Cover Sheet

14th ICCRTS

“C2 and Agility”

Deriving Reliable Model Revisions from Executed Plan Data Analysis

Topic 5: Experimentation and Analysis

Alice M. Mulvehill

BBN Technologies
10 Moulton Street
Cambridge, MA 02138
617-873-2228
amm@bbn.com

Brian Krisler

BBN Technologies
10 Moulton Street
Cambridge, MA 02138
617-873-4899
bkrisler@bbn.com

Renu Bostwick

BBN Technologies
10 Moulton Street
Cambridge, MA 02138
617-873-4543
rbostwick@bbn.com

Point of Contact:
Alice M. Mulvehill

Deriving Reliable Model Revisions from Executed Plan Data Analysis

Alice M. Mulvehill, Brian Krisler, Renu Bostwick
BBN Technologies
10 Moulton Street
Cambridge, MA 02138
[amm, bkrisler, rbostwick] @bbn.com

Abstract

JAGUAR is a set of tools that perform model-based planning and replanning, real-time execution monitoring, and adaptive modeling in the domain of military air operations. Model driven planners are susceptible to biases, omissions, and errors within the models. Operating in a dynamic, real-time environment requires continual model updates to meet the ever changing requirements. This paper describes how the JAGUAR Model Adaptor tool uses Case-Based Reasoning (CBR) technology, visualization tools, machine learning, and decision trees to evaluate executed plan data and drive model revisions. The paper also presents some results obtained about the Model Adaptor from several experiments that were conducted to evaluate the usefulness of the entire JAGUAR tool suite to operational users.

Introduction

JAGUAR is a set of state-of-the-art tools that provide plan generation and execution monitoring for military air and ground operations. Leading edge technologies and research incorporated in JAGUAR include model-based plan generation, real-time plan execution monitoring, replanning, and adaptive modeling. The main JAGUAR components include a Problem Definer, a Plan Generator, a Plan Monitor, and a Model Adaptor. All of the components communicate through XML information products. The Problem Definer provides the JAGUAR components with a description of the problem, which includes the objectives that the planner needs to satisfy. The Plan Generator takes each objective, information about available resources, the current plan and execution context, and references models of entities and processes to generate a plan. The Plan Monitor tracks the plan as it is being executed, updates the world state, and triggers replanning requirements.

Because JAGUAR is still under development, we use a simulator to provide a planning environment and to support monitoring of plan execution. The simulator supports context updates with data from a variety of sensor systems. When plan execution is in violation of certain environmental and/or modeling constraints, the Plan Monitor will publish an anomaly describing the violation.

In this paper we describe our experiences in developing Model Adaptor (MA) tools which help JAGUAR operators evaluate executed plan data and subsequently make model revisions. Our paper also describes a set of experiments that were conducted across the entire JAGUAR system. Only the results that pertain to the usefulness of the Model Adaptor tools are described in this paper.

CBR and Model Adaptation in JAGUAR

Automated planning systems achieve plan goals in a variety of ways. The most prevalent implementations include generative planners, case based planners, mixed-initiative planning and combinations thereof. [See 1 for an overview of planning.] Research evidence indicates that if a planning system can leverage historical experiences, the resultant planning time and expense are reduced, and the results are comparable or better than the performance of a generative planner alone.

Most planning software relies on internal algorithms and access to domain models for problem solving. The planner in JAGUAR is primarily model-driven. When a planner is model-driven, biases, omissions and errors within the models have a direct influence over the final executed plan. Analyzing executed plan data is a method that allows human operators to assess how the Plan Generator is using the models. If analysis reveals plan biases or modeling errors, the operator can modify the models to, in effect, change the biases or correct the errors.

