

System Transformation: Be Careful or You Will Not Get What You Asked For

By Larry Goldberg and David Pedersen

Introduction

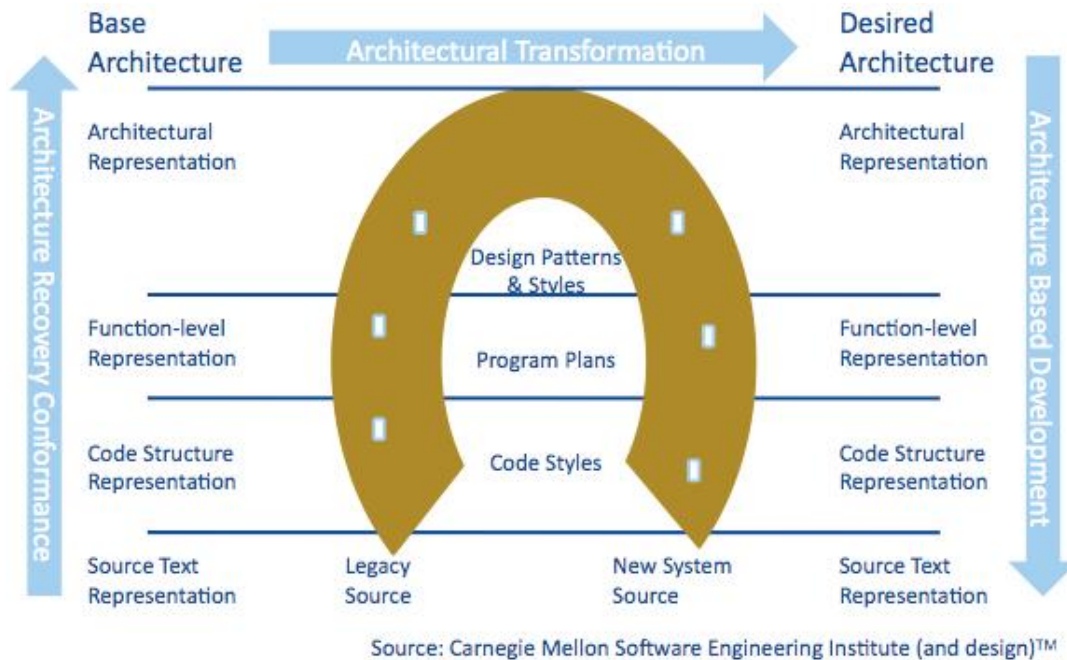
Companies will sooner or later have to re-invent their legacy systems, add improvements to them, or significantly change their capabilities. This may be driven by business re-engineering, process improvement efforts or replacing an out of date system that can no longer be supported. All of us who have been involved in these projects know that these are amongst the riskiest of projects. We set out some ideas to mitigate that risk, and potentially save money.

Legacy systems typically have little or no existing documentation and often the staff that built the system has moved on. Companies often try to rediscover the legacy business logic but these projects typically fail to deliver adequate value and can be expensive. This is because popular short cut “approaches” involve “mining” the business logic, and then re-coding it, without change, directly into a new technology. A second approach is the “bottom up” method which generates very large discoveries of logic, most of which is system focused, with little business value. In either approach the possibility exists that outdated, incorrect, and poor business logic is moved from the legacy system to the new system. To understand how our method improves on current practice, it is important to first understand some basics about system transformation.

System Transformation Basics

System transformation involves discovering the current “as-designed” architecture and business logic, and reengineering it into the new, desired, architecture. The best way to understand the fundamentals and processes of system transformation is to review Figure 1, The SEI “System Transformation Horseshoe”.

Figure 1: The System Transformation Horseshoe



The System Transformation Horseshoe has three phases and starts on the lower left side and circles around to the lower right side. The first process recovers the “as-designed” architecture by extracting and analyzing artifacts from source code. The second process is the architectural transformation of the recovered “as-built” architecture to the new architecture. The third process completes the cycle and implements the desired architecture into the new system.

