Redefining Static Analysis, A Standards Approach

By: Rama S. Moorthy, CEO, Hatha Systems
Ioan (Mike) Oara, CTO, Hatha Systems

Introduction: What is Static Analysis?

There are two ways to gather information about a system and analyze it for security or other purposes: one is to look at it as it operates and the other is to look at its artifacts. These two methods correspond to dynamic and static analysis. Although the dynamic approach has often been the easier path to take for analysis, it is the static approach that can render more comprehensive results.

Take for example the case in which a security analyst tries to determine if there is a particular user interface in which certain confidential information (such as a customer personal address) is displayed. With dynamic analysis, one would have to execute all possible operations of the system, enter any possible combination of codes, and even try to supply values that do not make sense. This task may be overwhelming and may never provide a 100% assurance that the confidential information would never surface. However, through static analysis, which involves looking either at the source code or at information extracted from it, the analyst can discover with absolute certainty if the customer personal address is displayed somewhere. Moreover, if it is displayed, the analyst may also find the precise combination of input data and user actions in which this is happening.

While static analysis could be as simple as looking at the source code of an application, the last decade saw the emergence of specialized tools which deliver both high productivity and precision. The amount and complexity of data make such tools indispensable. To perform static analysis, such tools usually go through two phases:

(a) Data gathering through the parsing of the source artifacts
(b) Specialized analysis that digest the information and present it in a useful form.

The specialized analysis could render diagrams which help the analyst get a view of the application at any level of detail. The following are examples of those views:

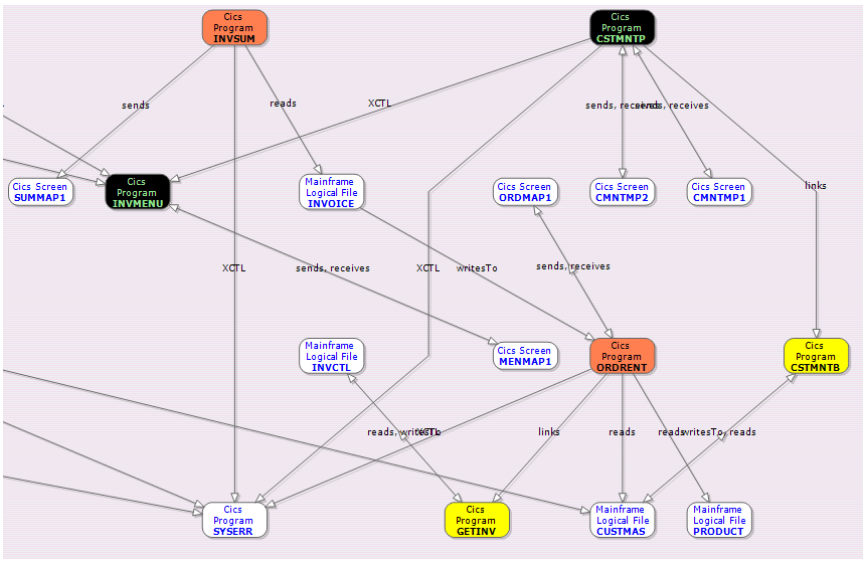


Figure 1: System Architecture

A system architecture diagram indicates how various platform components interact. In the Figure 1 diagram, one can see how particular programs interact with the screens and read or write data to files.

