Rule inferencing and object orientation: an application to boundary element mesh generation

N.K. Salgadoa, M.H. Aliabadia, R.E. Callanb

aWessex Institute of Technology, Ashurst Lodge, Ashurst, Southampton
bSouthampton Institute of Higher Education, East Park Terrace, Southampton, UK

Abstract

This paper describes an Object-Orientation and Rule Inferencing approach to control the Boundary Element Mesh generation in a Damage Tolerance Design System.

A general inference engine is used to apply a set of rules, describing how the Boundary Element mesh must be generated, to a database, containing information that defines the structural configuration.

1 Introduction

Fatigue crack propagation is of great importance in aircraft damage tolerance assessment. Air frame designers are often faced with the task of determining how cracks, that are present since the very moment the aircraft is put into service, will propagate during it's operational life. Figure 1 illustrates a typical fuselage panel stiffened with longitudinal stringers and transverse frames.

Numerical simulation via the Finite Element (FEM) or Boundary Element method (BEM) is extensively used to predict damage tolerance behaviour. It requires the creation of a model, usually called mesh, describing the geometry of the structural configuration in a discretized form, in terms of elements and nodes. The mesh plays a fundamental part in assuring that results obtained from the numerical simulation are accurate.

Crack propagation is an incremental analysis. Once the initial mesh is generated, a numerical analysis is performed and it's results are used to

*on leave from EMBRAER S.A., Brasil
predict the crack growth. The mesh is then updated to reflect the changes to the cracks and a new analysis is performed. This cycle is repeated until crack sizes exceed critical values. If the designer needs to change the initial configuration, in order to improve damage tolerance characteristics, then the whole process has to be repeated.

The work described in this paper is part of an effort to develop a computer system that will integrate and support all tasks involved in the process described above, from the initial definition of the structural configuration to the crack growth simulation, including the generation and updating of the mesh. Due to the different nature of these tasks, any single representation formalism or paradigm fails to efficiently capture the complexity of the whole problem domain. For example:

- Object Orientation provides a good representation for the structural configuration objects such as the sheet, stiffeners, cracks and their interaction.

- Procedures are well suited for algorithmic tasks like the crack growth simulation.
Rules are ideal to describe some of the problem solving activities such as the mesh generation and updating strategy.

In this paper, it is demonstrated how rule inferencing and object orientation can be combined to control the generation of Boundary Element meshes to be used in the analysis of cracked stiffened panels [1].

2 The problem domain

From the BEM mesh generation perspective, the relevant objects in the problem domain are the panel, the sheet, the stiffeners and the cracks. The panel is geometrically defined by its boundaries. Boundaries are formed of segments, which can be either line segments or arc segments. The BEM mesh is a set of elements used to discretize each of the boundary segments in the problem. Figure 2 presents the relevant aspects of the problem domain object model in Rumbaugh et al. [2] notation.

![Figure 2: Problem domain object model](image-url)
Generating a BEM mesh means choosing the size and type of the Elements used in the discretization of each boundary segment. A BEM specialist would usually describe this task using a set of rules of thumb, such as:

- IF:
  - Segment belongs to an edge Crack;

- THEN:
  - Use discontinuous quadratic elements;
  - Start at the crack tip with an element sized 15% of the total Crack length;
  - Progress towards the Boundary increasing the size of the Elements according to a geometric progression.

The mesh generation can be implemented in a rule-based fashion. This solution is attractive in this case because:

- The problem solving knowledge maps directly to the implementation strategy;
- Complexity can be increased by the addition of new rules;
- The rule set can reside outside the program, allowing different strategies, based on precision requirements for example, to be implemented just by switching to an alternative rule set file.

For more information about Boundary Element mesh design see Portela et al. [3].

3 Rule Inferencing, Object Orientation and C++

Rule inferencing in C++ can be provided by a third party product such as Clips [4]. However, in order to keep the impedance between the object representation and the rule representation low, we decided to build an inference engine using object oriented concepts.

Object oriented implementations of an inference engine have been presented elsewhere, see for example Franke [5], Miranker et al. [6] and Czejdo et al. [7]. In the project presented in this paper the intention has been
to prototype an exploratory system so that we can investigate a number of possible strategies in order to arrive at an optimum design.

An inference domain class library was developed. It's use allows objects of the problem domain to take part indirectly in the inferencing process. Object member functions can be called from within a rule by making small changes to the original object classes.

The objects of the inferencing domain (see figure 3) are the constant, variable and compound terms. The key interaction between these objects is unification. Constants can be of type: string, integer, float or any other user defined type. A constant term will unify with any other constant term of the same type according to the unification protocol for that specific type. For example: an integer constant will unify with another integer constant if the two integer values that the constants encapsulate are the same. For user defined types, such as points and segments, suitable unification functions have to be provided.

A variable term can be typeless or have a type. In the first case, it can be bound (through unification) to any term. In the second case, it can be bound only to terms of the same type. Compound terms are used mainly to define predicates. For example: "edge(Crack)" or "discretely_attached(Stiffener)". A compound term unifies with another compound of the same type (e.g.,
edge, discretely attached) if all the components unify individually.

Essentially, a constant term can be viewed as an extension to the idea of a class which allows an object to take part in the inferencing process. In order to use object member functions, the programmer creates an interface function for the user defined class. The interface function is designed to circumvent C++ strong type checking and to make use of the virtual mechanism. The requirement for a client program to provide a specific function is common practice in C++ library implementations. A more advanced approach could use a pre-compiler. The interface member function takes the form:

```plaintext
function(NAME,ARGS,RETURN_VALUE)
  switch NAME
    case: "function_A"
      call function A
      return
    case: "function_B"
      call function B
      return
  ...
```

ARGS is a compound term that can take an unspecified number of function arguments. RETURN_VALUE is a variable term that will be bound to the value returned by the function.

The interface function represents the only significant extension to the problem domain classes and is necessary only for the types for which member function execution will be required.

4 Implementation of the Mesh Generation

The mesh generation task is described by a set of rules similar to that presented in section 3. These rules define the meshing strategy to be used for each segment according to its type and relative position. For example: an external boundary segment neighbouring a crack boundary segment, must be meshed starting at the point common to both segments, with an element which is of size similar to the neighbouring crack element and proceeding towards the other extremity using an arithmetic progression. Once the strategy is chosen, the actual meshing is performed by a segment member function that is called when the rule is fired. Polymorphism is used to make sure that the proper line segment or arc segment function is selected at run time.
When the mesh generation is required, a temporary database containing the description of the current state of the panel in terms of inference domain objects is generated. After the database is prepared, the inference engine cycle is executed. In this process, the database is searched using a goal driven depth first algorithm [8].

Figure 4 presents an example of a BEM mesh generated for a cracked stiffened panel.

Figure 4: generated BEM mesh for cracked stiffened panel

5 Conclusions

The rule based strategy presented can potentially tackle most of the different mesh generation situations likely to be present in the Damage Tolerance Design problem. The resulting system is well suited for incremental implementation. This is done by adding new rules to the rule set and, sometimes, new facts to the database.
The generation of the temporary database is not computationally expensive. Performance is highly dependent on the efficiency of the search algorithm and has proved to be acceptable. However, the inference engine will not be the bottleneck, in terms of performance, in a system executing numerical simulation such as the one described here.

The work carried out so far has provided the basis on which to build an optimum system.

References