The Model Adaptor, the focus of this paper, stores all versions of published models, the executed plans, objectives, and published anomalies for use in determining model updates. The Model Adaptor also includes a set of executed mission analysis tools that aid in the discovery and identification of planning biases and/or modeling errors. Several of these tools leverage case based reasoning (CBR) technology. While UML modeling tools are available for use by the Model Adaptor operators to assist in model creation and refinement, these tools do not help the operator evaluate model usage by the Plan Generator. Instead, diagnosis and analysis tools were developed to support this requirement. The diagnosis tool [2] analyzes generated plans as they are being executed, reading the published anomalies and evaluating them with respect to historical plan execution data. The diagnosis tool will generate model repair suggestions when it finds a correlation between some number of published anomalies and properties of the published models. The analysis tools operate on specified case features defined to describe certain aspects of plan execution.

Real-time model adaptation is a challenging problem because of what is implied by the term “adaptation” and because of the stresses introduced by real-time operation.

For example, if a person learns to drive in a moderate climate, this person will likely be faced with several challenging model update problems the first time that she attempts to drive in snowy and icy conditions. To support problem solving, a set of interdependent models will likely be invoked; and through the process of trial and error, these models will be incrementally revised and adapted to meet the newly experienced environmental conditions.

However, some problem solving environments do not allow the luxury of model adaptation by trial and error, especially if errors can result in catastrophic states that must be avoided. Additionally, in a dynamic problem solving environment, it is highly unlikely that any modeling tool will have properly and completely modeled a given problem domain; therefore, adaptation must also incorporate learning about new entities, behaviors, relationships, and constraints. A good problem solving tool must be able to adapt and learn from experience and allow operators to update models dynamically in order to respond to real-time problem solving environments and unexpected situations. Although the JAGUAR Model Adaptor does not currently support all of these objectives, it has been designed with these objectives as drivers. CBR technology was chosen to provide this element of the system's incremental learning capability [3, 4, and 5].

CBR is the process of applying knowledge from past experiences to the solution of a current problem [6]. In a CBR system, a case base is comprised of a set of cases. Each case represents an episode or event that has occurred in time and space. In order to facilitate search and highlight defining characteristics of a given case, each case is annotated by a set of case features or indices. These features are generally attribute-value pairs. The CBR approach relies heavily on similarity matching between features (indices) with information about the current problem state. For example, in the CBR logistics domain application as described in [7], when a new event such as a hurricane occurs and a disaster relief logistics deployment response is required, the CBR force deployment tool can be used to rapidly construct a new basic deployment plan. Because previously developed similar (but not exact) disaster relief force deployment plans are used to construct the new basic plan, the tool supports the adaptation of the new plan to meet the specific requirements of the current situation.

The CBR research community has developed numerous techniques for supporting similarity evaluation and partial matching in planning domains [8, 9]. In many applications, historical cases rarely exactly match the requirements for the current problem solving context. For example, a CBR tool called CASEY helps doctors diagnose heart failure in patients [10]. CASEY includes case adaptation methods that support the merging of parts of previous cases so that the resulting product—the adapted case—can be more effectively applied in the current problem-solving episode. Such adaptation is a form of analogical learning [11], another powerful learning technique that can be obtained with a CBR implementation.

Similar to the CASEY system, JAGUAR helps human military operators diagnose the usefulness of the models that support plan generation and execution by retrieving historical executed mission cases and associated models. Since JAGUAR is model-driven, if a model for some element in the world is not available, the JAGUAR Plan Generator will not be able to reason about it; if a model is incorrect, the Plan Generator will not be able to correctly solve the problem. To support model diagnosis, modeling omissions and errors that result during plan execution are detected as anomalies and deviations by the Plan Monitor. This data is provided to the Model Adaptor for use in identifying modeling errors. Data about the anomalies are stored along with the executed mission data in the form of cases. The case bases are then used by the diagnosis tool for generating hypotheses about the likelihood that plan execution anomalies are caused by modeling problems. The case data is also used by a set of analysis tools to identify both successful and failed trends and to aid operators in performing plan assessment. In this paper, we focus on the MA analysis tools.

