the teaching of either a specific subject area (e.g. CAD), or the introduction of a design project in which OO concepts are used to encourage stu-

dents to think more complexly about strategies of architectural production.

Following the OO paradigm, the teaching of architecture should have an expanded notion of its *object* of study. In OO, objects are instances of *concepts*. They can be tangible (e.g., house, drawing, door, pencil, bed, ruler) intangible (e.g., community, neighbourhood, time, security, critique), roles (e.g., architect, teacher, client, student, structural engineer, external examiner, occupant), judgements (e.g., good design, beautiful house, comfortable room), relational (e.g., partnership, ownership, employment, taught), as well as events (e.g., passing an exam, graduation, mortgage approval). By defining architecture explicitly in such an expansive way, modelled on ways in which we conceptualise the world, teaching can be framed more holistically.

Objects contain within them both structure and behaviour.

The word structure *is a visual, spatial metaphor that refers to a static vision of how objects are laid out in space. Structure can specify various object configurations, such as* employees, documents, *and … designs. In contrast,* behavior *refers to how our world changes over time. For example, behavior can hire an employee or tell us that the employee has reached retirement age.*[6]

In the next sections, I will present the teaching of architecture through concepts used to describe both structure and behaviour in OO.

Structure

Class

A *class* defines a set of objects, which might be a member of many different classes – a phenomenon called *multiple classification*. For instance, the object Perspective can be taught in the *classes* of instruction including History, Theory, Presentation Skills, etc. In addition, class membership can change over time; this is referred to as *dynamic classification*. Due to a change in pedagogical objectives, Perspective might cease to be taught in Theory. These classes then breakdown into *instances* which might refer to different types of presentation (drawings, models, digital_images, etc.). This relationship of class to instance includes the concept of *inheritance*: properties of a class are inherited by its sub-classes and instances. At the level of abstraction of, say Presentation Skills, the property of *(re)presenting information to another* is also a property of its instances drawings, models, and digital_images, even though that information will be presented differently (i.e., have a different *value*). Similarly, drawing could be seen as a class as well, and transmits (propagates) certain properties to freehand_drawing, diagramming, and construction_detailing.

Domain

Domains are defined as a collection of objects in a particular area of interest. Domains are closely related to *concern spaces*, which for the software programmer or systems analyst provide a mechanism to concentrate on a specific area of investigation, *encapsulating* it in order to exclude irrelevant information, both protecting and isolating it. In the teaching and learning of architecture, there is very little which might be considered extraneous; often the most inspiring of objects lie outside a specific area under study. But we do need to be able to encapsulate certain domains in order to focus expertise, hire appropriate staff, communicate about *specific* issues in education, and to help students see how parts of learning come together to form the whole course. Domains help us to specify priorities as well; the domain of the final year might include objects such as programme_development, media_choice, and generally independent_work, whereas in first year the objects might be programme_given, presentation_in_pen, and biweekly_tutorials. In addition, encapsulation of subject areas enables individuals or smaller groups rather than an entire departmental staff to reach

certain decisions concerning curricula. By delineating domains more clearly, we can design the necessary links with which to integrate them.

Associations

In OO, objects are associated with one another through *mappings* and *relationships*. According to Martin and Odell,

[r]elationships enable us to join objects into units called tuples, or links. Furthermore, using these tuples enables us to map the objects from one type to [sic] object to another – and back again.[7]

A tuple refers to a link between *n* – number of objects; there are couples, triples, quadruples, etc. The link (teacher_Lieberman, student_Jane McGregor)[8] is an example of a teaching-related couple; (history, studio_project) or (history, structures) are *instances* of an underlying general relationship between areas of study. If we expanded and abstracted the latter, Architectural Education Integration {(course-1, course-2), (course-1, course-3), … (course-x, course-y)} can be understood as the set of possible couplings of different areas of learning. Architectural Education Integration (AEI) is an object in its own right, and might be further generalised: AEI (course-1, course-2, course-3, … course-n) denotes the integration of all areas of study. There are several common forms of relationship: *classification*, through which both drawing and model_making are related to Presentation Skills; *generalisation*, through which lecture_courses and individual_tutoring relate to teaching_methods, and *aggregation*, through which attendance, submissions, and design_work relate to a pass.

