AN ADAPTABLE METHODOLOGY FOR AUTOMATION APPLICATION DEVELOPMENT

Christian Van der Velden*, Cees Bil*, Xinghuo Yu and Adrian Smith*****
***School of Aerospace, Mechanical and Manufacturing Engineering, RMIT University, Melbourne, Australia**
****School of Electrical and Computer Engineering, RMIT University, Melbourne, Australia**
*****451° Consulting, Australia**

Keywords: *Knowledge Based Engineering, Design Automation, Routing*

Abstract

The automation of engineering processes through Knowledge Based Engineering and Design Automation methods and technologies can provide significant savings in project scheduling and cost, increasing competitiveness in a changing aerospace market. In this paper we present outcomes of a research project aimed at improving engineering automation capability through development of a tool for automatic rule based path-finding for the complex engineering task of aircraft electrical harness and pipe routing.

1 Introduction

The benefits of employing automation technologies in engineering are clear from the literature [1], [2]. Automated solutions can be used to reduce low level and repetitive tasks, integrate tools and datasets, and simplify and standardise more complicated processes, achieving significant savings in development lead time and cost. Gathering and implementing knowledge electronically can also ensure knowledge retention within organisations, independent of changes in personnel.

Knowledge Based Engineering (KBE) and Design Automation are two sets of methodologies and technologies for automating engineering processes through software. KBE refers to the capture and modelling of rules and engineering knowledge for implementation in intelligent systems which automate processes and emulate human decision making. KBE applications are typically reusable, dynamic, generative, generic, and integrated. By comparison, Design Automation refers to the automation of relatively straight forward, sequential steps in an engineering process. Resultant Design Automation applications are generally applicable to specific situations with limited reuse, and often contain hard coded rules and knowledge.

The decision to implement either a KBE or Design Automation solution to satisfy an industry need depends on a number of factors. While KBE solutions generally offer more flexible and intelligent results, development schedules and costs are often inhibiting. Design Automation solutions, although often lacking dynamic capability, can often represent a more practical solution to a problem in terms of technical feasibility, time and cost, and with reduced risk.

A flexible methodology for the development of engineering automation applications of varying complexity from high level KBE to lower level Design Automation is proposed. This methodology is based on the premise that Design Automation and KBE application development methodologies are not mutually exclusive, instead the former represents a subset of processes required for the later. Whereas these two development techniques with seemingly opposing requirements are generally treated independently by industry and academia, the

proposed methodology links them together in a practical way.

The methodology associates the many processes required for development of a full KBE application with a set of governing attributes which are used to distinguish between the two types of applications. These distinguishing characteristics relate to capabilities of target systems. System developers specify the attributes desired of automated solutions, and based on their selection, sub-processes of the full methodology are either invoked or omitted, resulting in a comprehensive development processes without unnecessary low-value tasks.

In this paper a brief overview of the proposed methodology is presented, and is applied to the complex domain of aerospace electrical harness routing. A software tool for automating the layout of aircraft electrical harness and pipes is developed.

2 Knowledge Based Engineering in Aerospace

KBE and Design Automation solutions have been implemented in the aerospace and automotive industries since the early 1980's through the implementation of customised Computer Aided Design (CAD) based applications for automating rule based design tasks. Many of these early systems were developed using the ICAD system from KTI (and later Dassault Systemes) [3].

Over the two to three decades that followed, new methodologies and technologies were introduced, and after a short decline in the early 1990's, the use of KBE and Design Automation began to gain momentum in engineering industries worldwide. In this time, the technology has developed far beyond applications solely for implementation on CAD systems, incorporating automated analysis techniques and the automatic generation of manufacturing data for CAM systems.

Since the early 1990's, the aerospace industry has migrated to a wholly digital approach to development of aircraft, with the Boeing 777 wide body, twin engine passenger airliner the first developed entirely in a virtual environment [4].

As a result of technology improvements making virtual product development possible, large aerospace Original Equipment Manufacturers (OEMs) such as Boeing, Airbus, BAE Systems, Lockheed Martin, Northrop Grumman, and others, all implement design automation at some level in product development.

Over time as these engineering organisations grow and mature, a considerable base of engineering knowledge and expertise of product development capability is established. This base provides the company with the resources, technical capability and confidence to attain a share of work in new projects. Such knowledge is a valuable asset and must be managed effectively to remain competitive.

Experienced companies establish standard methodologies for design and analysis processes, using proprietary (often empirical) data. Other techniques include development of handbooks, best practice guides and software templates. This knowledge, which can be articulated with relative ease, is explicit in nature. However, unavoidably, much of a company's knowledge asset resides in the expertise and experience of the engineering staff themselves. This knowledge is tacit in nature, often difficult to express in words or on paper, and requires a deep understanding of the problem domain. This presents a problem of retaining tacit knowledge as engineers retire or change companies to pursue personal careers. Great care must be taken to mitigate impact to the company's knowledge base caused by such staff losses. Effective mechanisms must therefore exist to capture this knowledge. Development and deployment of such techniques for capturing knowledge remains one of the significant challenges facing knowledge engineers today.

It is important to recognize that not all processes are suitable for engineering automation. Some tasks, especially those requiring tacit human judgment, will always require some level of user input, and the effort

required to automate such processes often does not outweigh the benefit gained by automated capability. Processes that are well suited to automation typically exhibit one or more of the following characteristics [2]:

- Low level, repetitive, and/or highly manual tasks,
- Integration of tools and datasets (e.g. CAD / Computer Aided Engineering (CAE) / Computer Aided Manufacturing (CAM)),
- Automated documenting and report generation,
- Simplification and/or standardization of more complex processes.

Two main implementations of automated engineering solutions are commonly used in industry today. The first involves a formally identified task with well defined requirements, built by a team of developers. There must be a business case for developing such solutions, i.e. provide a positive return on investment (the ratio of the number hours required to perform the task completely manually, versus the number of hours to develop an automated solution and complete the task automatically) [2]. Outputs from these processes undergo rigorous testing and are formally released to engineers for use in design and analysis tasks. Resulting applications generally exhibit qualities of KBE applications.

