Goals, Domains, and Enterprise Architecture in the Model-Driven Organization

Desmond D'Souza

Kinetium, Inc.

desmond.dsouza@kinetium.com www.kinetium.com

In a Model-Driven Organization (MDO), all aspects of planning, designing, implementing, deploying, operating, and evolving the organization and its supporting infrastructure are based on models. Goals form an anchor for other models in an MDO. This paper is a summary of my keynote talk at AMINO 2013 in Miami, Florida, in which I covered a variety of ways to use Goal Models as a recurrent focal point to improve model coherence in an MDO.

Goals, Domains, and Refinement

A goal is a property desired over a domain. In Figure 1(a), a goal of the Golden Gate Bridge is shown with a green circle *G1: Get vehicles from A to B* i.e. given some *domain property* about the *inflow* of vehicles at *A*, establish a corresponding *outflow* at *B*. A goal is anchored to domain elements that are its subject matter, and evaluates to true or false over any domain instance. *G1* is anchored at end-points *A* and *B*, controlling *outflow* based on *inflow*. A dashboard wired to the anchor points can, in principle, monitor the goal.

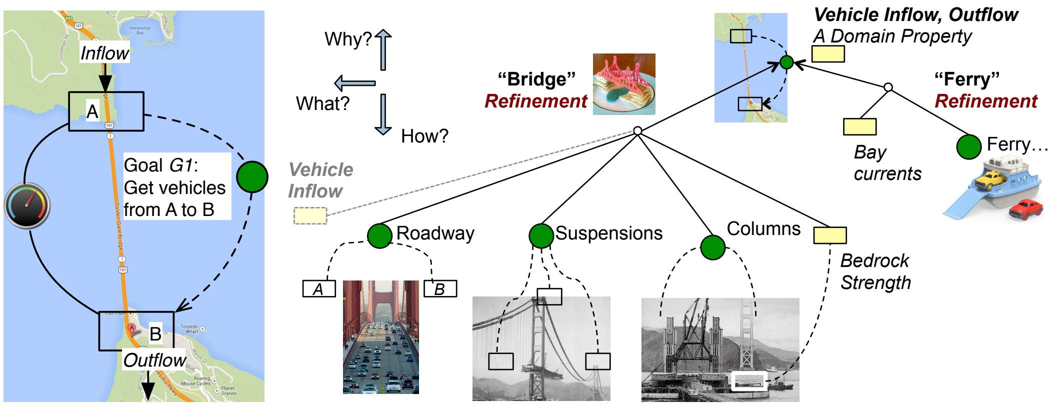


Figure 1: (a) A goal with its end-points and a monitor. (b) Goal and domain refinement.

A goal *refinement* consists of sub-goals and domain properties that, combined, satisfy a parent goal across its end-points. The *bridge refinement* (small white circle in Figure 1(b)) moves *inflow* from *A* to *B* via a roadway, supported by cables suspended from columns built on bedrock, and establishes sub-goals for each of these based on their mutual load and support properties, bedrock properties, and inflow. Domain properties (yellow rectangles) are facts or assumptions about the domain, also anchored on domain elements. A goal can have alternative refinements: *G1* can be met by a *bridge-refinement* or a *ferry-refinement*. Choosing one or the other uncovers different domain properties: *bedrock strength* is vital for bridge columns, and *bay currents* is for a *ferry*, while *inflow* and *outflow* is part of the problem context of *G1* itself and common to both. When analyzing any part of a goal model, only certain refinements apply, and the corresponding domain model is *composed* from the domain properties of all applicable refinements. Goal refinement, domains, and architecture choices are intrinsically intertwined.

The domain model can range from a simple average vehicles-per-hour, to a detailed "film-strip" with individual vehicles moving along the bridge. Goals are predicates over that model, and evaluate to true or false on a domain instance. An *objective* associated with a goal can quantify how well the goal is met, based on measurable domain attributes and often in an aggregate sense.

By making *intention* explicit, goals answer some key questions (Figure 1(b)):

1. *What* are we trying to accomplish? The goal specification, *G*, with end-point domain elements, gives a precise success criteria over any domain instance.
2. *Why* do we want to accomplish *G*? Refinement answers this in the form: because if we meet *G*, given additional domain property X and assuming we meet other sibling goals, then in concert we meet the higher level goal.
3. *How* will we accomplish *G*? Examine the refinement of *G* and follow the "line of reasoning" through its domain properties and sub-goals.
4. *How well* does refinement *R1* meet *G*? Provided all children of *R1* can be expressed in a sufficiently detailed form, evaluate *G*'s objective function against domain instances assuming that all the sub-goals are met.

Monitoring a goal is clearly useful, but *assumed* domain properties are also candidates for monitoring. As a secondary goal, the bridge should accommodate shipping traffic, which introduces assumptions about ocean level and ship height, and goals about minimum roadway height (Figure 2(a)). The assumptions can be monitored, as changes could jeopardize a goal. In the larger feedback loop of an extended enterprise with its environment, assumption tracking must happen in some form, whether proactive or reactive.

