Keeping Architectures Relevant: Using Domain-Driven Design and Emergent Architecture to Manage Complexity and Enable Change

by Brandon Satrom and Paul Rayner

**Abstract**

Too many systems seem to become legacy upon release, while some never even have a chance to move into production before they are undermined by the calcification of unmet expectations and mismatched domain needs. Regardless of the design effort early in the lifecycle, neglecting the domain model and producing inflexible design results in the increasing irrelevance of the initial architecture of a system. The accidental complexity of that system rises and communication between developers and customers deteriorates. Changes and new features become more difficult to accommodate as the richness and value of the system's essential complexity is eroded. Sustainable and successful software development is all about managing complexity and enabling change, and successful architects create designs that clearly address both.

Architects, domain experts and developers collaborate to mitigate complexity through strategic modeling and design. This requires a focus on the core domain and the continuous application of germane design patterns. Ongoing effort should be expended on defining and refining the domain model through the establishment and exercise of a language that everyone shares. The development of this ubiquitous language, along with the use of domain-driven design techniques, enables business problems and their solutions to be expressed through rich domain models that are both meaningful to business experts and executable by the development team.

Keeping our architectures relevant also means enabling change. As architecture is allowed to emerge, evolve, and mature, it becomes a true reflection of the deep understanding of both domain experts and developers. Architects who expect their initial design to evolve, and who design with evolution in mind, create architectures that deliver a strong competitive advantage to the business.

**Reader Takeaways**

1. The establishment of a ubiquitous language, which removes the built-in translation layer between domain experts and the development team, is key to relevant modeling.
2. Domain-driven design enables the articulation of a distilled architecture through models that mitigates complexity while remaining relevant to the business and clear to the development team.
3. Architects must collaboratively drive architectures which emerge, evolve and mature in order to deliver systems that improve in their ability to respond to the changing needs of the business.