As can be seen from the figure, there are short cuts across the process. It is these short cuts that lost the potential value in the system transformation projects, generating systems that are “the same but different” – new systems that may be based on newer technology, but have little added value to the business.

To get the most out of a rules harvesting project, we focused on a Lean approach to completing a full architectural transformation, but at a cost that is no higher than if we had taken the short-cut across the Horseshoe. The approach includes:

1. Eliminating waste by focusing the project only on business logic that is valuable and relevant to the stakeholders.
2. A methodology that harvests business logic with the least amount of effort and cost.
3. Placing business logic (i.e. business rules) in a business context so they are meaningful and relevant. Without context, business logic is meaningless.

System Transformation and the Decision Model

The process of digging through program code in search of business logic will never be simple. However, the Decision Model simplifies the mined business logic representation for all audiences. The Decision Model is a deliverable that can achieve transformation from the low-level existing code to a high-level architecture. It can provide significant improvements of efficiency over other methods. The Decision Model fills a significant gap that is missing in architecture because:

- There is no standard way to code the business logic received from the business (in artifacts called requirements). Consequently, the logic in code is often subject to personal style and preference.
- Some of the most important logic of businesses today is buried, lost, and unknown.
- Business logic has a significant impact on business performance, especially in an age that demands enterprise agility.

The Decision Model organizes business logic into decision-based structures that are ideal for BPM and SOA, and are:

- Simple to create and interpret because it is visual,
- Universal because it is rigorous and repeatable, and
- Technology-independent because it is unbiased by software constraints
- The Decision Model may achieve for business logic what the Relational Model has for data. The new representation for business logic would become a model that:
 - Serves as a common communication between business and technology professionals
 - Functions as the first step for automation on various technology platforms
 - Enables faster changes and better impact analysis
 - Encourages better practices whereby non-technical people can create models of business logic
 - Fills a missing gap in today's technologies, methodologies, and business practices.

The Decision Model is a platform independent model of business rules that provides for the organization of business logic into a stable, repeatable, and normalized structure. It is the subject of a soon to be released book, "The Decision Model: A Framework for Business Logic and Business Driven SOA" (von Halle & Goldberg, Auerbach 2009).

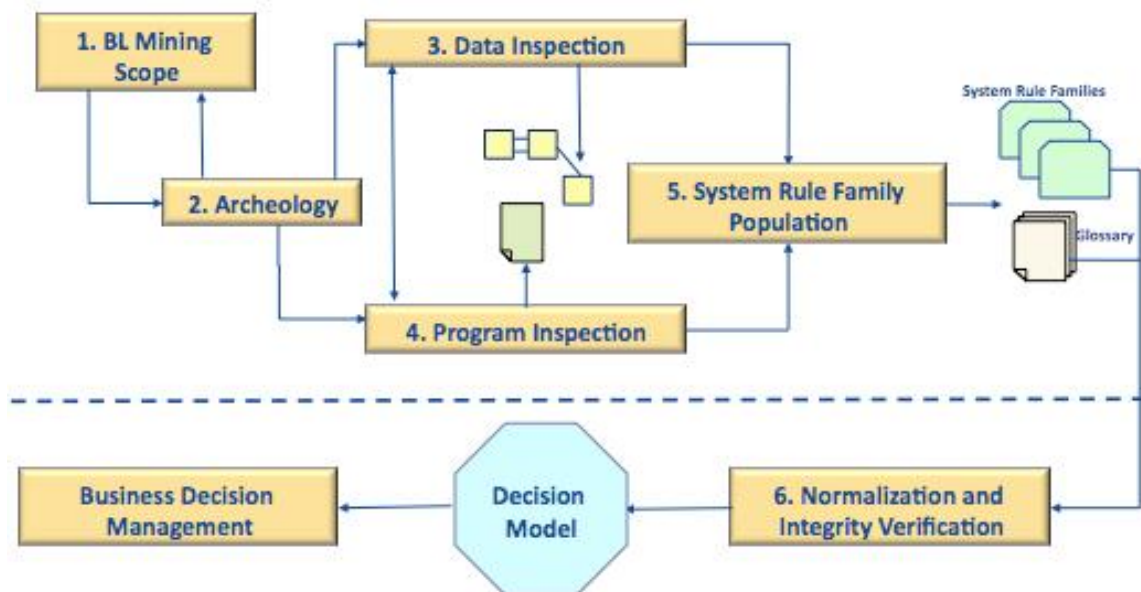
Harvesting the “As Designed” Architecture Approach