Mappings

Inherent in these links, which "treat connections as related wholes" (Martin and Odell) are *mappings* which traverse the distance between related objects; in mathematical terms, mappings are *functions*, as in: y=f(x). In the (teacher_Lieberman, student_Jane McGregor) relationship one type of object is mapped (assigned to) the other: a *student* mapping assigns Mr. Lieberman to Jane; the reverse also simultaneously occurs: a *teacher* mapping assigns Jane to Mr. Lieberman. In one of the AEI relationships, (history, structures), history might map to structures through the teaching of a *chronological*, i.e. historical, approach to the ways in which structural elements are put together in a building considering the influence of one structural *temporal* event on the next. In the reverse direction, structures might map to history by looking at the way in which history is itself 'structured'. What becomes apparent is rather serious play at work: any course can and indeed should be mapped to any other to begin to investigate and evolve curricula which are adaptable to changing circumstances of pedagogical intentions, benchmarking, i.e. teaching objectives and expectations, staffing, and student make up.

Behaviour

In OO parlance, *object state* is

… a collection of relationships an object has with other objects and types.[9]

An object's state changes when the relationships it has are created or discontinued. For example, the way in which history of architecture is taught depends on its relationship to various other *objects*: available staff, perceived importance (say, of a particular period), current historiographical discourses, techniques (lectures, slide presentations, discussion groups), etc. When one of these changes (e.g., a teacher retires and is not replaced) the *state* of history of architecture in a program changes and must be adapted to by other objects, be they people, additional courses, time schedules, etc. By explicating the relationships throughout a system, the OO model helps to project repercussions of, and thereby responses to, changes within a curriculum.

The behaviour of objects consists of other types of objects, one of which is the *event*, a 'noteworthy' change in state. Whether the state change is noteworthy or not depends on one's perspective: if the teacher of a particular CAD program resigns and is replaced, the resignation is an event for the department (relationships have been formed, a replacement needs to be found, etc.) but not necessarily for a student learning the CAD program.

Behaviour also consists of *operations*, which are the processes that change state. The teacher's state changed due to an operation such as being offered_better-paying_job. Encapsulation, discussed above with regard to domains, plays a significant role in operations. Here it refers to an object's 'protection', which regulates whether other objects have access to its information. Specific operations are associated with particular types, which mediate that access. For instance: a teacher might design and implement a way of teaching a particular design theme which she does without 'interference' from other staff members; the operation review, however, accesses the results of that design course, a critique ensues, and the course is modified accordingly. Events are indicators of changes of state and operations are the *agents* of state change. In an institution such as university, the operation review cannot occur without an agreed upon *method* of processing that operation. This method differs depending on the level at which the review occurs (encapsulation at work here as well). At the departmental level, the process of review includes methods of informal discussion, formal, minuted academic policy group meetings, student response questionnaires, internal staff appraisals, etc. Other methods are invoked at the level of the faculty and still others at the university level.

Aspect-orientation

The OO paradigm as used and developed over the years has been extended in order to cope with limitations which have arisen.[10]

. … objects are limited in their ability to modularize systemic concerns that are not localized to a single module's boundaries. […] these concerns tend to cross-cut the system's class and module structure. Much of the complexity and brittleness in existing systems appears to stem from the way in which the implementation of these kinds of concerns comes to be intertwined throughout the code[11].