The second type of automated solution is typically much smaller in scale, aiming to automate a particular task faced on the engineering floor. Such tasks generally require a large amount of manual work, such as manual manipulation of datasets, and passing information between different tools. These automated solutions are often developed by design and analysis engineers working on an individual basis. Many automated solutions are created by engineers who are not proficient in computer programming. Tools used to develop these automated methods can include spreadsheets, macros, databases, and APIs using simplified scripting languages. Engineers use these tools because of their familiarity and understanding of their capability and scope.

The development of such applications can provide solutions for problems which surface on the engineering floor with short notice. The ability to quickly code solutions to deal with such problems can greatly reduce the time taken to complete tasks and eliminate or reduce bottlenecks in the development lifecycle. These solutions are especially suited to tasks with high levels of repetition, or those subject to change numerous times during development.

These smaller applications are typically developed for a specific purpose, with hard-coded rules or knowledge, and reusability is not a key factor in their development, resembling a Design Automation approach as opposed to KBE.

Significant productivity improvements can be made by encouraging engineers to implement automation principles in everyday engineering design and analysis. This can be facilitated by introducing engineers to a structured process for automating engineering tasks, providing appropriate levels of training in development tools, and providing awareness of existing infrastructure such as generic code libraries for performing common tasks.

3 Adaptable Methodology for Automation Application Development

Whereas established KBE application development methodologies such as CommonKADS [5], [6], [7], MOKA [8], [9], [10] and others [11] can be very time consuming and complicated to implement, many of the principles incorporated are important to ensure sufficient coverage of the domain. Most KBE methodologies consist of approximately seven key phases which generally include the following activities:

1) Problem identification
2) Feasibility analysis
3) Knowledge acquisition
4) Knowledge modelling
5) System development
6) Validation
7) Deployment / ongoing support

Various methodologies refer to these phases in different ways, or incorporate one or more of the above phases into a single phase, but the list above forms the core of the majority of KBE development methodologies. Indeed, these seven phases are fundamental to the proposed methodology.

The main addition of this proposed methodology to previous KBE application development methodologies is an additional step implemented in the feasibility analysis phase to assess the level of complexity required of the application to satisfactorily meet requirements of the task to be automated.

To determine this level of complexity, a series of simple questions are posed to the knowledge engineer regarding the nature of the identified task and features desired of an application to automate the processes. Based on the responses to these questions, sub-processes of the seven key phases are invoked or filtered from the full KBE methodology as required to reduce unnecessary steps. This proposed methodology is termed "Adaptable Methodology for Automation Application Development" (AMAAD).

3.1 Complexity Analysis

The key feature separating AMAAD from existing methodologies is the Complexity Analysis step in the Identify phase that relates desired features of automation applications to sub-processes in the full KBE methodology.

Each sub-process in the seven AMAAD lifecycle phases is associated with the capability extended to resultant applications through a set of key attributes that distinguish between KBE and Design Automation applications.

Table 1 below compares characteristics of KBE and Design Automation applications. The majority of characteristics in this list directly oppose one another. The task of the Complexity Analysis is to determine which of these characteristics should apply to the desired automation application, thus specifying the methodology required for the its development. As the required complexity of the automation application is reduced, sub-processes relating to the reduction in system complexity become redundant and are filtered from the AMAAD, thus producing a customised methodology for developing the automation application to fulfil the identified needs.

Table 1. Characteristics of KBE and Design Automation applications

KBE	DESIGN AUTOMATION
Reusable	Problem specific, limited reuse
Generic	Hard-coded knowledge
Generative	Non-reconfigurable.
Integrated solution	Standalone applications
Detailed development required	Shorter development times
High level, more abstract	Lower level, more detailed

Based on the differentiating characteristics shown in Table 1 and the discussion of differences between KBE and Design Automation applications above, a set of six attributes are selected to describe the level of complexity of automation applications: Reusable, Generic, Generative, Integrated, Detailed, and High level. These attributes are either required of identified automated solutions or not.

To determine the required complexity of an automation application, a series of simple yes or no questions relating to each attribute are posed to the system developer or knowledge engineer. For each negative response, the related sub-processes are removed from the development methodology. The six complexity analysis questions are listed below with the related attribute in parentheses.

Q1: Will the application be used to automate a task for a single project, or a similar task on an ongoing basis? (Reusable)

Q2: Will the task be encountered in different fields or on projects where rules will vary? (Generic)

Q3: Are inputs to the system likely to change often? (Generative)

Q4: Will the software communicate with existing systems? (Integrated)

Q5: Does the task require a large amount of engineering rules and knowledge? (Detailed)

Q6: Of this knowledge, is there a lot of expert only knowledge required? (High Level)

A software applet was written to facilitate the Complexity Analysis step and output the resulting customised application development methodology for the specific automated solution to be developed (Fig. 1).

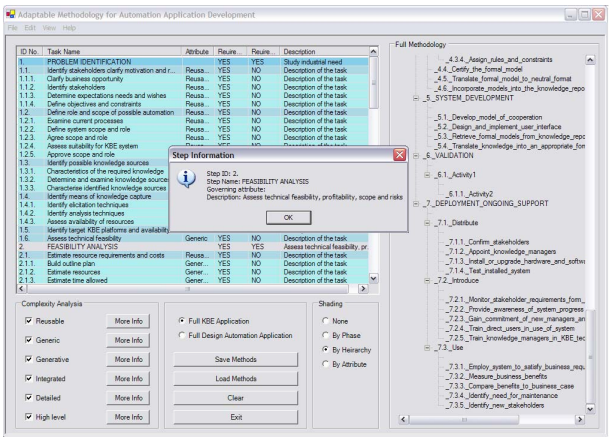

Fig. 1. AMAAD software tool

At this stage in the AMAAD development, the majority of sub-processes for the seven phases have been translated from corresponding MOKA Route Map phases for implementing the MOKA methodology [10]. Techniques for modelling knowledge have also been adapted from the CommonKADS methodology [5]

4 AMAAD Applied to Aerospace Harness Routing

This section provides a practical example of applying the AMAAD methodology to the domain of aerospace harness routing. Due to space limitations, not all sub-processes will be detailed in this paper. A brief overview is given below.

