

Emergence of Software Architecture: Modules, Platforms and Networks

Our research project seeks to understand the logic and patterns of emergence of software architecture from two complementary perspectives. One: we view the emergent architecture as a network of connections among a set of components (modules) that interoperate across one or more platforms. Two: the emergent architecture is the result of moves and countermoves by different companies supplying these modules and platforms; and as such the architecture is dynamic and evolving.

1. Software architecture emerges in a dynamic network of components and platforms. In this view, software systems are a set of components that can be linked together to deliver a particular business requirement; they can be represented as networks. Under this network representation, nodes denote components and links represent dependencies between components (Barabasi, 2002). We adopt the network perspective to model several networks of interrelated components across different operating environments and across different markets. Our aim is to develop and compute indices of modularity and interoperability that are important building blocks of analytically describing software architecture. Interoperability, in particular, is an important metric but is often solved on a case-by-case basis. It is the overall interoperability of a collection of applications that solve a particular problem that is important and no general index exists today. Modularity is another metric that is important for firms dealing with uncertainty, as firms need this measure to make and manage trade-offs because modularity creates options but at a cost.

2. Software architecture emerges as a set of moves and counter-moves by major software companies. Software is distinct from other sectors because the products are launched to interoperate with a specific set of complementary products. This distinction underpins the business strategies of software companies as distinct from strategies in other settings (Shapiro and Varian 1999). Software companies launch their products with one of two intents: 1) to make their product as modules that subscribe to a platform architecture that is specified by another company; or 2) to become a platform architect themselves – through middleware services – and gain support of a network of complementary developers to establish their platform architecture.

We focus on four major categories of moves by software companies. These moves are interconnected, yet distinct lenses to understand the emergent software architecture. *Acquisitions* made by software companies to change their product scope. *Joint ventures and alliances* (technology licensing, marketing agreements, joint R&D, minority equity investments) to acquire complementary resources and could potentially redefine software architecture. *Patents* awarded to the software companies, which serve as a distinct technology-based capability; these patents can be used as offensive and defensive moves to influence the emergent software architecture. *Product launches* – either as a module or as a platform; the stream of product launches reflects the specific moves made by the companies to influence the emergent software architecture.

Software Architecture and the Role of Design Operators

We adopt and extend Baldwin and Clark (2000)'s set of six core design operators to understand emergent software architecture. The core operators are: *splitting*, *substituting*, *augmenting*, *excluding*, *inverting* and *porting*. *Splitting* takes a single-level design with interdependent parameters and converts it into a hierarchical design with a core set of independent modules. *Substituting* allows designers to choose among competing designs to find the component that best meets their needs. It forms the basis for competition and hence is affected by the economic system surrounding it. Substitution opportunities arise especially after splitting. *Augmenting* means adding a module and *excluding* means leaving one out. The *inversion* operator allows the designer to make public previously hidden information about modules. This need arises when a particular module starts to be reused. *Porting* permits a system to be mapped from one context or infrastructure instance to another – e.g. Windows to UNIX.

We add two more design operators to their list that we believe has relevance and importance in the software sector: these are *interoperability* (Madnick 2002) and *linking or preferential attachment* (Albert and Barabasi 2002). Interoperability refers to the ability of two or more systems or

components to exchange information and to use the information that has been exchanged (Madnick 2002). One way to interoperate is through porting. The other option is middleware. Using the middleware, a system is able to communicate with other systems using the same interface. The key difference between interoperation and porting is that in the case of porting the interfaces exist but are not published.

Two components are linked if they depend on one another to deliver business functionality. Linking can occur either at random or preferentially. Preferential attachment refers to the assumption that the likelihood of a component connecting to another increases with the components degree (number of pre-existing connections). This is a useful operator to understand the decisions to launch products within a particular architecture. During this study, we seek to test the efficacy of these two 'additional' design operators to the software sector but also to settings beyond software.

Research Questions on the Emergent Software Architecture

1. *Stock market reaction to major acquisitions.* How does the stock market reward companies when they acquire other software companies? Are acquisitions to augment the product portfolio valued more or less than acquisitions to port a program to a different platform? Can we gain new insights on software mergers through the lens of software architecture?
2. *Stock market reaction to major alliance moves:* How does the stock market reward major alliance moves by software companies that influence the emergent software architecture?
3. *Pattern of product launches by component players and platform players:* What are the characteristic moves made by companies that focus on being a module player? How does this differ from those seeking to establish middleware and platforms? How has the network of relationships between the different set of players changed over time? How has this network impacted the emergent software architecture?
4. *Interoperability in software:* How do the modularity and interoperability scores of software networks change over time? What are the key determinants of these changes?
5. *Network evolution:* Does the software industry exhibit a winner-take-all characteristic through the emergence of dominant platforms? Can we apply some recent developments in network dynamics to understand the evolution (Barabasi 2002)?

Research Design

We have adopted a research design that involves both primary and secondary data. The secondary data that we have assembled include (1) the data on software sales by various market categories provided by IDC Corporation for the years 1990-2002; (2) the data on software mergers, alliances, and other major relationships; (3) data on patents from the NBER database; and (4) data on product launches from various public sources. We are coding the data using a set of guidelines reflecting the design operators. The two perspectives that we adopt will inform the academic disciplines of information systems, management, and software design. The role of information systems in determining the emergent architecture via its connections to one another can be systematically established. General managers can understand the effectiveness of design operators via the lens of mergers and acquisitions and JV's. Finally, software community will benefit by understanding the migration of value along the layers of the industry stacks and how they interconnect.

Researchers' background

Professor N.Venkatraman's (venkat@bu.edu) research has made contributions at the intersection of strategy and information technology for the last two decades. Recently, he completed a set of papers on network evolution in the video game sector. He is interested in software industry as a prototype of networked markets. Professor Bala Iyer has been conducting research on enterprise architectures for the last five years. Professor Chi-Hyon Lee specializes in applying network analysis techniques to IS research questions. Iyer and Lee proposed and tested a set of metrics on modularity and interoperability for software using data supplied by IDC.