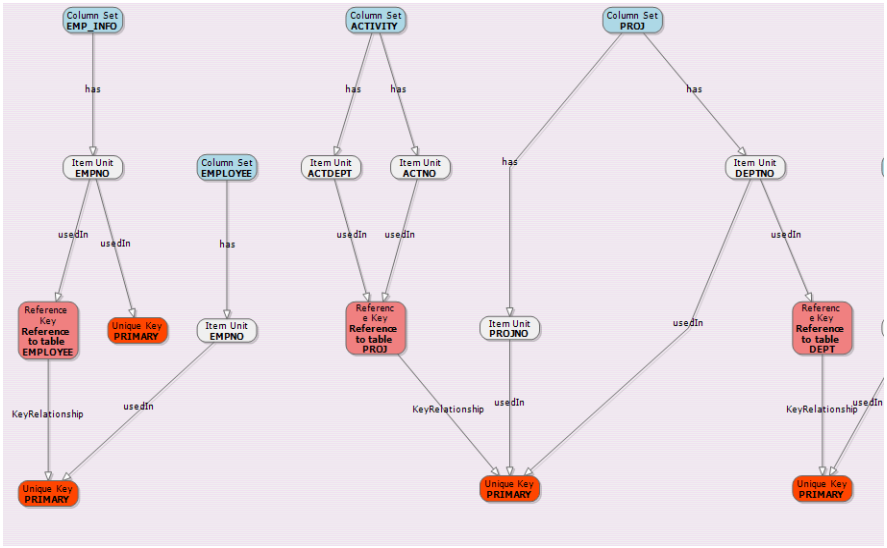


Figure 2: Data Architecture

Some Static Analysis tools are capable of automatically extracting database schemas and discovering complex data relationships. Figure 2 above is an example of that extraction.

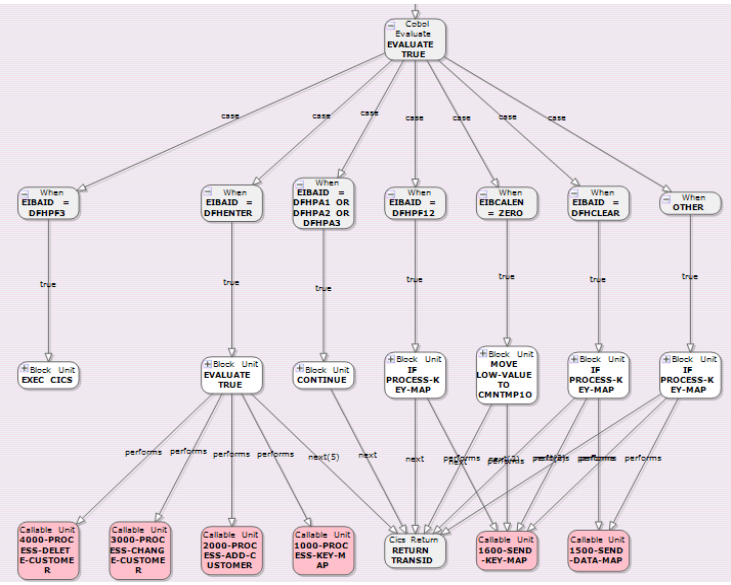


Figure 3: Control Flow diagram

A Control Flow diagram, as in Figure 3 above, helps discover the various paths through the code and the conditions on which they branch. This in turn helps discover business rules, data validations or process composition.

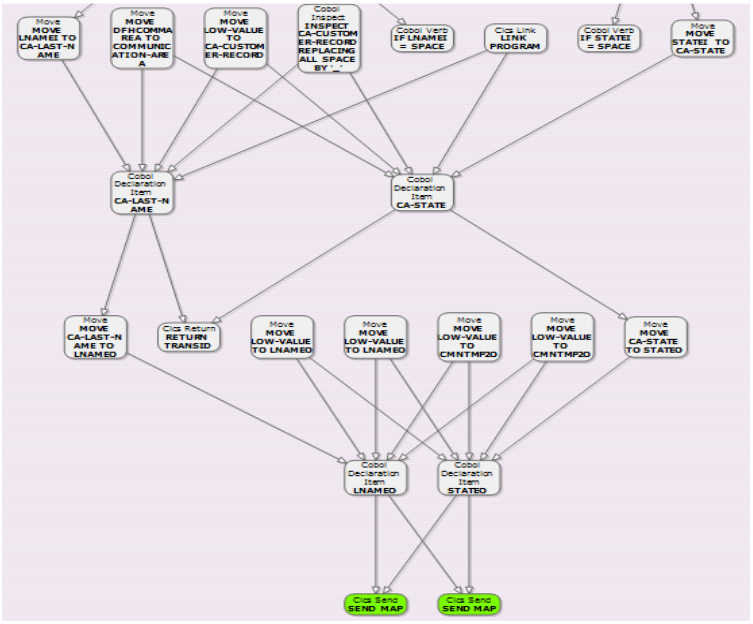


Figure 4: Data Flow diagram

A Data Flow diagram, as in Figure 4 above, helps discover the paths of data through the application. In particular, one may discover what data is presented to the operator and what the data origin is.

