Abstract:

A major part of software systems are not designed for performance. Very often there are flows in the system that appear late in the life cycle of the system. Even if there are profiling and monitoring tools that can detect various problems, they don't provide a solution. It takes a very experienced designer or developer to deal with such problems efficiently. Solutions for a good design are the use of design patterns, refactorings and antipatterns.

Our framework provides a way to automatically correct design problems preserving the functionality of the system and improving its performance. The approach is to provide a system of patterns, antipatterns and refactorings that will provide the instruments for automation of design optimisation. The idea is similar to the work that a compiler does when it optimise it's output. The tool will use these instruments mainly to change the design of a model. Another task is to comprehend it. It will use patterns to understand how a system is modelled. It will divide the system into subsystems and components making it easier to process. It will gather information about participants to existing patterns.

An important feature of the process is that it will take a non-intrusive approach when redesigning. The functionality of the system will be maintained. Each pattern that solves a problem will be defined as a sequence of refactorings and/or it will have aspect-oriented strategies.

Other aspects are reuse of the experience and interoperability.

Finding similarities between different models, in order to better recognize patterns that are implemented into the new model, manifests the reuse of experience. Additional problems can be detected by finding similar scenarios between two models. Also, since the definition of the context a pattern address isn't usually comprehensive or complete, it can learn new situations in which a pattern may be used.

The interoperability is addressed by the use of UML and MOF standards for representing knowledge.

In conclusion the main targets are the system of patterns and the expert system that has the ability to compare models, find similarities and localise the context of a pattern or a performance problem.

As stated by the POSA books system of patterns should comprise a sufficient base of patterns, it should describe all its patterns uniformly, it should expose the various relations between patterns, it should organise its constituent patterns, it should support the construction of software systems and it should support its own evolution.

References: http://www.eeng.dcu.ie/~gabor/unofficial/gabor-it&t.pdf