DYANA-2012 provides the means for checking DRE model specifications expressed in terms of temporal logic formulae. In order to avoid full implementation of model-checking techniques, DYANA-2012 uses a well-known DRE model-checking tool UPPAAL as a target verifier.

UPPAAL is capable of checking temporal properties against input models expressed as networks of timed automata (NTA) [**?**]. NTA is a very expressive, and at the same time very simple DRE model which can effectively handle real time in model description and specification.

On the other hand, DYANA-2012 uses UML statechart diagrams for DRE model descriptions. The advantages of UML statecharts against NTA in respect to model design are obvious: UML statecharts provide system component hierarchy, and a wide choice of label expressions and ther interpretaions. However, these features make UML statechart diagrams unappropriate for a direct verification. To connect DRE description and verification, we fix semantics for UML statecharts and provide a translation algorithm which transform fixed statechart model to NTA model.

To describe a formal operational semantics for UML statecharts, we use the hierarchical timed automata (HTA) model first introduced in [**?**]. According to HTA model, each nontrivial component of a system is described as a set of consequent and concurrent subcomponents. Thus, each component of the HTA is either an envelope for its concurrent subcomponents with a standard interleaving semantics, or an automaton. In the latter case nontrivial system components can also be states.

HTA allows the usage of bounded integer variables and real-time clocks. Expressions over variables have syntax and semantics similar to those in C language, while expressions over real-time clocks are restricted by two types: a comparison of a clock against a natural number, and a comparison of a difference between to clocks against a natural number. Expressions are used as labels for states and transitions. State labels express necessary conditions to be satisfied while the state is active. Transition labels express necessary conditions for the transition to be performed. Variables and clocks can be changed when a transition is performed, but the clock change is restricted to the assignment to zero.

System components can synchronize not only via variables, but also via handshake and broadcast signals. Thus, transitions can be marked with special actions to send or receive a signal via specified channels. Channels operate in standard handshake and broadcast semantics: when a signal is sent, one transition (for handshake) or all transitions (for broadcast) which can receive it are performed immediately. Strict syntax and semantics of HTA (except of broadcast synchronization) are given in [**?**].

The translation algorithm is based on the algorithm, presented in [**?**]. It receives UML statechart written in HTA syntax as an input and generates an equivalent NTA which can be verified with UPPAAL. An NTA can be considered as a "flat" HTA, i.e. an HTA which has one concurrent envelope containing a set of consequent labelled automata with no hierarchy. Strict syntax and semantics of NTA are given in [**?**].

To flatten an HTA to an NTA, the algorithm produces a consequent timed automaton for each nontrivial component by replacing its nontrivial subcomponents by usual automaton states and providing auxiliary states and transitions to synchronize activation and deactivation of nested components. According to the translation, one execution step of HTA is replaced by a sequence of NTA execution steps, and activity of HTA components is simulated by activity of specific NTA automata states. Finally, a specification for HTA is translated into a specification for NTA according to the correspondence between components of HTA and states of timed automata.