4.1 Problem Background

The aerospace electrical harness routing task is a complex problem that is faced on each aircraft development program. The increasing complexity of aircraft electrical systems in recent years has led to an increase in the number and size of electrical harnesses required to connect subsystems and equipment throughout the airframe. Wiring looms are typically comprised of hundreds of harnesses, which are generally manually routed by experienced engineers using personal knowledge and experience of the problem domain.

By way of example, the Airbus A380 aircraft has the equivalent of approximately 800 kilometres of electrical cables, the majority of which are designed manually. The design and assembly of the electrical wiring system has been the cause of major delays in delivery of this aircraft to customers [12].

The installation of wireless systems onboard aircraft as an alternative to wired systems is not a trivial matter, evidenced by Boeing's decision to install a wired entertainment system on the Boeing 787 Dreamliner aircraft rather than the originally proposed completely wireless configuration, due to weight, complexity and bandwidth problems [13]. Indeed the majority of aircraft systems will remain hard wired for some time to come

Also adding to the size and complexity of the problem, subsystem design (including the wiring system) is often conducted in parallel with principal structural design in large scale projects. Therefore changes in structure and subsystem layout occurring over the development phase can impact wiring looms, requiring time-consuming and expensive rework.

Major aerospace companies often have proprietary standards and practices for harness routing, which often varies for different aircraft development programs depending on requirements.

The generic process for harness routing involves manually creating a set of points in the CAD structural model at which the harness will be clamped to the main structure. Following this, the spine of the harness is passed through these points; ensuring sufficient clearance from structure, subsystems, moving parts, areas of high heat, and harnesses of certain categories. The process can be largely trial and error, and often the only way to determine whether sufficient clearance has been allowed, is to make manual measurements in the CAD model which can be time consuming.

These characteristics make the routing task a prime candidate for process automation.

4.2 Methodology

4.2.1 Problem Identification and Analysis

The first phase of the AMAAD methodology, "*1: Identify*", consists of problem identification and an initial investigation of feasibility and scope of a possible automated solution.

The first sub-process of the Identify phase translated from [10] is "*1.1: Clarify Motivations & Objectives*". Under this sub-task, a number of smaller activities are defined to explore the possibility of developing an automated solution to address a capability gap in a business process.

The business opportunity proposed is a system to automatically route electrical harnesses through aircraft structures. It is anticipated that this system will implement path-finding methods from existing domains including Integrated Circuit routing and Artificial Intelligence (AI) in computer games, modified for use in an aerospace context. The objectives can be stated as:

"The automatic definition of routes for electrical harnesses or other medium through obstacles including structure and systems, that satisfy relevant design rules and constraints, with a reduced lead time compared to the equivalent manual process."

The second task, "*1.2 Define Role & Scope*" from [10], investigates current processes for completing the task and further investigates suitability of an automation system. These requirements are translated into a more formal definition of the proposed system in terms of scope and boundary.

4.2.2 Complexity Analysis

Following task and scope identification, the third sub-task of the Identify phase, "*1.3 Complexity Analysis*", is conducted to establish attributes required of the automated solution and the sub-processes from the full methodology to be followed for its development.

The AMAAD software tool is used to facilitate this step, guiding the user through the complexity questions. Based on the initial objectives and scope the system, the required attributes of the system are: Reusable, Generic and Generative, drawing positive responses from the corresponding complexity questions. These required complexities are detailed as follows.

- The automated routing application will be designed with reusability as a major requirement. It is required to accept any arbitrary set of geometry (in a given format) and is not case-specific.
- The application will be generically applicable to any number of different routing domains with addition of new rule libraries (e.g. electrical harnesses, hydraulic/pneumatic pipes, fuel lines, etc.).
- In the event of changes in geometry, minimal effort will be required to reproduce paths. Session files for sets of harnesses will be stored, such that when geometry is modified, an update process can be run and the routing job re-executed.

The remaining attributes: Integrated, Detailed and High Level are not required by the proposed automated solution.

- The application is not required to integrate into existing frameworks and is designed to be independent of existing software proprietary formats. Geometry will be described in a discrete neutral format, and results output as a platform-independent CAD model.
- The majority of rules to be implemented within the system can be reduced to instances of a number of rule types, reducing the requirement for a detailed knowledge base.

Based on these complexity results, a customised methodology for the application development is output from the AMAAD software applet. All processes relating to Integrated, Detailed and High Level attributes from the methodology.

4.3 Knowledge Modelling

The representation of such knowledge in KBE applications is one of the most critical tasks in developing automated solutions. The fourth phase, "*4: Knowledge modelling*" organises knowledge required to describe and solve the

routing problem into a model for implementation in software.

Knowledge exists in a number of forms ranging in complexity. In the CommonKADS methodology, on which a significant portion of AMAAD is based, knowledge is considered to be of three primary types: domain, inference and task knowledge [5].

4.3.1 Modelling Domain Knowledge

Domain knowledge consists of the static concepts, relations and facts required to reason about performing tasks in a given problem domain [5]. Domain knowledge components describe the basic information necessary to define a problem and find its solutions. Domain knowledge elements are used in inferences and tasks to construct or derive more useful information.

A number of key concepts comprise the automated path-finding problem domain including: the representation of obstacles within a search space, definition of paths, rules governing path placement, and the search technique used.

All data defining a specific problem is incorporated into a single object, termed a "maze", which contains geometric data, and the definition of routed paths. The software engineering representation of the maze concept is a class, which is instantiated and populated with input geometry and path requirements when a specific case of a routing problem is executed. The maze class is a collection of a number of parameters describing a problem instance.