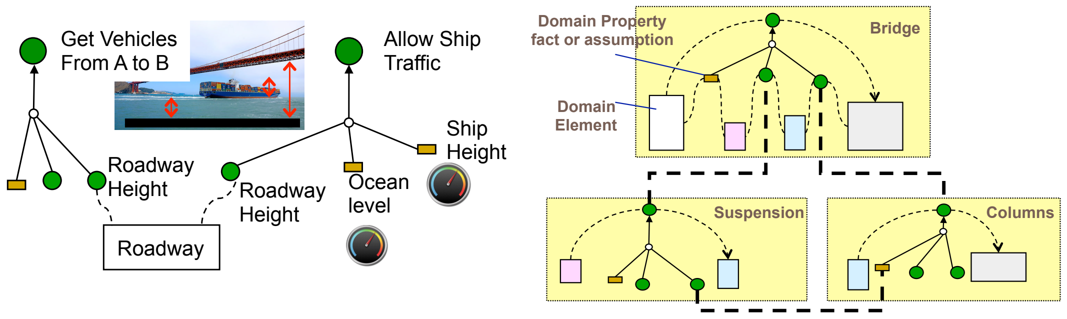


Figure 2: (a) Monitoring assumptions. (b) Federated goal models and alignment.

Goal models can be federated. Figure 2(b) shows federated goal models for the bridge, suspension, and columns. Note that each goal "frames" a problem; goals and end-points must align from child to parent goal model; one model's assumption can be (or otherwise depend on) another model's goal; and goal models have different projections of shared domains. Since dependencies are between properties of domain elements and not directly between elements, value, risk, and impact can be assessed in terms of affected properties.

If the underlying domain fits within a governance structure, goals can be extended with strategic dependency relations between the person desiring the goal, the one responsible for meeting the goal, and the one with authority over any domain element needed in refining the goal.

Goals, Architecture, and Architecture Style

An *architecture* of a system is a *model* describing system *properties* to be understood and analyzed *together*. Architectural components are part of the domain model. Goals and domain properties demarcated by refinement define *"together"*-ness and are part of architecture. The "ends" in "end-to-end architecture" are precisely framed by goals.

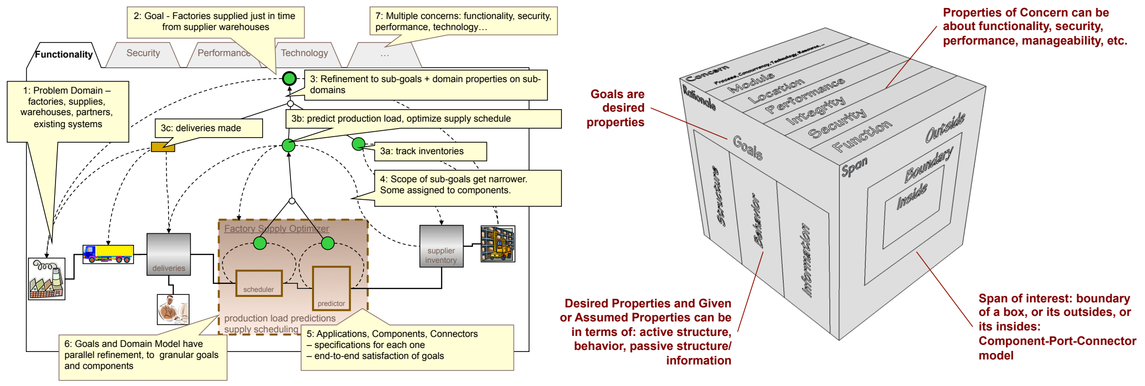

2: Goal - Factories supplied just in time from supplier warehouses
7: Multiple concerns: functionality, security, performance, technology…
Functionality
Security
Performance
Technology
…
1: Problem Domain – factories, supplies, warehouses, partners, existing systems
3: Refinement to sub-goals + domain properties on sub-domains
3c: deliveries made
3b: predict production load, optimize supply schedule
3a: track inventories
4: Scope of sub-goals get narrower. Some assigned to components.
Factory Supply Optimizer
deliveries
scheduler
predictor
supplier inventory
production load predictions
supply scheduling
6: Goals and Domain Model have parallel refinement, to granular goals and components
5: Applications, Components, Connectors – specifications for each one – end-to-end satisfaction of goals
Goals are desired properties
Concern
Process, Concurrency, Technology, Resource, ...
Rationale
Module
Location
Performance
Integrity
Security
Function
Goals
Span
Outside
Boundary
Inside
Structure
Behavior
Information
Properties of Concern can be about functionality, security, performance, manageability, etc.
Desired Properties and Given or Assumed Properties can be in terms of: active structure, behavior, passive structure/ information
Span of interest: boundary of a box, or its outsides, or its insides: Component-Port-Connector model

Figure 3(a) Example of fractal refinement. 3(b) Scheme for fractal goals with architecture.