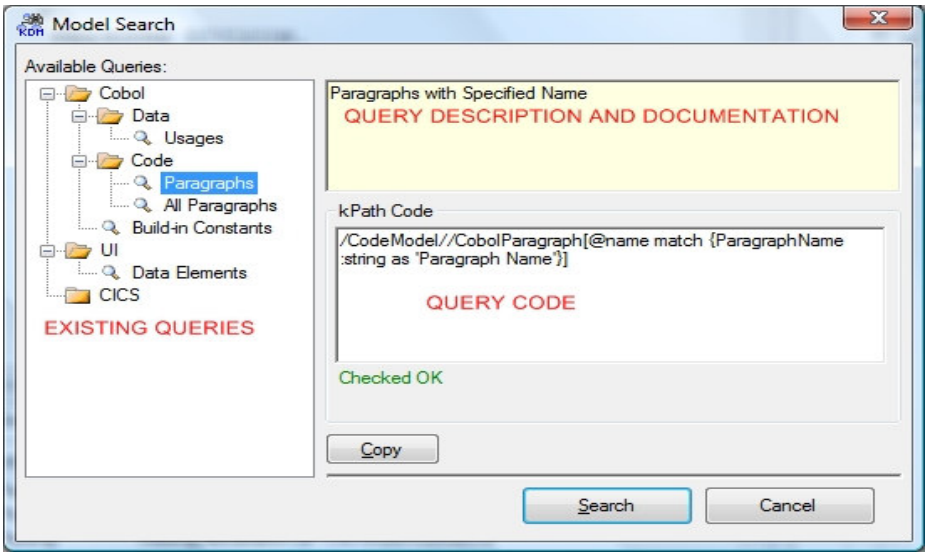


Figure 5: Static Analysis queries

A Static Analysis tool may also provide the analyst with query capabilities. Figure 5 is an example of how such queries are defined. Various tasks, such as business rules extraction or code weakness discovery

Engineering Applications into Securely Composed Systems

One particularly strength of static analysis tools is the capability to analyze the strengths and weaknesses of a composed system. Modern paradigms require that applications work in concert as opposed to separate stacks. As many applications were developed in the past without a requirement for connectivity, bringing them together requires extensive analysis, which is helped by static analysis tools. They can help with a number of distinctive tasks.

1. Discover if the data provided by one application is consistent with the data required by another. This involves data formats, data validation rules and semantics of data.
2. Discover if the security rules of one application are consistent with the security rules of another. For instance, one may require a special authentication, which is not required by another.
3. Discover if different components are at similar levels of assurance, such that one will not degrade the other.

One simple example refers to the input validation rules. One component application may accept a customer regardless of age, while a second may impose some age restrictions. If the first one lets a customer pass and makes a request to the second, this in turn may end up processing unqualified customers. A static analysis tools may be able to collect all validations from both applications and compare them for consistency.

Issues with Static Analysis Tools

There are a number of issues that have continued to impact the advancement of comprehensive tool aided analysis. The speed of innovation in the development of new languages and technologies, although providing tremendous efficiency and ease to the development community, has forced the discipline of tool aided static analysis to focus on only a narrow set of issues (e.g., extraction of code weaknesses, business rules extraction). Such solutions address only isolated issues for a few established languages, and fail to offer a comprehensive approach capable of simultaneously viewing code, architecture, platform, data, business/operational rules and business/operational processes. All of these characteristics of a system play a part in fully understanding the state of a given system. In other words, full contextual knowledge of any system being analyzed is critical to providing a comprehensive view of its strengths and weaknesses. Additionally, the state of constant change in software also dictates the need for this knowledge to be extracted on-demand.

Given the heterogeneity of languages and the continuous state of change in technology, the most optimal way to address the knowledge extraction and automated static analysis is to use standard models. Standard models allow for the use of a common language (ontology) and can be applied to build out an eco-system of automated tools, regardless of the system being analyzed or the type of analysis being performed. Once a standard representation is used, tools which are otherwise specialized in either particular technologies or in particular types of analysis can come together and complement each other. Even for specialized tasks, such as the discovery of code weaknesses, it was determined that different tools deliver slightly different results. Using them in combination would assure a more comprehensive and higher quality analysis of a particular system.