- Geometry is described in a discrete, regular, rectangular format. The representation of obstacles within the maze itself is defined in a three dimensional array of "maze nodes", characterised by an address and obstacle type.
- Paths to be routed are defined by end points, the type of path, and the governing rule library. Additional to these properties, completed paths are specified by a list of nodes connecting the two end points through the matrix of maze nodes.
- Lists of nodes indicate areas where rules were implemented within the search space. This is used for analysing sensitivity of the solution to particular rules and is described in more detail in the following section.

As defined in the system requirements, the automated routing tool supports path-finding in numerous routing domains. Rules are used to describe constraints governing path placement for these domains, and are of several different types. In knowledge engineering, it is desirable to define a standard form for rules from which many instances can be created. Multiple instances of these rule types with different properties describe each domain and are hierarchically represented in rule libraries.

One of the most common rule types implemented across a number of domains is the clearance rule, that specifies a minimum or maximum distance the routed path must maintain from obstacles of certain types. These rules, termed min/max clearance rules, can be formulated as shown below with parameters in bold representing the rule properties.

*"Paths with type **RoutedType** have relationship **Min/Max** clearance with obstacles of type **SubjectType** with an area of influence of **Radius** around the path."*

A weight and decay property is assigned to each rule of this type, determined internally by the system. Min/Max Clearance type rules are implemented in the system with the properties shown in Table 2.

Table 2. Properties for specification of Min/Max Clearance rule

RULE PROPERTY	DESCRIPTION
RoutedType	Harness type being routed
Min/Max	Whether clearance is min or max
SubjectType	Obstacle to be encountered for rule to activate
Radius	Rule area of effect around harness
Decay	Gradient term forcing higher influence of rule closer to rule subject
Weight	Value applied to cost function for node immediately adjacent

4.3.2 Modelling Inference Knowledge

Inference knowledge consists of the functions or operations on domain knowledge to perform basic tasks. From a logic point of view, inference functions describe how domain knowledge can be combined to derive new information [5].

Inference knowledge is defined by the operation to be handled by the inference (e.g. “compute”, “evaluate” or “verify”), and a role that describes the type of domain knowledge implemented in the function (e.g. “parameter”, “formula”) and in what capacity, either static or dynamic [5].

The specific variables accepted by inference functions are not included in their definition, but are related through ontologies (described below). For example, the main inference function implemented in the automated routing tool is the “Route Paths” function, the structure of which is given in Fig. 2. This function has a relatively simple structure to perform a calculation operation, accepting an input parameter and details of the formula to be computed, and returning a modified version of the same parameter.

The grey text labels in Fig. 2 are not included in the general definition of the inference, but are included for clarity to indicate the common context in which the inference us used in the routing tool.

The internal procedures invoked by inference functions are not relevant from a knowledge modelling perspective, and are considered at the system development phase of application development.

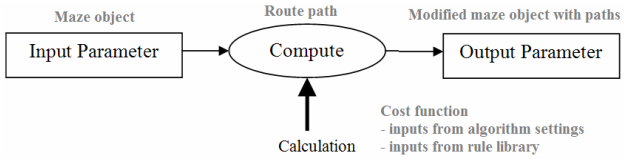

Fig. 2. Structure of “Route Path” inference

4.3.3 Modelling Task Knowledge

Task, or control, knowledge describes the hierarchical decomposition of top level tasks into a series of sub-tasks steps executed by a sequence of inference functions.

The solution of an instance of a routing problem has three primary tasks: firstly, input geometry obstacles are read into the system, secondly, paths are computed, and thirdly, results are written. Further decomposition of these tasks is given in Fig. 3.

4.3.4 Ontologies

The relationships and interactions between the three levels of knowledge is significant in the CommonKADS methodology [5]. These relationships and interactions are specified through a number of ontologies, constructed from different viewpoints and levels of abstraction. These ontologies are organised into a multi-level structure with each level representing a type of interaction [5].

The set of ontologies specifying interactions of domain knowledge and the “Route Path” inference is given in Fig. 4.

