# State of Art on Software Comprehension.

This Short document describes some articles talking about tools and strategies that help the software understanding. The articles are described beginning by the most relevant and generic, and ending with the most specific.

**Documenting Software Systems with Views VI:Lessons Learned from 15 Years of Research & Practice.**

**Author:** Scott Tilley, Department of Computer Sciences, Florida Institute of Technology.

**Motivation:** How to get a better approach to a system structure using accurate graphical or textual information for a certain stakeholder.

**Solution:**

Fifteen years of research over several workshops and tools used different ways to represent the structure of specific software. Each strategy is slightly different, but provides information about the same artifact. Each way of representing the information (textual or graphical) is called a view. The article specifies about legacy systems views, web application views and design patterns view.

Legacy systems views can be created by searching for building blocks thoughtthe software and making reverse engineering of them, generating subsystem aggrupation, which creates the legacy subsystems view. Also legacy systems needs useful documentation that have to be high quality, integrable and with single perspective. This is achieved using a proprietary tool, generating and reorganizing the documentation output on xml files stored in a common repository.

The web application views describes the structure of a web application in three views. The execution model which explains the navigation between pages, the transactional view that specifies the required web operations for a certain functionality and the navigation model explaining the grouping of interfaces by functionality.

Finally, the application design patterns view shows several ways to organize software elements into a user defined, yet generic or proprietary design pattern for better comprehension. The visualization should be well distributed because every concern has a particular stakeholder. This view is important to detect critical points of an application for maintenance.

**JOURNAL OF SOFTWARE MAINTENANCE: RESEARCH AND PRACTICE**

**Maintaining a legacy: Towards support at the architectural level..**

**Authors:**

Reinder J. Bril, Loe M. G. Feijs, Andr ́eGlas, Ren ́e L. Krikhaar and M. (Thijs) R. M. Winter.

**Motivation:** Improve the maintainability and quality of the PBX software developed by Philips Business communications.

**Solution:**

This paper describes the evolution of software development environment in Philips company, mainly triggered by the demanding consumer services with new technologies. The need to improve those process motivates the implementation of URSA. URSA tool is used for gather information of programs from the Philips company and extract different points of view from the software analyzed. These views are created by using processes of extraction, abstraction and presentation. Information extraction is from three sources: system history, system experts and the software itself. Abstracting information means reorganizing it on a way to present it on a higher design level . Finally the visualization (presentation) , is where the information is shown in a user friendly visualization. They describe the system architecture into three types of visualizations: The logical view, describing the services provides, a process view, describing the systems concurrency and synchronization aspects, a development view, describing the system's static organization (classes, modules, etc), a physical view, representing the mappings between software and hardware, and the scenarios, that shows all the four views together. Each visualization provides low level details when zoomed in.

**Automated Reverse Engineering of UML Sequence Diagrams**

**for Dynamic Web Applications.**

**Authors:** Manar H. Alalfi , James R. Cordy, Thomas R. Dean.

**Motivation**: Recover accurate information about dynamic behavior and interaction behavior over PHP applications, identifying interaction elements, loops and conditions, and similar execution trace patterns on static and runtime, obtaining a visualization showing these behavioral changes.

**Solution:** The creation of PHP2XMI tool provides MDE approached sequence diagrams based on PHP dynamic and static applications. This sequence diagrams shows the execution traces of a particular browser session reconstructing the whole activity . The way this tool gets this result is by three steps: Parsing and dynamic instrumentation, which inserts proves into the source code to collect dynamic information such as page URLs, http parameters, sessions and cookies. Filter and Storing , which gathers all the information of the previous steep and stores it into a SQL database. This information is filtered in order to avoid redundant data. Finally, the Database analysis and model generation gathers the execution traces stored an builds an UML 2.1 sequence diagram based on its meta-model.

# Recovering High-Level Views of Object-Oriented Applications from Static and Dynamic Information.

**Authors:** Richner, Tamar, Ducasse, Stephane

**Motivation:** How to use static and dynamic information to generate high level views of Object oriented Applications applying an iterative process driven by the devoloper (specifying the views declaratively). All of this over Smalltalk applications (a programming languaje).

**Solution:** This article classifies the information gathered from code into static and dynamic information. This information is stored in a model which classifies its elements into Basic relations (can be static or dynamic) and Derived relations, which are created by querying the basic relations. All of these between classes. After the model, a high level view can be created by selecting a cluster of elements and querying again over the model to select which type of relations you want to show. They generate views such as invocation between clusters, creation invocations and method groupings. This method uses a MDE approach powered by Moose (providing the Smalltalk Metamodel and Wrappers (info finders)) producing a FAMIX model, then, the information is processed by the GAUDI engine written in SICTUS Prolog (consisting of a set of rules and queries over the information of the model). The last one generates all the relations, clusters and functions used to create views. Finally de dot tool is used to display the views generated by GAUDI (Graphical engine). All of this was tested using the HotDraw painting application written in SmallTalk.