Using Multiple Case Bases

Case creation, retrieval, publication and analysis within the JAGUAR Model Adaptor is provided through a suite of custom built software tools augmented with a set of well supported open-source products. Our CBR application uses a Tomcat Servlet Container [12] running a Xindice Native XML Database system (NXD) [13]. An NXD is a specialized database designed to store XML data and its model intact [14]. By retaining the structure of the XML data, queries can be implemented with standard XML tools that access specific elements within the data. Because our CBR application runs within Tomcat, we automatically gain client-server capabilities via the HTTP protocol, allowing multiple simultaneous connections to the case base.

All case searching operations are supported by the Lucene open-source text search engine [15]. Lucene uses the concept of a document, with each document consisting of a set of terms, representing the unit of search. In our usage of Lucene, a document is equivalent to a case, and each case feature of a case is equivalent to a unique term. Lucene also provides efficient, scalable, high-performance indexing capabilities. In the JAGUAR MA case bases, each case is indexed at the document level; therefore, all case retrieval also occurs at the document level. While we do provide the ability to retrieve a document based on a match with a document excerpt (i.e., element), if the user is interested in finding the matching content, they need to dig through the XML within the returned document to locate the specific instance of the matched term. A "Find in Document" custom designed capability is provided to support searching within the data of an XML case. If this data is determined to be of use in the long run, then a new case feature can be added.

In order to better manage JAGUAR domain data and information products, we have partitioned aspects of the problem into separate case bases. The two main partitions are the model case base, which contains all of the model versions that have been provided to support the JAGUAR system, and the realized plan case base, which contains all of the executed missions comprising a plan generated by the JAGUAR system. In the model case base, each model constitutes a case. In the realized plan case base, an individual mission within a plan constitutes a case. As JAGUAR has evolved, additional case bases have been created. These case bases include one that stores information about mission anomalies and one that stores information about the goals or objectives associated with a mission.

Each case base in JAGUAR is defined as a collection, where a collection is synonymous with a table in a relational database. The data structure of the collection is based on an XML specification. A collection can be generalized with arbitrary content, or it can be specialized through the addition of a set of domain features. By leveraging the fact that our domain data exists in XML, with a well-defined schema, the case features can mirror the XML tag names. While the usage of existing XML tags as features has gained some popularity among CBR researchers in recent years [16, 17], no standard for what constitutes the content of an XML tag exists. Since we would like some of the case features to describe functional and semantic content, we have developed a variety of other methods for creating case features that leverage the content of the case.

Previous CBR systems have leveraged XML in various ways, such as a communication protocol between client and server components [18], and as a complete case base markup language, CBML [19, 20]. Like CBML, our case base definition contains a schema that defines the data structure for a specialized case base. However, our work goes beyond the work of CBML by supporting the creation of custom features that can be derived from the underlying XML content. For these custom features, a rule defines the query used to extract the value for the feature from the data. Utilizing this approach, we can easily annotate the case data to assist with search and analysis.

Because the information products in JAGUAR are represented in XML, we gain access to a well-developed operational toolkit for XML structures. Part of this toolkit includes various standardized query languages developed for the querying and extraction of information from XML marked up data. Using components of this toolkit, we created a set of rule processors for extracting case features from the data. One of the rules we developed uses XPath for extracting specific information from the desired information product. A widely documented and supported query language for addressing parts of an XML document, XPath allows users the ability to quickly develop feature extraction queries without learning a new, custom query language [21].

Each case in a specialized JAGUAR case base contains the full JAGUAR information product (e.g., executed plan, anomaly or objective) and a set of descriptive features. Each case feature is defined by the following data elements:

- *Name* - human readable name of the feature.
- *Description* - a short description of the feature name, e.g., what it is useful for or how it is derived.
- *Weight* - the importance of the feature (primarily used for retrieval ranking).
- *Feature Type* - the method by which the feature is defined, e.g., XPath (rule-based), XSLT (transform-based), system (Java code-based).
- *Data Type* - the data type for the feature value, e.g., string, Boolean, or other.