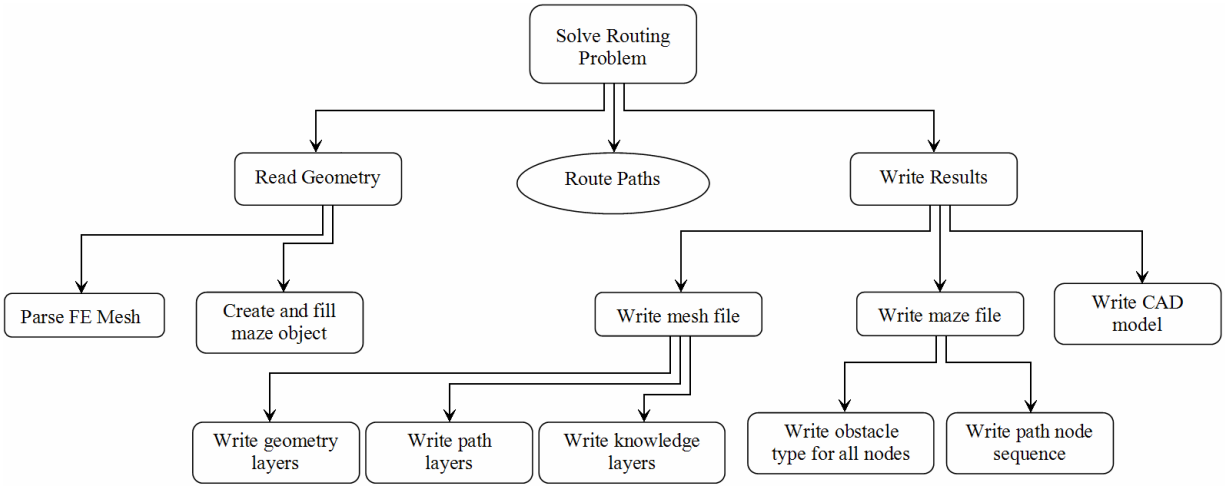

Fig. 3. Structure of “Solve Routing Problem” task

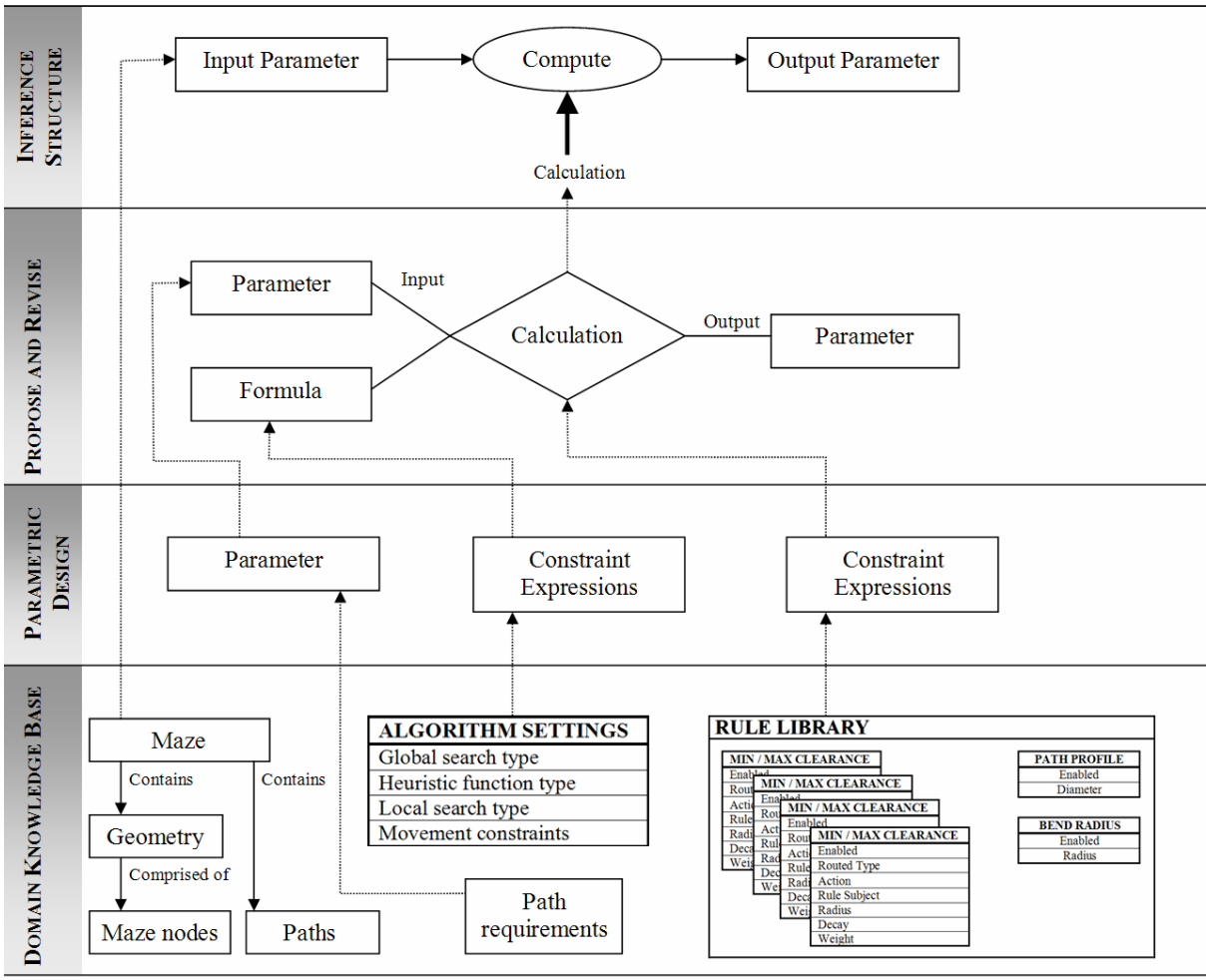

Fig. 4. Relationship between knowledge components and inference structure.

5 Automated Routing Tool for Electrical Harnesses and Pipes

This section describes the routing automation tool resulting from the customised AMAAD processes described above.

5.1 System Description

The routing system described in this paper supports path-finding from numerous routing domains including electrical harnesses, hydraulic and pneumatic pipes, and fuel lines. Engineering knowledge of each particular routing domain must be accurately represented in the system for results to be valid. Effective knowledge capture processes are therefore important to obtain an accurate and complete coverage of all rules applicable.

The automated routing system includes a rule editor for modelling domain rules. The editor consists of a simple form containing a set of controls for defining rule types, conditions for validity, area of influence, and action to be taken. Sets of rules for different routing domains and path types are stored in separate libraries in comma separated format.

Rules supported by the system currently include: bend radius, path profile, min/max clearance rules, turn penalty rules to restrict unnecessary deflections, restrictions on movement (1D, 2D and 3D movement), and clamping rules. The implementation of some of these rules within the solver is discussed below.

Solver settings, import and export options and hard-coded rules are defined within a database termed the “knowledge base” which can be customised for different routing applications. These are accessed through a “knowledge editor” included in the routing tool.

5.2 Test Model

Functionality of the automated routing tool will be described with reference to a test case consisting of a simplified model of an internal weapon storage area, typical of new generation fighter aircraft such as F-22 Raptor and F-35 Lightning II.

Weapon bays are complicated and densely populated assemblies consisting of complex metallic structure, payload envelope, and various subsystems and equipment with hundreds of interconnecting electrical harnesses and pipes. Fig. 5 shows one of the weapon bays of the F-35 Lightning II [14]. The photo shows a complex weave of harnesses and pipes throughout the length of the bay. As discussed above, the route for each harness is determined manually on CAD workstations. It does not take a lot of imagination to see that design of such a complicated loom is a very time consuming and resource intensive task.