Others have referred to this necessary 'intertwining' as 'tangling' of code which needs to be integrated into a program but cross-cuts those modules which are decomposed along functional lines. Kicszales et al point out that all languages, object-oriented included, are

… all rooted in some form of generalized-procedure […] which break systems down into units of behavior or function. This style has been called functional decomposition. The exact nature of the decomposition differs between the language paradigms of course, but each unit is encapsulated in a procedure/function/object, … as a functional *unit of the overall system.*[12]

Kicszales et al introduce here *aspect-oriented programming* in order to understand and provide aspects, or properties, to a system "that affect [its] performance or semantics" by *cross-cutting* its functional components. In programming parlance,

[a]spect-oriented programming and related approaches are "multi-dimensional" in that they enhance object-oriented programming by providing separation of concerns along additional dimensions, beyond "class."[13]

This is a development towards even greater holism in the way we model complex situations, towards a way to combat the "tyranny of the domi-

nant decomposition" (Tarr et al). Aspects in architectural education are those things that cut across the ways in which we modularise an educational system, which cut across specific subject areas and methods of teaching. For instance, the property, or aspect, of an ethical_viewpoint should be implicit (at least!) in all subject areas. Another aspect might be developed which would *weave* together seemingly irreconcilable viewpoints of absolute fantasy in design and requirements of construction to enable a student to hold both viewpoints simultaneously and develop a project which need not compromise one with regard to the other. Another aspect of the student's education concerns the amount and depth of simultaneous assignments, whether they be essays for history, details for structures, or diagramming exercises for their studio project. This aspect, 'workload,' is in each subject but needs to be understood as a variable which cross-cuts all areas and can help teachers reduce undesired over-extension by the student. If thought about as an aspect of study, workload could be coordinated along its *dimensional line*, allowing concentrated work in different subject areas at certain times.

Conclusion

"The attempt to construe experience as an objectively analyzable datum, or to construe the whole of reality in terms of a system of logically related propositions is a betrayal of experience."
Gabriel Marcel

This paper introduces into the teaching of architecture object-orientation as a conceptual framework which corresponds to the ways we perceive, comprehend, and situate ourselves in the world, and not as a method to analyse and quantify every credit taught. For those involved in architectural pedagogy, it promises a way of thinking about curricula which reflects the complexity that is architecture, and a way to increase the level of creative (and not merely reactive) adaptability to changing circumstances in education and in the profession as whole. Object-orientation is not a panacea; as we develop implementations, we must remember those very important aspects of 'meaningfulness' and 'significance' that will guide the infiltration of these concepts into the domain of architectural education.

Notes

1 Martin, James and James J. Odell. 1998. *Object-Oriented Methods: A Foundation*. 2nd ed. Upper Saddle River NJ.:Prentice Hall. p. 4
2 Typically, projects given in design studio differ only in terms of the size of the program and the building. Implicit is the assumption that by incrementally increasing scale from project to project over the course of the student's education, complexity is being increased as well. This, however, is not necessarily the case, and other criteria need to be investigated.
3 Martin and Odell, p.2.
4 For a discussion of such exchange possibilities in the IPDB – integrated project database – see Amor, R. and I. Faraj (2000) Misconceptions about an Integrated Project Database. Paper presented at *National Conference on Objects and Integration for AEC*. Watford, Herts, UK March 13-14
5 Castle,Graham; Paterson,Graham (2000): Computer Aided Design Informator (CADI) an Object Based Contextual Approach to Teaching and Learning for the Built Environment. Paper presented at *National Conference on Objects and Integration for AEC*. Watford, Herts, UK March 13-14.
6 Martin and Odell, p. 10
7 Ibid., p. 38
8 I have retained the OO notation of these links: (object-1, object-2, … object-*n*).
9 Martin and Odell, p. 112
10 Due to the limited scope of this paper, I will introduce just one of these extensions; in further work, others will be investigated as to their relevance to architectural education.

11 Overview of Aspect-Oriented Programming. Xerox Corporation 1998. http://www.parc.xerox.com/csl/projects/aop/overview.shtml
12 G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. Loingtier and J. Irwin, *Aspect-Oriented Programming,* in Proc. European Conference on Object-Oriented Programming (ECOOP), Finland. Springer-Verlag LNCS 1241. June 1997
13 Peri Tarr, Harold Ossher, William Harrison, and Stanley M. Sutton, Jr. "N Degrees of Separation: Multi-Dimensional Separation of Concerns." In Proceedings of the 21st International Conference on Software Engineering (ICSE21), May 1999.

Oren Lieberman
Department of Architecture and Building Science, University of Strathclyde
oren.lieberman@strath.ac.uk