Since each case includes the complete XML information product, analysts can add new case features to meet changing needs and requirements. When a case feature is defined using the XPath feature type, the values for the features are automatically computed either during an import of raw case data or when feature regeneration is requested by the user. The down side of this approach is that if the XML structure for the content of a defined XPath feature changes, we need to update the associated feature value XPath extraction method to handle the change.

Because the XPath query language does not support the computation of values from the aggregation of multiple paths, or across multiple information products, specialized features were developed to allow operators to create feature/value pairs that are derived from a numerical computation of the value of multiple features in any of the JAGUAR case bases. These features are harder to create within the constraints of the XPath query language, and thus require custom developed Java code; however, we find that they are very powerful. For example, we have developed a specialized feature to compute the number of anomalies recorded for an executed plan by anomaly type. Another specialized feature has been developed to compute the completeness of activities for a given executed mission. These specialized feature types are extremely useful in providing an abstracted description of a mission case for subsequent analysis.

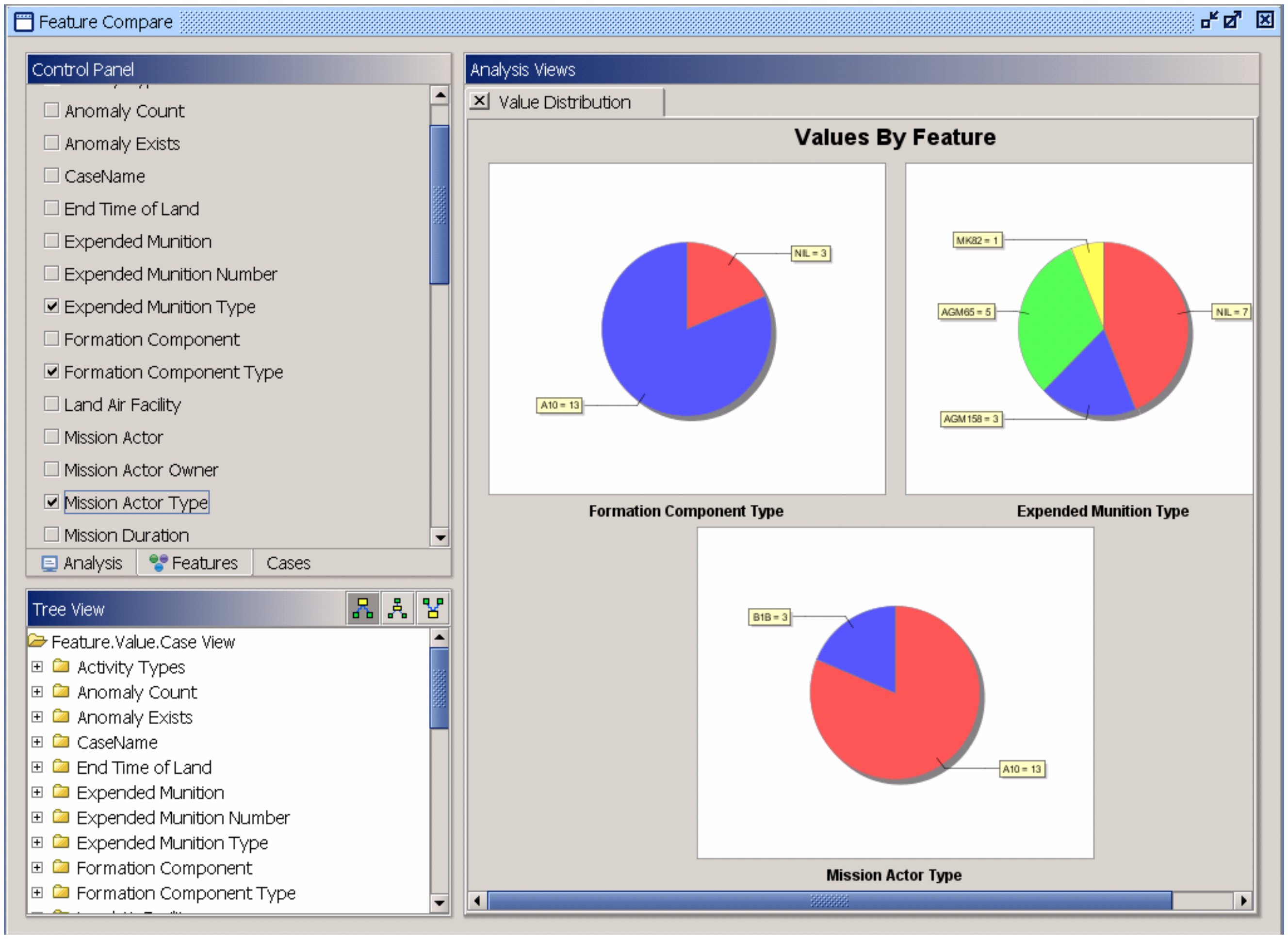
Analyzing Case Base Data

As a military operation evolves in time, the requirement for operators to efficiently collect, analyze and reflect on that data becomes more important. Additionally, patterns start to emerge that may or may not be desirable. Since the JAGUAR Plan Generator is model driven, some of the bias patterns that emerge may also be indicative of modeling inefficiencies or errors in other JAGUAR components.

To support the domain experts and Model Adaptor operators, we have developed several tools that allow the operators to view the data from various perspectives. Each of the information views makes use of case features.

One of the analysis tools developed to support analysis is a pie chart tool which is shown in Figure 1. This tool provides the analyst with a quick, descriptive distribution of the data within a specified analysis set. Through a custom developed graphical user interface called the Browser, the operator can choose a case base and either generate a set of pie charts using all of the data from the selected case base, or from some subset of cases within the case base, derived from a case base query. The pie charting analysis component of the Browser allows for dynamic filtering, providing the ability to turn on and off the views of various features within the charting analysis without the need to reload the feature data. This helps the analyst restrict the analysis space and focus upon key areas of the data. Screen-prints of the pie charts can be used to generate permanent reports.