Figure 3(a) sketches an example of the approach used in a fractal manner for factory supply optimization: the granularity of goals and domain elements, including software components, events simple or complex, and other behaviors, can all be refined recursively; 3(b) shows a general scheme for incorporating goals as a fractal structure alongside other architecture descriptions.

An architecture description can obscure, suggest, or reveal intention. In Figure 1(b), hanging cables longer than some length need to be reinforced as their own weight adds to the cable strain at the top. Below are three architecture descriptions of the cables, ranging from obscuring to revealing intention.

1. Cables 1-6 are normal, and cables 7-9 are reinforced.
2. For every cable: *(weight, strength, load)* must satisfy a *strength rule*.
3. Cables = map *cable_reinforcer* basic_cables

The first reveals no intent; it simply lists the final result of implicit reasoning. The 2nd reveals intent as a *checkable* rule, without helping shape the architecture. The 3rd provides an intention-revealing *transformation* that is generative.

An *architecture style* defines a set of architectures, and is described like any object type: attributes that name relevant architectural elements (*cables*), attributes of those elements (*weight, strength*), functions that determine attributes from others (*cable_reinforcer*), invariants (*strength rule*), desired and given domain properties. Like the *cables* example, architecture styles can span a spectrum from check-only (given an architecture, return Pass/Fail) to generative (given an architecture, evaluate to a transformed architecture). Since architectures include goals and domain properties, the most generative kind is an "architecture compiler": given goals and domain context, produce an architecture realization.

Goals and Roadmaps

Roadmapping is a decision-making technique used to support medium to long-term strategic planning, examining linkages between the domains of technology and business capabilities, organizational objectives, and the customer and market environment, considering multiple perspectives and their relationships over time. An MDO could analyze goals, facts, assumptions, and architecture elements in these domains on a timeline; key assumptions could bear monitoring over this timeline. Figure 4 illustrates what such a roadmap looks like.

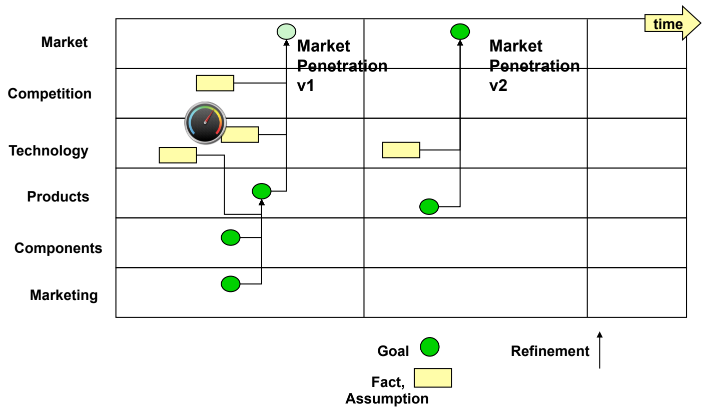
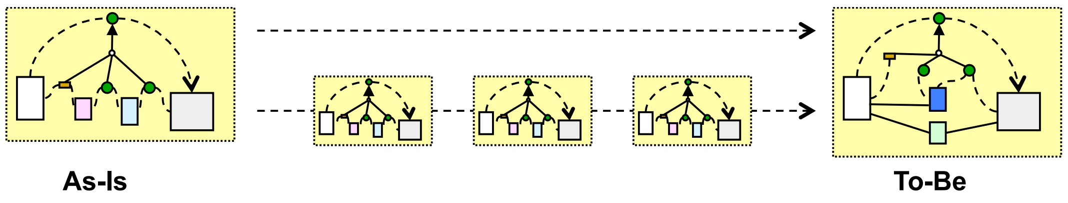


Figure 4. Example of roadmap based on goal modeling.

Goals and Migration Plans

Large-scale architecture initiatives deliver an *as-is* and a *to-be* architecture, and an MDO could use goal models in developing these. A *migration plan* is a sequence of staged architecture changes from *as-is* to *to-be*.



Migration planning is a difficult problem, with its own goals and domains. The obvious domain is a filmstrip of architectural stages, but there are peripheral components, applications, processes, people, skills, and regulations to consider; other roadmaps to co-ordinate, such as infrastructure upgrades. The obvious goal is to convert *as-is* to *to-be*; but there are others, such as minimizing disruption risk to critical business processes. Migration planning even has its own *architecture styles*, such as *Legacy Coexistence* and *High-Risk First / Fail Early*.

Acknowledgements

This paper is based on a modeling approach we have evolved over several years (called MAp, www.kinetium.com). It uses ideas from KAOS, Problem Frames, Catalysis, and Model Driven Engineering. I also owe thanks to past collaborators including Tony Clark and Hans Gyllstrom. The keynote slides also discussed goal-models and portfolio management, excluded here for lack of space.