**ABDiSE Extensibility Design Document**

The agent types of ABDiSE 1.0 is fixed after the compilation has been done. If user wants to add new disaster scenarios or different agent type behaviors, the source code needs to be edited and recompiled. The user can not add or change newly created disaster scenarios without having ABDiSE source code. To solve this problem, ABDiSE 1.0 is redesigned for extensibility; the next version is named ABDiSE 2.0.

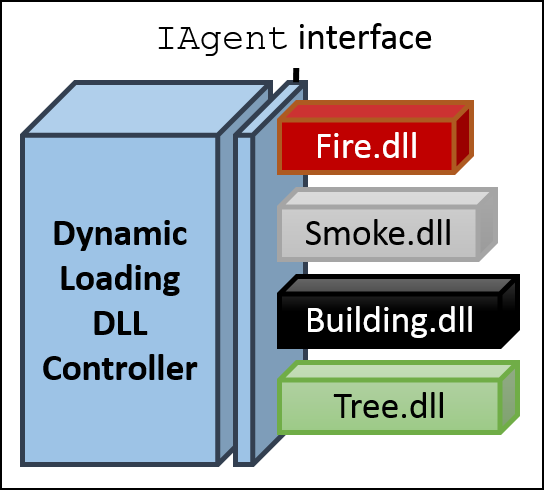
ABDiSE 2.0 is extensible: Agents and external simulators needed to model elements and dynamics of new disaster scenarios and define behaviors and interactions of agents can be added without requiring recompilation.

ABDiSE 2.0 allows user to add agent types, models, and simulators which they need. Moreover, user can create new agent types by override the abstract class Agent. The new agent code needs to be compiled as DLL (Dynamic Link Library) functions. When ABDiSE 2.0 framework is restarted, user can select new created agent types, and deselect agent types they do not need. This extensibility mechanism can be achieved by improving original controller to Dynamic Loading DLL Controller and creating the function pointer for the interactions of different agent type classes.

Below is the abstract class Agent pseudo code:

(TODO: rewriting)

ABDiSE 1.0 needs redesign to implement this new mechanism, including the Dynamic Loading DLL Controller and rewrite all agent classes to override new abstract class. The version number of this extensible ABDiSE is 2.0. In ABDiSE 2.0, different agent types are in different classes, and all of them derive from the abstract class. After agent type classes are compiled to DLL functions, Dynamic Loading DLL Controller can recognize them. The relationship of Dynamic Loading DLL Controller and different agent type classes is shown in Figure 1.

Figure 1. Dynamic Loading DLL controller and Agent classes (TODO: change image to abstract class)

When ABDiSE 2.0 is restarting, the user needs to choose agent types. In default configuration, fire, smoke, building, and tree agent types are selected. These classes are all derived from abstract class Agent and have been compiled to DLL functions.

Different disaster scenarios need different agent types, the user can select DLL functions at this moment. In this step, user can choose new created agent type DLL, too.

Figure 2. ABDiSE 2.0 DLL configuration GUI Screenshot (TODO)

After DLL configuration, the controller got information of selected agent classes. Figure 2 shows an example of loading variables and methods in DLL files.