One particular area in which the standards may prove decisive is related to the issue of composable systems discussed previously. If one tries to integrate two systems built on separate technologies, it is highly probable that while analysis tools may be available for both technologies separately, no tool has to date been capable of dealing with both of them. However, if the two useful tools are built on the same standard model, their data, results and conclusions may be integrated.

Standards Progress

Over the last seven to eight years, a group of industry leaders have been addressing the need for international standards that knit together to provide a comprehensive analysis framework. This effort has been driven by the modernization community which requires full system knowledge of the 'as is' system, a critical component for both analysis and reuse when migrating the system. This community is entrenched in system engineering methodologies and process, and has brought that same rigor to addressing software analysis as an engineering

effort. The result is a number of standards that set the stage for comprehensive static analysis. The standards include:

Knowledge Discovery Metamodel (KDM): an ISO/OMG Standard providing ontology (a set of definitions) for system knowledge extraction and analysis. KDM provides a framework for the capture of code, platform and other software system characteristics. This further allows the extraction of data flows, control flows, architectures, business/operational rules, business/operational terms, and the derivation of business/operational process; the extraction can be delivered from source, binary, or byte code. Additionally the intermediate representation of the extraction is in executable models creating the possibility of simulation and code generation.

Business Process Modeling Notation (BPMN): an OMG standard delivering a modeling notation used to capture business/operational processes in support of system and organizational process simulation and analysis. It is used today to capture both human and IT system processes for the purposes of simulating environments both 'as is' and 'to be' for software modernization. This notation is compatible with KDM so that system extraction can be represented in BPMN for gap analysis of the current state of the system vs. what is thought to be the current state of the system – critical for modernization and compliance.

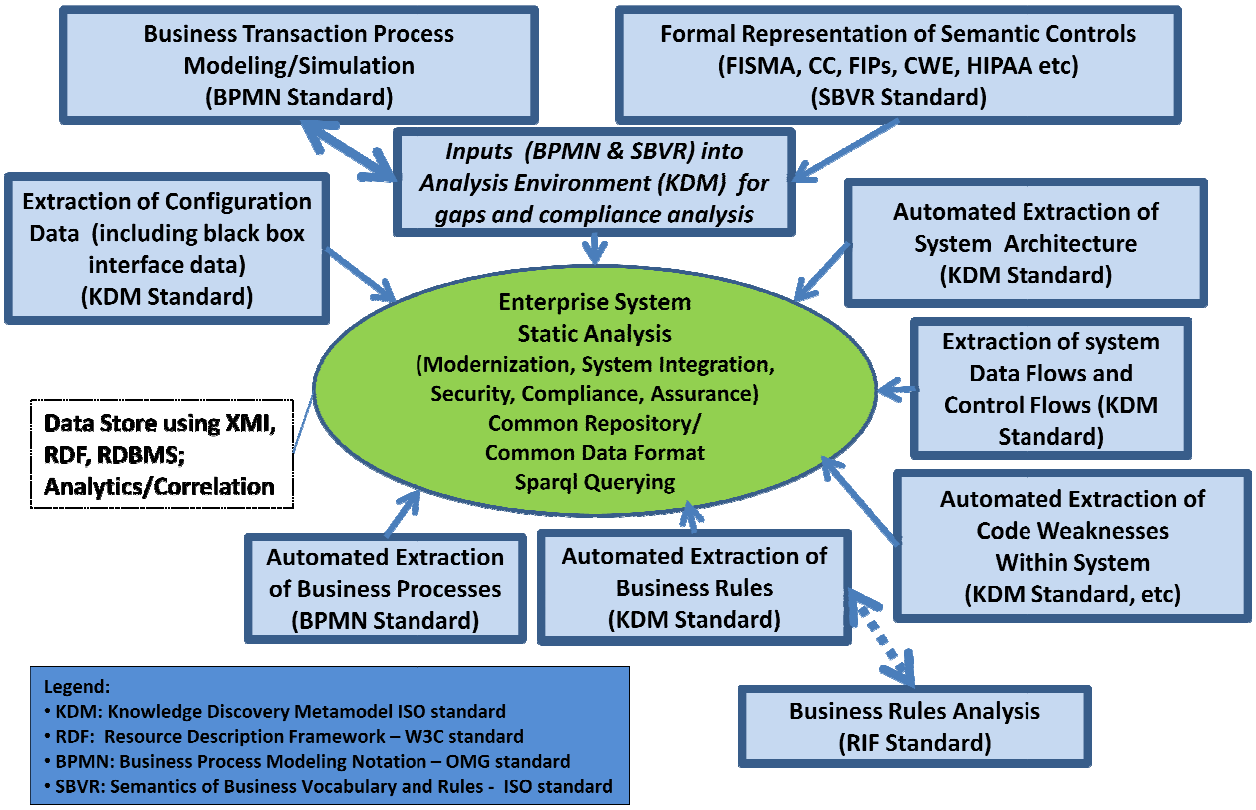
Rules Interchange Format (RIF): W3C standard, this standard delivers representation used for specifying, analyzing and exchanging information about business rules. Once captured in this format, business rules may be also be used in simulation, gap analysis and compliance analysis. The analysis of business rules is also an important aspect of application modernization.

