Appendix A. The CODE Model

The CODE model is represented as an abstract syntax tree, the nodes of which are C++ objects of a class derived from a special tree class (cAbsTree). The class definitions in concise form follow. Method signatures and other implementation specific attributes have been deleted. The intent is to show the attributes that define a CODE program.

Trivial aspects of the abstract syntax tree have also been omitted. For example, the classes that represent expressions, statements, and some type definitions have been left out. These are represented in a manner that is typical of common procedural programming languages such as C.

The classes are listed bottom-up to avoid forward reference. It may make as much sense to read this appendix backwards, starting at the last page. That is where you will find the definition of "Program."

Function signatures

```
class cFuncArgSpec : public cAbsTree {
   MString TypeName;
   MString ArgName;
   int Reference;      // True if passed by reference
};


class cFuncSig : public cAbsTree {
   MString FuncName;
   MString ReturnType;
   int DontCheckArgs;      // for syntax foo(...)
   cObList Args;           // list of cFuncArgSpec
};
```

Arcs

```
class cArc : public cAbsTree {
   MString Name;
   cSType *BaseType;
```

```
   int ConnectsNSRel; // 1 iff arc is an nsrel rather
                      // than dataflow arc.
   int ArcType;
   int FromUID;
   MString FromPort;
   cObList CallVarList;              // List of cVarDecl
   cObList NodeVarList;              // List of cVarDecl
   cObList PortVarList;              // List of cVarDecl

   int ToUID;
   MString ToPort;
   cObList CallExprList;             // List of cExpr
   cObList NodeExprList;             // List of cExpr
   cObList PortExprList;             // List of cExpr
};


class cTypeDef : public cAbsTree {
   MString Name;
   cAbsTree *Type; // cNamedType, cArrayType, or cStructType
};
```

Classes for variable definitions

```
class cVarDecl : public cAbsTree {
   MString VarName;
   MString TypeName;
   cSizeSpec *Sizes;
};
```

Name Sharing Relations

```
class cNSRel : public cAbsTree {
   MString Name;
   cObList FuncSigs;   // list of cFuncSig
   void *FuncDefs;
   cObList SharedVars; // list of cSharedVar
   cObList LocalVars;  // list of cVarDecl
   cStmt *InitComp;
};
```

UC classes

```
class cReplSpec : public cAbsTree {
   MString IndVar;
   cExpr *Bound;
};


class cReplThing : public cAbsTree { // Class is abstract
   cObList Repls;     // List of cReplSpec;
   cExpr *ReplCond;
};


class cGuard : public cReplThing {
   cExpr *Cond;
};


class cArcInput : public cReplThing {
   cExpr *PortExpr;
   cExpr *VarExpr;
};


class cBinding : public cReplThing {
   cAssignStmt *Bind;
};


class cArcOutput : public cReplThing {
   cExpr *PortExpr;
   cExpr *ValueExpr;
};


class cRule : public cReplThing {
   int FiringRule;
   cObList Guards; // List of cGuard or cArcInput
   cObList Binds;  // List of cArcOutput or cBinding
};


class cPortSpec : public cAbsTree {
   MString TypeName;
   MString PortName;
```

```
   int IsInputPort;
};


enum NSClass {NSReader, NSWriter};

class cNSPortSpec : public cAbsTree {
   MString TypeName;
   MString PortName;
   NSClass UseRule;
};


class cUC : public cAbsTree {
   MString Name;
   int TermNode;
   int StartNode;
   cObList FuncSigs;
   void *FuncDefs;
   cObList InputPorts;  // List of cPortSpec
   cObList OutputPorts; // List of cPortSpec
   cObList NSPorts;     // list of cNSPortSpec
   cObList LocalVars;   // List of cVarDecls
   cStmt *InitComp;     // can be any kind of statement
   cObList InRules;     // List of cRule
   cStmt *Comp;         // can be any kind of statement
   cObList OutRules;    // List of cRule
};
```

Classes for Formal Parameters

```
class cIntNode : public cAbsTree {
   MString Name;
   MString Type;
   int InParam;  // TRUE if an in parameter, FALSE otherwise
};


class cCrepNode : public cAbsTree {
   MString Name;
   MString Type;
};
```

Call Nodes

```
class cCallNode : public cAbsTree {
   MString Name;
   MString CalledGraph;
};
```

Graphs

```
class cGraph : public cAbsTree {
   MString Name;
   cObList Types;       // list of cTypeDef
   cObList FuncSigs;    // list of cFuncSig
   void *FuncDefs;

   cObList IntNodes;    // list of cIntNode
   cObList CrepNodes;   // list of cCrepNode
   cObList UCNodes;     // list of cUC
   cObList CallNodes;   // list of cCallNode
   cObList NSRelNodes;  // list of cNSRel
   cObList Arcs;        // list of cArc

};
```

Programs

```
class cProgram : public cAbsTree {
   MString Name;
   cObList Types;    // list cTypeDef
   cObList FuncSigs; // list of cFuncSig
   void *FuncDefs;
   cObList Graphs;   // list cGraph
};
```