Module 3 Homework

Steve Mazza

April 27, 2012

i) number one

ii) number two

iii) number three

I will be describing the capabilities and limitations of the Dori Object-Process Methodology (OPM) and describing some of the situations where you may find it more or less effective. Lastly, I will address its fitness to my present job and area of responsibility.

Dori builds Object-Process Diagrams (OPDs) from Object-Process Language (OPL) which are a human-readable constructs. Furthermore, OPM categorizes everything into nouns, verbs, and noun-verb states which they refer to as *objects*, *processes*, and *states* respectively. Objects, processes, and states are connected by graphical primitives identifying *specialization*, *aggregation*, *consumption*, and *results*.

Due largely to the flexibility and orthogonality of its constructs, Dori has been shown to be equally adept at modeling both artificial and natural systems. Furthermore, its descriptive language is expressive at most levels of engineering detail and consequently supports *in-zooming/out-zooming*, allowing the model to selectively expose or hide detail as appropriate.

This *zooming* is illustrated in a series of graphics in the class reading.[1] The illustrations begin with the highest level diagram of the OPM model and zoom down into appropriate portions, exposing successive levels of detail that support the activities (stages) *System Developing*, *Requirements Specifying*, *Analyzing & Designing*, and *Implementing*.

The use of natural language in the construction of the OPL sentences means that even users unfamiliar with OPD can validate specifications. OPL is an automatically generated result of OPD and so there is no need to maintain parallel resources; the system does this

[1] Jeff A. Estefan, "Survey of Model-Based Systems Engineering (MBSE) Methodologies," Jet Propulsion Laboratory (May 2008): 46-49.

for you. OPM is supported through the commercially available tool, OPCAT, which has a restricted academic version that is suited for evaluation.

Dori has the conspicuous limitation that it does not easily and naturally integrate with industry standard notation such as UML/SysML. This may be a significant barrier to adoption by an organization with an existing investment in infrastructure and process supported by these standards.

Dori would be a highly effective tool supporting ground-up architecture and design of complex systems, providing a mechanism that facilitates capture and communication of requirements and specifications that can be viewed and presented at various appropriate levels of detail. Dori would be less effective if it were required to integrate into an existing system that relied on the industry standard UML/SysML specification standards.

Most of the focus of my current work is in support of software engineering efforts aimed at producing warfighter capabilities on commercial smart phones and tablets where there is a small amount of hardware integration. Given the current stated functionality, I believe that Dori would be well suited for use in our work environment and would nicely support our efforts both now and as our need for tighter hardware integration increases.

Simple, orthogonal diagramming constructs that produce natural language descriptions, capture and hide complexity, support the architecture and design process, and are available in a commercial tool all conspire to make Dori an attractive tool set. The ability to ingest and output UML/SysMML would facilitate integration into existing processes and design systems and significantly reduce the barrier to acceptance of this capability suite.