A down side of the pie charts is that it can be difficult to recognize areas of missing or misrepresented data. Figure 1 shows an example in which correct interpretation of the data is difficult without a sophisticated understanding of the models. In this figure, executed mission data involving both A10 and B1B aircraft are being analyzed. This figure shows the value distribution of three features found within the set of data being analyzed: “Formation Component Type”, “Expended Munitions Type”, and “Mission Actor Type”. Note that the “Mission Actor Type” feature pie chart shows feature values of both A10 and B1B, but the “Formation Component Type” feature pie chart shows values of only A10 and NIL. The NIL value, added during case import when the desired feature value is missing, could indicate an instance either where the value was not provided by the planner when it should have been, or


Control Panel

- [ ] Anomaly Count
- [ ] Anomaly Exists
- [ ] CaseName
- [ ] End Time of Land
- [ ] Expended Munition
- [ ] Expended Munition Number
- [✓] Expended Munition Type
- [ ] Formation Component
- [✓] Formation Component Type
- [ ] Land Air Facility
- [ ] Mission Actor
- [ ] Mission Actor Owner
- [✓] Mission Actor Type
- [ ] Mission Duration

Figure 1: NIL values caused by entities type values not being recorded in the data.

where the feature does not exist or apply for the particular case. If more than two different values for "Mission Actor Type" existed in the case base, however, the user would find it difficult to determine which of the cases had no values for "Formation Component Type". In the simple example in Figure 1, the NIL values seen in the "Formation Component Type" feature pie chart are easily identified to be those of B1B actors. From a modeler's perspective, the B1B aircraft are known in the models to fly solo and not in formation, and thus those B1B cases correctly do not have "Formation Component Type" information. Without a detailed knowledge of the models, deciphering the missing data is not always possible.

In the third pie chart for "Expended Munitions Type" shown in Figure 1, the seven NIL values are harder to categorize. The user would have to delve further into the underlying data to find out if seven cases incorrectly flew without munitions at all, if the seven missions flew with munitions but did not use the munitions (not expended) because the mission was aborted for some reason, or if the mission was completed without using munitions that were loaded. As the case base grows, analyzing what information is missing gets more difficult.