SBVR (Semantics of Business Vocabulary and Business Rules): An ISO/OMG standard, this specification provides a structured process for formalizing, in natural language, the existing English language representation of compliance points. The standard enables the various compliance specifications (e.g. FISMA, HIPAA, SOX, FIPs, CWEs, etc) to be formalized reducing the room for interpretation from organization to organization when implementing the compliance and auditing requirements.

Data/Metadata Storage Standards (old and new): With the emergence of the standards noted above and the need for storing this information for analysis, a set of storage standards needed to be embraced. XMI, RDMBS, and RDF (Resource Description Framework) are the three formats that are compatible with these standards. RDF - perhaps the least known of them - is a W3C standard that is compatible with KDM and BPMN. There is a specific approach in the standard called RDF triple store which is currently being used in semantic web applications. The value of RDF is that it can manage large amounts of data and metadata which is critical for doing comprehensive static analysis.

Knitting of the Standards for a Comprehensive Static Analysis Approach

The diagram below provides a pictorial representation of the system information that is extractable using the various standards and how the standards knit together to deliver the foundation for software system knowledge extraction and comprehensive static analysis.



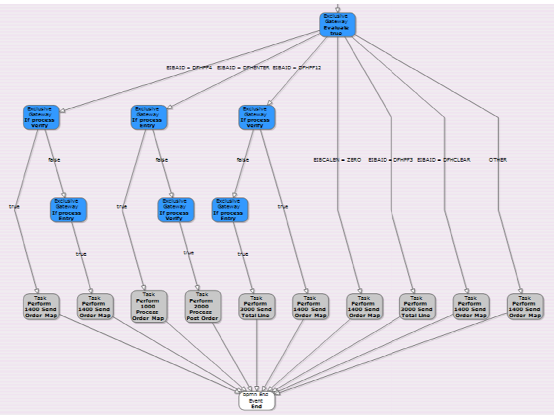
Possibilities when automated static analysis tools embrace these standards:

1) Business/operational logic can be extracted to derive business/operational processes into a BPMN format for documentation, re-architecture (including SOA and Cloud enablement), gap analysis and migration purposes.
2) Rules can be extracted and correlated with business/operational terms and processes for 'as is' system analysis.
3) Rules extracted in RIF format could be used to generate code or may be migrated to a business rule engine.
4) System architectures, data flows and control flows associated with them can be extracted and represented visually. These representations may be used to document the 'as is' system for the purpose of modernization, compliance or security analysis.

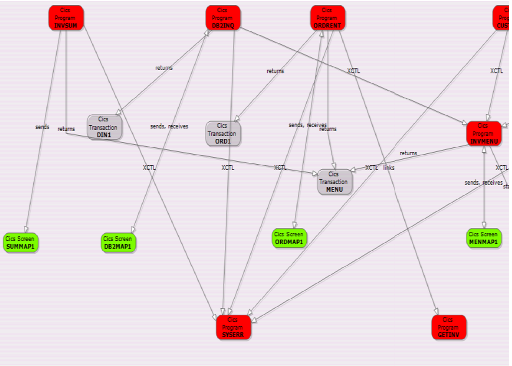
5) Code weaknesses can be discovered and then associated with the data and control flows in which they occur in order to determine their possible impact for security, safety, etc. within the context of a specific system. While various technologies aid in the discovery of weaknesses, standards-based static analysis approaches can help place them within a process or architecture context. This in turn can help to better estimate their impact.
6) Compliance points or controls can be represented in SBVR to formalize each control, then expressed in queries placed in a reusable repository for extracting compliance knowledge of the system. This effort can aid in creating a set of institutionalized compliance queries to be used repeatedly in the system lifecycle of a given organization.
7) Configuration data can be extracted in a KDM repository for those components that do not have source code, making them part of the overall system analysis for security or modernization purposes.