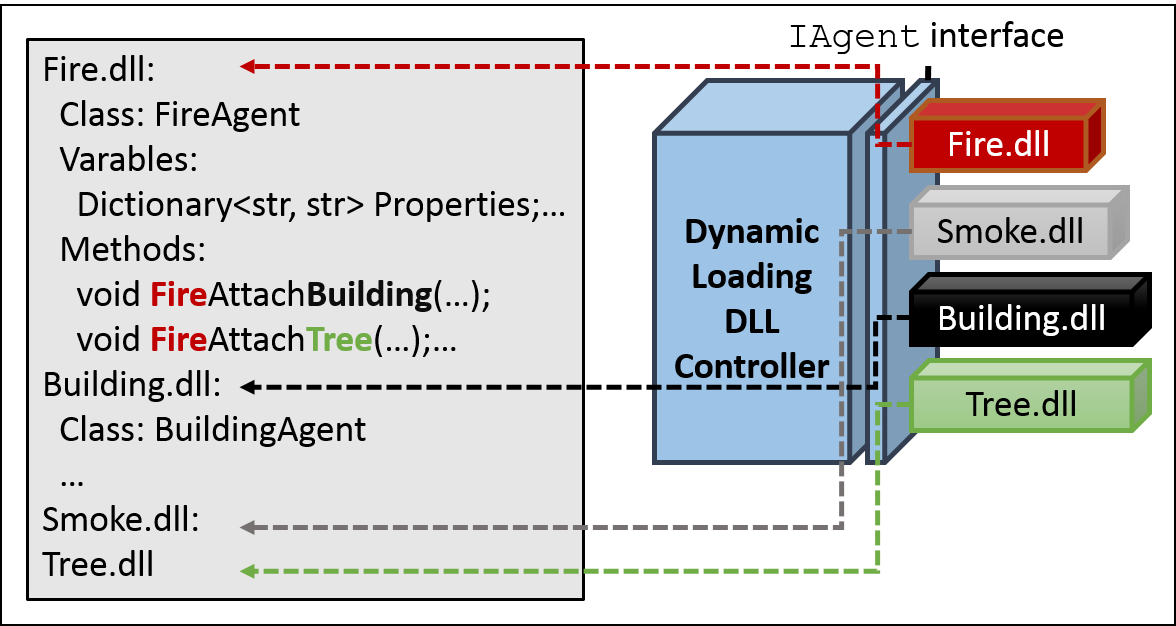


Figure 2. Example of loading classes, variables and methods in DLL files

Now the Dynamic Loading DLL Controller knows about behaviors of a single agent. In order to simulate interactions between two distinct agents from different agent types, a table with correct information of agent interaction behaviors is needed. The Agent Interaction Table includes various function pointers, which links to functions (methods) in compiled DLL files. Dynamic Loading DLL Controller can create function pointers to correct interaction methods automatically. Because there is only one possible option for one function pointer in Agent Interaction Table if user followed the rules of writing agent type classes. Therefore the controller can automatically assign correct links to methods in selected DLL files. Figure 3 presents an example of created Agent Interaction Table.

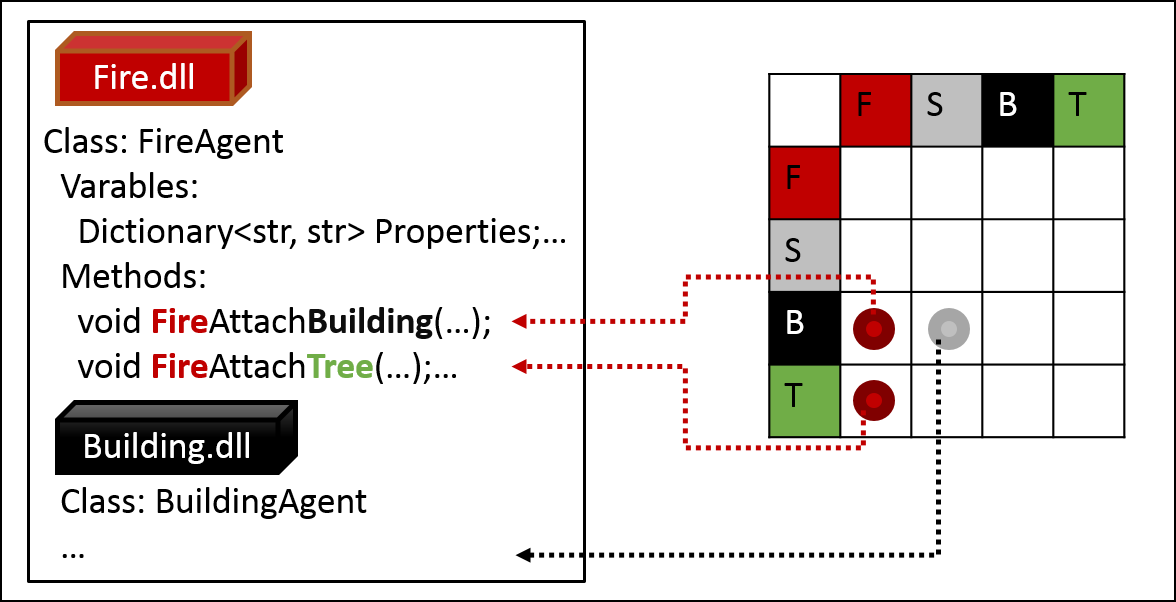


Figure 3. Example of automatically created function pointers in Agent Interaction Table

After the creation of Agent Interaction Table has finished, Dynamic Loading DLL Controller can trace correct function pointer from the table, and then execute the instructions in methods which the pointer links. While simulating agents’ interactions, Dynamic Loading DLL Controller looks up the table and executes correct methods. The path is similar to the example in Figure 3.

The initialization steps of ABDiSE 2.0 are listed below:

1. Ensure all needed agent type classes in disaster scenario which inherits abstract class Agent and compiled in DLL functions. If a class is newly created, ensure its methods follow the naming rules in document.
2. Select needed DLL in configuration window.
3. Dynamic Loading DLL Controller loads the content of DLL files automatically.
4. Dynamic Loading DLL Controller starts to building Agent Interaction Table, assigns function pointers automatically.
5. ABDiSE 2.0 initialized. Ready to simulate agent behaviors in the custom scenario.