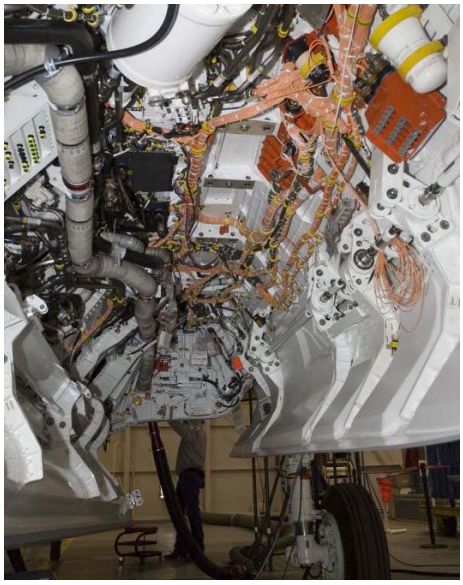

Fig. 5. Weapons bay of F-35 Lightning II [14]

The test case for the routing tool is a simplified version of the weapon bay assembly shown in Fig. 5 above. The test model was developed, based on observations of the CAD model of the F-35 weapon bay structure and subsystems. The test geometry consisting of principal structure (excluding bay doors), arbitrary subsystems, and sample payload was modelled using CAD software and is shown in Fig. 6.

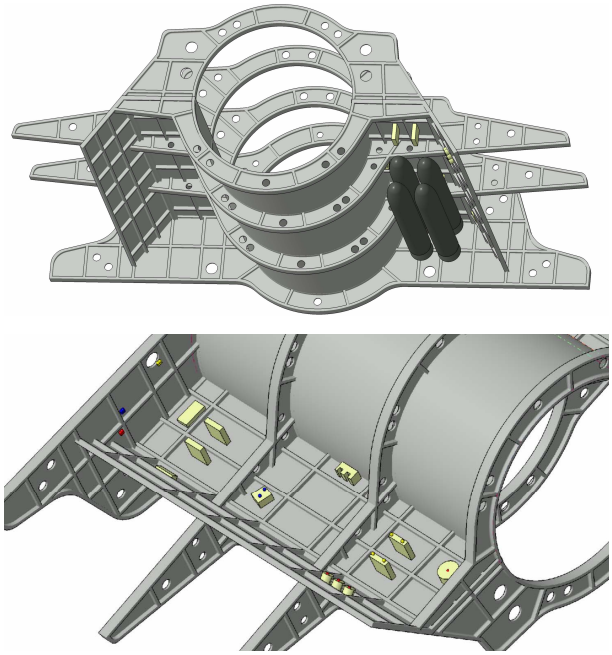

Fig. 6. Test case geometry

Geometry obstacles are represented in the routing tool as a discrete, grid-based maze object with maze nodes characterised by an integer address (in Cartesian coordinates) and node obstacle type (e.g. principal structure, subsystems, payload, harness/pipe, searchable space, etc.). The categorisation of nodes within the solver is significant as it allows various obstacle types to be treated independently by rules implemented in the path-finding algorithm.

5.3 Path Discovery

5.3.1 Routing Automation

The routing problem is encountered in numerous fields including electronics (e.g. microprocessors and printed circuit boards), navigation systems, and computer game AI. Many algorithms have been developed in these fields, but few have been tailored and applied to the aerospace domain.

The design of microprocessors employs powerful path-finding algorithms including maze routers, channel and switchbox routers, and line routers [15]. These algorithms are driven by technology improvements in component manufacture including the reducing size of interconnecting wires and number of 2D layers available for routing. Commercial software packages for multilayered printed circuit board design are also available, employing similar techniques.

Improving AI in computer games is another area which continues to drive intelligent path-finding development. The way in which non-player characters (NPCs) move and react in the game environment largely determines the realism of the gaming experience. To this end, numerous algorithms have been developed incorporating various behavioural techniques including shortest path, stealthy movement for avoiding detection, cautious movement using cover to avoid damage, etc. [16].

Developments in these fields have provided an excellent base of knowledge which can be utilized for automating the layout of aircraft electrical harnesses and pipes. The routing tool implements path-finding techniques from microprocessor routing and game AI domains, together with knowledge based techniques for capturing and modelling design rules. This enables the model to be constrained sufficiently to produce paths that accurately satisfy requirements, minimizing the amount of manual work required.

5.3.2 Path-finding Algorithm

The algorithm implemented in the automated routing system extends the popular A* search algorithm used in shortest path problems in game AI [16] and other problems.

In the basic A* algorithm, path movement is determined by evaluating a cost function for each node interrogated (Equation 1). Node cost, *f(n)*, is equal to the sum of the shortest distance from the source node, *g(n)*, and the estimated cost to the goal using a heuristic function, *h(n)*. At each iteration of the search, the lowest cost node is selected as the new node to expand. Provided the heuristic function that calculates the remaining distance does not overestimate the distance to the target, the algorithm is both optimal and complete meaning that the shortest path will always be returned [17]. Example heuristic functions are discussed in [18].

$$f(n) = g(n) + h(n) \qquad (1)$$

To tailor the algorithm to the complex rule-based domain of electrical harness design, the cost function is extended to include additional terms representing rules in domain libraries (Equation 2). In this extended cost function, the *g(n)* and *h(n)* terms are determined in the same way as normal A*. Min / max clearances rules are implemented through the *i(n)* term which modifies the total node cost, depending on the searched node's proximity to particular obstacle types, causing the path to favour obstacles of a particular type and avoid others. The magnitude of the *i(n)* term is determined by performing a local search within a given radius for each node interrogated in the main search. A separate *i(n)* term is added for each clearance-type rule in the library, *k*.

$$f(n) = g(n) + h(n) + \sum_{N=1}^{k} i_N(n) + T(n) \qquad (2)$$

For example, a rule may specify that harnesses must be routed close to principal structure for clamping. Such a rule would reduce the cost for nodes that are close to structural obstacles, resulting in a lower cost solution closer to structure. Other rules may state that harnesses must have a certain clearance from heat zones, moving parts and cables with particular electromagnetic sensitivities. Such rules increase node cost for nodes near these obstacle types, resulting in a lower cost solution away from these areas.