Machine Learning With Weka

While the pie charts are useful for the high-level visual display of distributions in the data, in order for the analyst to identify correlations across features and cases, machine learning technology is being used to classify data and to generate decision trees. Decision tree technology is commonly used in many domains to find decision points. Inductive machine learning algorithms such as C4.5 support the generation of decision trees.

In an effort to determine if these types of algorithms could support our model revision needs, we have been experimenting with an open-source machine learning framework called Weka [22, 23]. Developed at the University of Waikato, Weka is a collection of machine learning algorithms (including an implementation of C4.5 called J4.8), accessible through a user-friendly toolkit that allows for complex data mining of varying amounts of data and provides several graphical visualization tools to aid in operator interpretation. Weka algorithms can produce useful decision trees, given properly formatted data. The case features in the JAGUAR Model Adaptor's case bases proved to be easily transformable into the proper format required for Weka. After some successful experimentation with select Weka tools, we developed an API between Weka and the Model Adaptor Browser so that an entire case base or selected cases from a given case base could be passed from the Browser to Weka for analysis.

One important metric resulting from the generation of a decision tree is the percentage of correctly classified nodes. For our investigations, we have discarded trees produced with less than 80% accuracy. In our experiments with Weka, we have had to add additional feature pairs to the case base data in order to help us generate more useful decision trees. For example, the feature "Expended Munitions Type" (displayed in Figure 1) was created specifically for Weka usage. Prior to the inclusion of this feature we were passing "Expended Munitions Type" instance data to Weka. Since instance data is unique, no trends were detected. The added feature allowed us to categorize the munitions data instances to yield better formed decision trees. Interestingly, these additional features improved the pie chart interpretation as well.

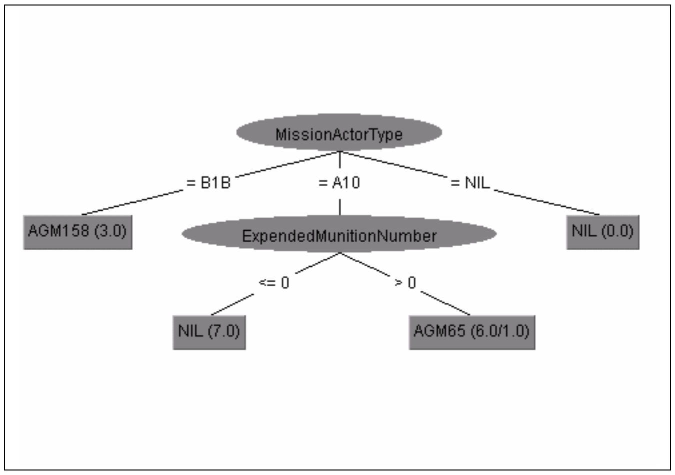


Figure 2: Decision tree describing the correlation between Expended Munitions Type and Mission Actors.

In general, our experience using Weka to generate decision trees about executed plan data was mixed. For example, Figure 2 is a decision tree generated with the C4.5 algorithm within Weka. It describes a correlation between "Expended Munitions Type" and "Mission Actor Type". The numbers in parentheses show how well C4.5 classified the data. For example, in 6 out of 7 instances, C4.5 found that the A10 used an AGM65. This particular tree is useful since it allows operators to quickly determine how weapons are being expended in the selected missions.

