calculating the distance or similarity to other cases in the database.

For the monopoly implementation we need to consider several basic strategies. Monopoly is based on investing in properties and receiving revenues from those investments. One of the basic strategies of the game is to build a set of properties that will bring constant income larger than the one of the opponents. So in time the opponents will have to declare bankruptcy. But on the other hand over investment can lead to too stretched resources with low income that will eventually drove the player to bankruptcy. To decide on these two we need a clear separation into two groups of cases in the CBR database. The first group of cases will represent a situation on the board where the player has significant income per loop formed of one or more color group properties, maybe railroads, some buildings on them and so on. It is important to note that in this case the player is better situated than his opponents so he only needs to survive long enough to win the game. In the other group of cases either the opponent is not well positioned on the board or its opponents are better situated. In this case further investments are necessary to improve the situation so the player can have a chance of winning in the long run.

These metrics can be owning color groups, valuing groups of railroads, evaluating the other opponents as well, and considering the amount of cash. As it is obvious in monopoly the number of streets is not as nearly as important as the combination of streets the player owns. It is also important to note that one CBR case does not hold only a single strategy in place, but its solution can have multiple different strategic goals. For example one CBR case might simultaneously say buy this land to form a color group but also trade some other unimportant property to increase cash amount.

The cases do not represent all possible combinations of board positions. They are only representation of typical game scenarios. The CBR Case solutions do not give exact instructions in general but rather strategic goals. For example one CBR Solution might say trade the streets that you only have one of each for the ones that you have two of that color already. Then the planner based on the situation on the board needs to decompose this high level task to a low level operations. Like offer "Mediterranean Avenue" for "Reading Railroad" and offer $50. The exact amounts and actual streets are left to the planer to evaluate.

The monopoly CBR database is currently in development on a monopoly clone game called Spaceopoly. The cases are architected based on human player experience and knowledge. There is a plan of making a number of slightly different strategies that differ on the style of playing and then running simulation tests that would determine the particular validity of each database as well as validity of certain segments of the strategy or even particular cases in the database.

The actual execution of the strategies will not differ from strategy to strategy since the plan execution is more related to the structure and rules of the game than to the actual playing strategy.

6.3 Details on the Planning Implementation

For the purpose of planning this implementation uses a modification of the JSHOP2 planner. The Java Simple Hierarchical Ordered Planner 2 is a domain independent HTN planning system [15].

JSHOP2 uses *ordered task decomposition* in reducing the HTN to list of primitive tasks which form the plans. An ordered task decomposition planner is an HTN planner that plans for tasks in the same order that they will be executed. This reduces the complexity of reasoning by removing a great deal of uncertainty about the world, which makes it easy to incorporate substantial expressive power into the planning algorithm. In addition to the usual HTN methods and operators, the planners can make use of axioms, can do mixed symbolic/numeric conditions, and can do external function calls.

In order for the JSHOP2 planer to generate plans it needs tree crucial components: Domain, State and Tasks. The Domain defines all the functionalities that the particular domain offers. These are simple and complex tasks. The complex tasks also called methods create the hierarchy with the fact that they can be evaluated by simple tasks of other complex tasks. This is how a hierarchical structure of tasks is formed. The problem reduction is done by reducing the high level complex tasks to simpler until all the tasks are primitive. The list of primitive tasks forms the plan.

The State represents the state of the system. It is a simple database of facts that represent the state of the system. The State is necessary to determine the way the problems or tasks are reduced to their primitive level. The reduction is done by satisfying different prerequisites set in the methods; these prerequisites are defined in the state. The Tasks are high level tasks or methods defined in the Domain. The planner based on the State and the goals selects one or more high level tasks that need to be reduced to plans [Figure 15].

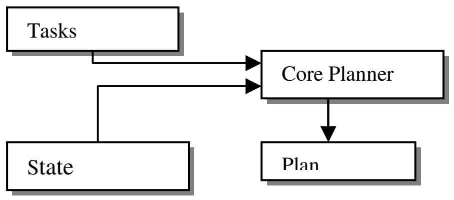

Figure 15: Diagram of a Planner

The plans then generate the game moves. The number of moves generated by the plans is just a fraction of the possible moves at that point. This reduces the game tree providing the opportunity to generate smaller and deeper game trees and making more efficient decisions in general.

7. Conclusion

Even though the results from the CBR database are not complete at this time partial strategies are implemented as cases and recognized during game play by the CBR system. These smaller local strategies coupled with more global higher level strategies that are particularly important at the beginning of the game would form a complete CBR database and represent a knowledge engineered style of playing of the AI player.

The AI Planning approach is a proven method by the tic-tac-toe experiment and is suitable for implementing the strategies associated with the CBR cases.

This approach in general benefits from both technologies, CBR as well as AI Planning and comprises an elegant solution. Even though AI Planning can be enough as a single technology for some simpler problems like tic-tac-toe the complexity of Monopoly would mean that the Planner would have to incorporate