A turn penalty term, *T(n)*, is also introduced for models with complex geometry, minimising unnecessary deflections in the path. The contribution of the *T(n)* term is determined from the turn deflection angle. Mild turns are penalised less than harsh or backwards turns.

In the test case given in this paper, a set of 12 paths were routed, separated into three categories with four harnesses in each. Domain

rules implemented for this routing job included a structural maximum clearance rule, minimising the length of unsupported runs for clamping purposes, and clearance rules for harnesses of the same type to be run together.

5.4 Results Output

The system delivers resultant paths in three way, described below.

5.4.1 Two Dimensional View

The first output method is a set of simple 2D diagrams of routed paths and input geometry along the three major axis, given in the automated routing tool itself. This first glance provides users with a quick indication of the quality of the routing job, allowing users to determine whether paths were placed in the desired region of the solution space. For example, if a model features a number of cut-outs and lightening holes which are not properly constrained, the path may follow the outside of the structure rather than the inside. This simple preview can reduce time analysing sets of results which clearly require refinement. Several statistics relating to path quality are also displayed, including path length, number of turns, solution time, and number of nodes searched.

5.4.2 CAD Model Output

The second output method is a series of CAD-readable wire-frame IGES models consisting of straight segments connected with a user defined bend radius. The path models can be imported and integrated into the existing CAD geometry assembly, and detail added as necessary for a complete digital representation of the routed part using CAD software tools (e.g. thickness, detail to terminals, etc.).

The CAD output for the test case described above is given in Fig. 7. The results clearly indicate successful implementation of the clearance rules, with all paths maintaining close clearance to primary structure and harnesses of the same category.

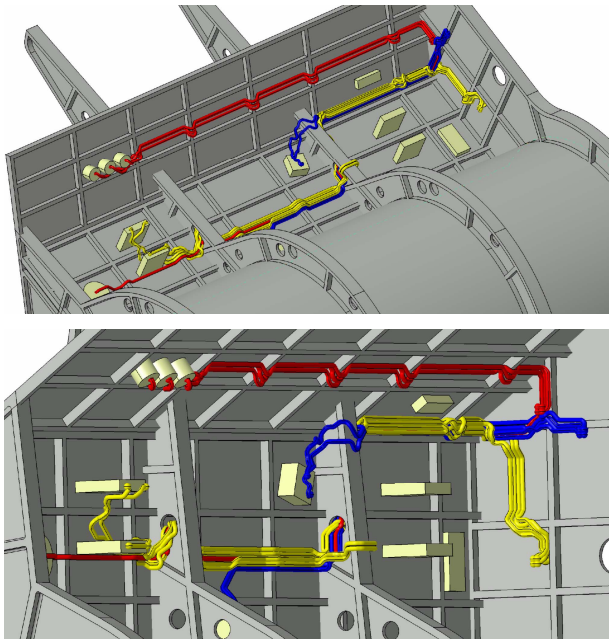

Fig. 7. CAD output

5.4.3 Discrete Model Output

The third output method is a discrete model comprised of several layers including input geometry, routed paths and knowledge implemented in the automated routing process. The purpose of this output method is to assist users in understanding the justification for path placement.

Geometry obstacles encountered in the search space are included for referencing rule implementation against input geometry. Geometry is represented by hexahedral elements colour-coded according to obstacle type. Routed path spines are represented in wire-frame using bar elements. Additional layers represent rules and search characteristics used in determining path placement, communicating design justification to the user. These layers include 3D maps showing locations where individual rules were implemented in the search space, allowing users to determine regions where particular rules are significant for any manual touching up required. A map of all nodes interrogated by the algorithm is also given, allowing users to determine whether the algorithm is considering the correct area in the search space, or whether additional constraints are necessary. The discrete model output for the test case is given in Fig. 8.

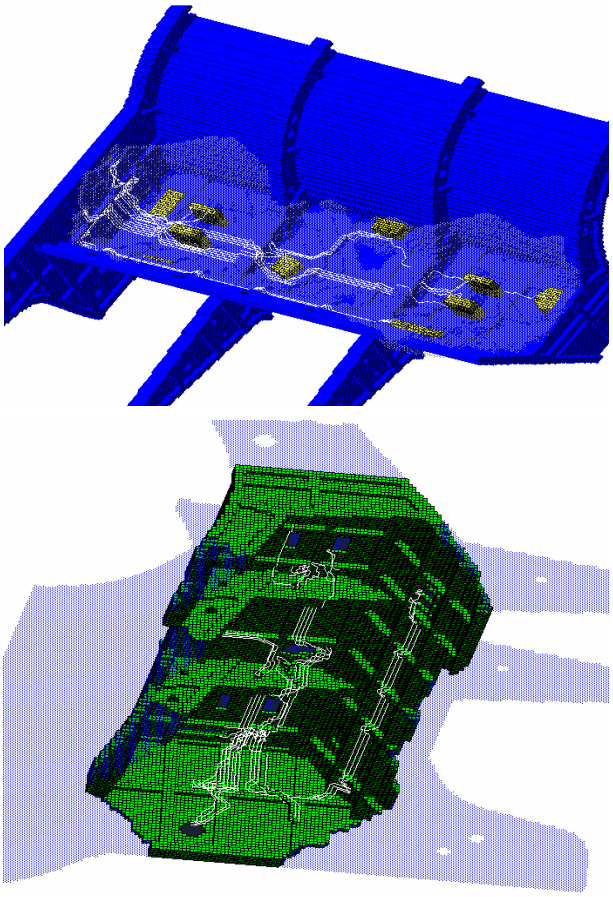

Fig. 8. Discrete model output

The discrete model is viewed using third party Finite Element Modelling (FEM) software. Various components of the solution can be activated independently using visualisation options available in the FEM software to view the interaction of particular rules. The two images above highlight different characteristics of the solution. The top image shows all nodes interrogated by the solver, referenced against input geometry. The bottom image shows all points on the reference geometry where the structural clearance rule was implemented. As expected, the majority of nodes searched were in the vicinity of points where this rule was implemented, as it was a lower cost solution for points close to primary structure. Similar outputs are available for points where the various path clearance rules were implemented.