Transformation projects often experience failures during this phase. Failures frequently occur because automated software uses a bottom-up approach that usually results in an expensive project that harvests large volumes of rules with no or little context, as well as questionable value.

The best and most efficient method for harvesting the “as-designed” architecture is to take a “top-down” approach. A “top-down” approach starts with a review of business use cases, high level process models, and user interfaces, to identify decision points. These are the candidate business concepts for which objects will be identified in the code by an automated tool. The logic in that path is decomposed into their logic statements, and is the foundation for building the Decision Model.

Figure 2 depicts an approach, used with success, for mining business decisions from code. The diagram depicts the first four steps as a conventional business rule mining process, whether an automated tool is used or the process is done manually. Step 1 includes scoping or drawing the boundaries of the code to be mined. This is done by using the “As-is” model to identify the high value candidate decisions; these are the groups of business rules that will be of interest to the transformation team. This method eliminates a very significant portion of unnecessary mining.

Figure 2: High level depiction of Mining Business Logic using the Decision Model



Step 2 includes archeology wherein the inventory of all the artifacts is compiled so that the techniques and tools to mine the code and data can be determined. Step 3 includes slicing the code into segments that may contain business logic and inspecting them to

mark and recover specific code snippets of buried business logic. Step 4 includes inspecting the data structure definitions and sometimes database content in search of business logic. Step 5 is critical as it is where the Decision Model adds improvement over other techniques. Step 5 introduces the notion of System Rule Families – representing the business logic and fact types found (either by hand or by automated tool inspection) in the existing code.

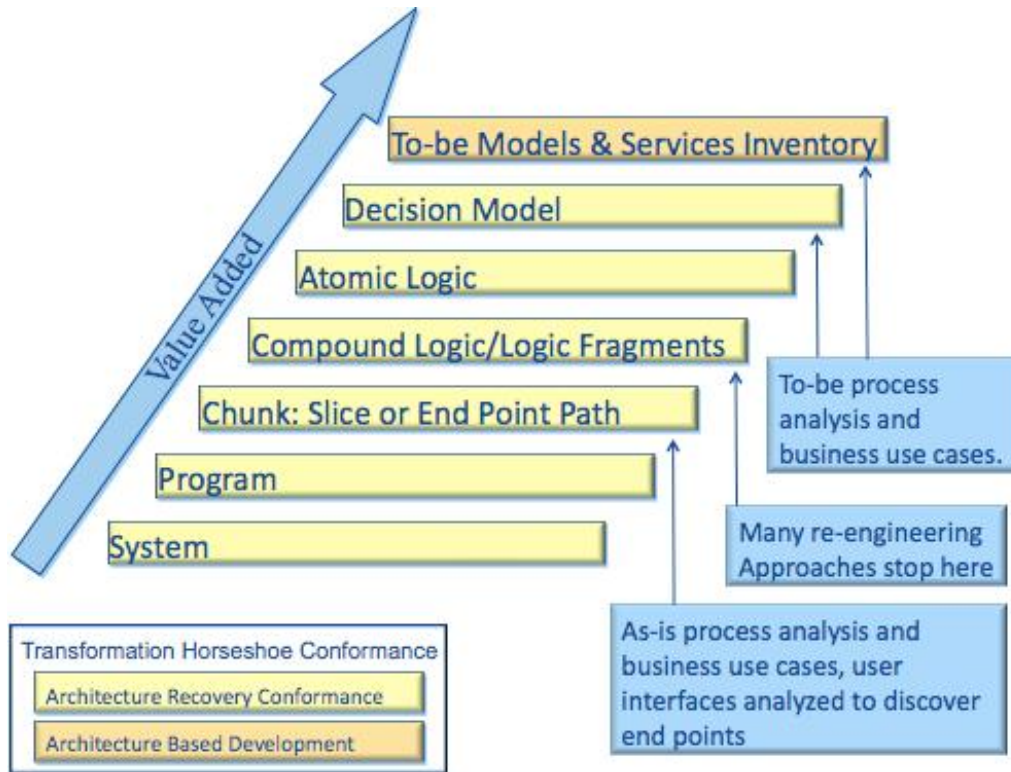
Step 5 involves inspecting the code snippets in search of condition or conclusion fact types, program variables, or database fields. These are assembled into Rule Family structures for conclusion fact types that have been discovered in the code. As the process continues, System Rule Families emerge. They are called System Rule Families because they represent the inherent structure of the business logic as found in the existing system, and using the variables and/or fact types used by the system. They are not in business-consumable form, nor are they normalized.