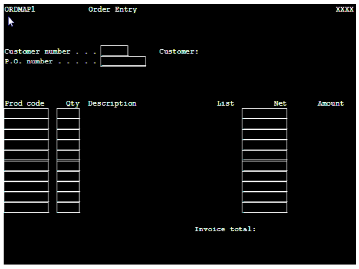
The screen captures below show views from a standards compliance static analysis tools environment:

Business/Operational Process Derivations:

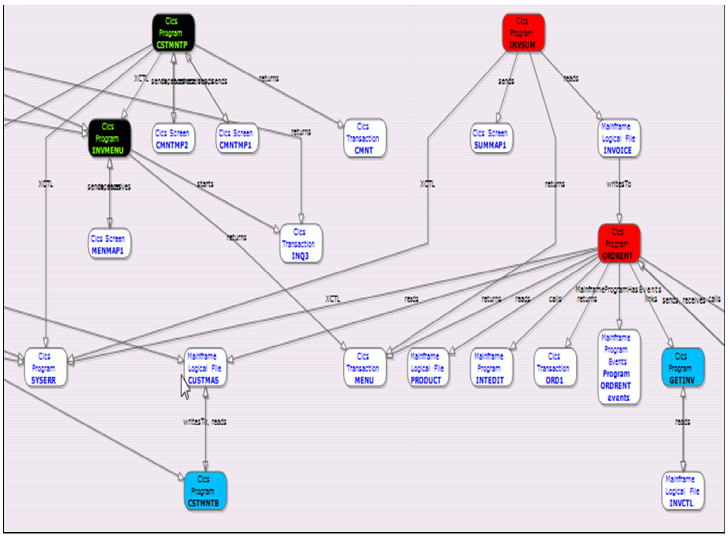


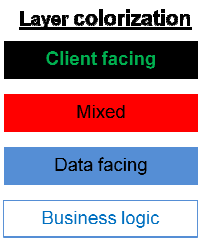
Screen Flow and Screen View Extraction:





Software System Layer Identification: Valuable for function or architectural extraction for security, safety, or SOA enablement or other applications requiring an analysis of functional layers within a system:





Conclusion:

Both large and small players in the software world have been adopting the various standards discussed above. There is an eco-system forming that brings together a set of tools that can provide knowledge extraction and analysis which can be integrated using a common data format for better correlation and comprehensive static system analysis. Hatha Systems with its Knowledge Refinery™ static analysis platform is the first of its kind to deliver a fully standards-based integrated solution. It provides full system transparency and comprehensive, traceable system analysis whether addressing software modernization or security analysis, delivering impact analysis and in turn risk management.

About the Authors:

Rama S. Moorthy is the Chief Executive Officer and Founder of Hatha Systems, a company providing automated analysis solutions for software systems modernization, compliance and security engineering, built on a foundation of international standards. She has over 22+ years of experience in creating new products, markets, and standards as well as running large revenue generating organizations at Sun, AMD, and Siemens. She has been a content contributor to the Department of Defense Software Assurance and Supply Chain efforts. She is a principal author of the NDIA – Engineering for System Assurance Guide, the DoD Supply Chain Risk Management (SCRM) Key Practices Guide as well as the SCRM NIST-IR. She has a BSEE from Purdue University and an MBA from Vanderbilt University.

Ioan (Mike) Oara, the Chief Technology Officer at Hatha Systems, is one of the veterans of the discipline of Application Analysis and Modernization. Mike started his career in early 80's working in the financial industry in New York. Over the last 20+ years, Mike has been developing technologies and holds several patents in the area of application analysis and modernization. He pioneered new solutions for business rule extraction, creating robust and practical extraction methodologies. At Hatha Systems he continued his innovations through the development of the next-generation of analysis and modernization solutions. Mike has published many articles focused on modernization and static analysis. Mike has a MS in Mathematics from the University of Bucharest.