5.4.4 Discussion

After conducting a number of test runs with the selected harness endpoints with various rule combinations, it was found that quality of resultant paths is closely coupled with the weight factor applied to individual rules. Thus a process for "tuning" the rule library for individual models is required to produce optimal results, which can be a time consuming process. Despite this, the system works very well as a proof of concept application. The solution time is sufficiently small that a number of solutions can be generated for various combinations of rules and weightings in a relatively short time compared to manual practices, allowing a large number of results to be assessed for suitability relatively quickly.

Further improvements to the system would focus on implementing an optimisation process for tuning rule weights, leading to higher quality results without the manual trial and error process for applying various rule combinations. Further work would also develop more effective methods for quantitatively assessing path quality.

6 Conclusion

Presented in this paper was a new methodology linking KBE and Design Automation application development phases through a simple complexity analysis process. Proposed solutions for processes identified for automation are assessed for the desired level of complexity in terms of several key attributes, and a customised set of steps for the application development is given.

This methodology was applied to the aerospace path-finding domain, and the development of software tool for automating the layout design of electrical harnesses and pipes in aircraft was described. The resulting routing tool successfully implements path-finding principles from microprocessor routing and game AI domains, to produce routed paths satisfying user defined design rules and constraints. Knowledge and rule editors simplify the knowledge capture process,

extending capability to domain experts to implement new routing methods and rules for application to new domains. The benefits of using such a tool in industry are clear, with solution times of a few minutes per path for detailed, high resolution models (depending on the number of rules applied, and the degree to which the model is constrained). The comparative manual design process can take upwards of three days.

In the competitive aerospace industry, project success requires rapid mobilization of resources to respond quickly to problems faced by both customers, and internally on the engineering floor. Implementation of KBE and Design Automation methods and technologies to develop automated solutions can provide this capability, delivering savings in time and cost, and gaining a competitive edge in the global engineering market.

References

[1] Cooper, S., Fan, I., and Li, G. "Achieving Competitive Advantage through Knowledge-Based Engineering" *Department of Enterprise Integration, Cranfield University*, 2001.

[2] Smith, A. L., and Bardell, N. S. "A Driving Need for Design Automation within Aerospace Engineering". *11th Australian International Aerospace Congress*, Melbourne, Australia, 2005.

[3] "Dassault Systemes Acquires KTI", *Dassault Systems Website*, [Online 2007] URL: http://www.3ds.com/news-events/press-room/release/796/1/

[4] "The Boeing 777 Program Background" [Online 2007] URL: http://www.boeing.com/commercial/777family/background.html

[5] Schreiber, G., Welinga, B., de Hoog, R., Akkermans, Hans., Van de Velde, W., "A Comprehensive Methodology for KBS Development", *IEEE Expert*, 1994.

[6] Kingston, J., "Applying KADS to KADS: knowledge based guidance for knowledge engineering" , *Artificial Intelligence Applications Institute*, University of Edinburgh, 1995.

[7] Kingston, J., "Designing Knowledge Based Systems: The CommonKADS Design Model", *Artificial Intelligence Applications Institute*, University of Edinburgh, 1997.

[8] Oldham, K., Kneebone, S., Callot, M., Murton, A., Brimble, R., "MOKA - A Methodology and tools Oriented to Knowledge-based engineering Applications" , *Volume 8, Proceedings of the Conference on Integration in Manufacturing*, Göteborg, Sweden, 1998.

[9] Brimble, R, Oldham, K, Callot, M and Murton, A. "MOKA: A Methodology for Developing KBE Applications". *Proceedings of the 8th European Conference on Product Data Technology*, Norway, 1999.

[10] MOKA Consortium "MOKA Route Map", 2000. [Online 2008] URL: http://web2.eng.coventry.ac.uk/moka/resources/Routemap/

[11] Studer, R., Benjamins, R., Fensel, D.,"Knowledge Engineering: Principles and Methods" *IEEE Transactions on Data Knowledge Engineering Vol. 25*, 1998, pp. 161–197.

[12] "Airbus confirms further A380 delay and launches company restructuring plan", *Airbus Website*. [Online 3 October 2006] URL: http://www.airbus.com/en/presscentre/pressreleases/ressreleases_items/06_10_03_a380_delays_company_res tructuring_plan.html .

[13] Gates, D., "Boeing Cuts 787 Wireless System" *The Seattle Times*, Jan 2007. [Online 2008] http://seattletimes.nwsource.com/html/businesstechnology/2003540074_boeing25.html

[14] "F-35A Manufacturing", *F-35 Joint Strike Fighter Program Website* [Online 2007]. URL: http://www.jsf.mil/gallery/gal_photo_sdd_f35amanf.htm

[15] Groeneveld, P., "Electronic Design Automation, Part 1". *Technische Universiteit Eindhoven*, The Netherlands, 2005.

[16] McCloskey, J., Miller, J., Prasad, A., Linden, L. "AI In First Person Shooter Games" [Online 2008] URL: http://emunic.emich.edu/~evett/GameProgramming/LectureNotes/FPS.pdf

[17] Russel, R., and Norvig, P., Artificial Intelligence A Modern Approach, 2nd ed., *Prentice Hall*, New Jersey, 2003, Chap. 4.

[18] Haslum, P., and Gener, H., "Admissible Heuristics for Optimal Planning" *Proceeding of the 5th International Conference on AI Planning and Scheduling, AIAA Press*, 2000, pp. 140-149.

Copyright Statement