Once the System Rule Families are fully populated (that is, the code has been fully examined), step 6 is the creation of a business-friendly glossary of fact types. The business-friendly fact types replace the system-oriented fact types in the System Rule Families. In this way, true Decision Models begin to emerge.

Naturally, in step 6, the fifteen Decision Model principles are applied to the evolving Decision Model, to ensure that they are fully normalized, just as in any Decision Model project. When that analysis is complete, and the principles fully applied, the Decision Models are complete, and ready to be implemented into the target technology.

Once the Decision Models are available, then the “To-be” process models and use case can be developed, using the Decision Models. The Decision Models become candidate services for the inventory of services, either to be used as stand-alone decision services, or as a part of a larger composite service. This step completes the business transformation, constituting the highest added value for the enterprise. This is depicted in Figure 3.

Figure 3: Depiction of Value Add in Mining Code



Conclusion

The result of this decision-aware system transformation is that we have delivered fully on the full trip around the System Transformation Horseshoe, at least for the logic of the system. Our experience shows that, using the Decision Model, it is no more difficult or time-consuming to do this than if we had taken “shortcuts” across the Horseshoe, using traditional rule mining methods and technology. The Decision Model provides the means to achieve a high value architectural result from System Transformation, at a cost that is potentially no greater than that which would have been paid for a lower value deliverable.

This methodology has been successfully used in large transformations in a variety of industries, promising to transform the art of transformation.

The Authors:

Larry Goldberg, Managing Partner, KPI

Larry has over thirty years of experience in building technology based companies, focused on rule-based technologies and applications. Prior to joining KPI as Managing

Partner, he was Senior vice President of Sapiens Americas, Inc., which he joined when he sold his company, PowerFlex Software Systems, Inc to Sapiens. Larry is co-author (with Barbara von Halle) of “The Decision Model: A Framework for Business Logic and Business Driven SOA”(Auerbach, 2009), co-editor of “The Business Rules Revolution” (HappyAbout 2006), Editorial Director of the BPMInstitute.org publication “The BDM Bulletin”, and co-chair of the Brainstorm Business Rules Symposium track.

David Pedersen, Consultant, KPI

*David has over 25 years of experience in finance, technology and developing solutions for global infrastructure initiatives for industry and non-profit organizations. Prior to joining KPI, he was director at Ernst & Young, LLP where he developed and implemented complex Global infrastructure systems. He also developed solutions for the hazardous waste industry, legal profession, insurance and printing industries, and non-profit organizations. He is a CPA and a licensed pilot, who lives in Ohio.
(dpedersen@kpiusa.com)*