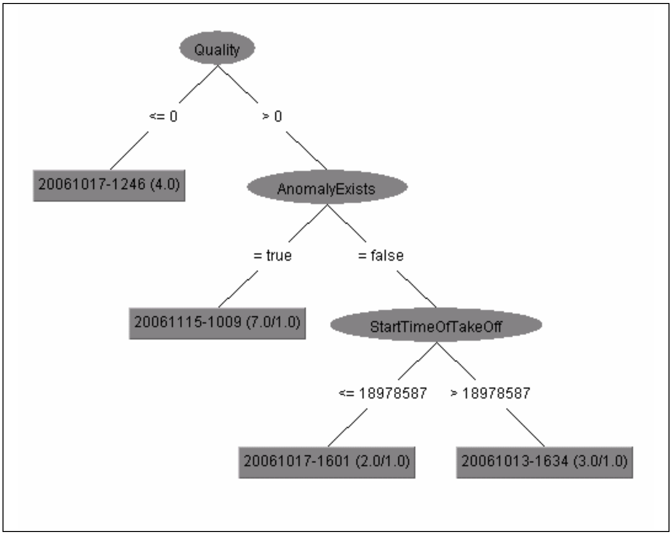


Figure 3: Decision tree associated with anomaly data.

However, we have encountered problems with the decision trees. While Weka tools such as C4.5 provide powerful ways of interpreting the data, in order to generate a decision tree; the analyst needs to select a starting node. Improper selection results in trees that make little or no sense. In addition, Weka supports multiple data types, such as dates, strings and numbers. This typing limits the construction of decision trees as a tree cannot be generated based upon a node that is numerically typed. For example, if the analyst is interested in understanding how anomalies are associated with the completion or quality of missions, they must experiment with node selection to generate a decision tree displaying the desired relations, often generating multiple useless trees in the process. Even with a properly generated tree, the observer still might have difficulty accurately understanding the meaning of the displayed data. For example, Figure 3 is an example of a tree generated with C4.5 that shows how the existence of anomalies is related to the completeness (or quality) of a mission. Quality less than 0 indicates that none of the mission activities were completed. The end node values in the figure represent trial dates. For JAGUAR, a trial with no completed mission activities indicates an aborted simulated trial. We can use trees like this to find good trial sets and in some cases to determine the cause of the anomaly. But, without sufficient, high quality, sample data and knowledge of the input data structures, analysis done with Weka tools can be misleading and ambiguous.

Experiments: Design and Observations

In order to test some of the assumptions about the usefulness of each of the JAGUAR component tools, a series of formal experiments were performed. An independent evaluator (not a JAGUAR tool developer) designed and managed the experiments. The evaluator requested that each JAGUAR component provide one or more checklists to describe how the subjects should use the tools. The subjects for a given component were also selected by the independent evaluator. Subject domain expertise ranged from retired subject matter experts to current operational Air Operations Center staff.

A total of five experiments were conducted. Table 1 describes the overall statistics for the experiments.

Table 1: Experiment Statistics

Experiment #	Experiment Duration	Number of Executed Missions	Subject Participation*
1	1 day	379	A, B, c
2	1 day	440	A, B, d
3	3 days	804	A, B, d, e, f, g, h
4	1 day	186	A, B, d
5	3 days	863	A, B, d, f, i

* Each letter represents an actual subject. When A occurs across experiments that indicates that subject A participated in the experiment. Subjects A and B participated in all of the experiments and hence became more skilled in the usage of the JAGUAR tools.

Each experiment ran for 1 – 3 days and incorporated one or more trials that correlated with JAGUAR test and evaluation exercises. A trial could vary from 30 minutes to 3 hours. Long duration trials often involved multiple subjects, each using the elements of the checklists that were applicable for that part of the trial.

While JAGUAR staff were available to support the test subjects for most of the experimental trials, in the final experiment, a selected set of operational users were expected to use the checklists to independently control each of the JAGUAR components during a simulated execution run.

For the Model Adaptor component, a set of checklists were developed to walk the subject and/or operational user through the main MA operations: publication of models, monitoring of models, analysis of executed plans, and finally changes to models. For each experiment, the subject was exposed to the full data sets that would normally be available during a simulator-based exercise. For MA this included an average of 330 historical executed missions (in an integration test setting) and an average of 800 missions (in formal test and evaluation setting). Associated anomalies and objectives, and access to the historical model repository were also provided. Each published model set included approximately 1000 model elements. To prevent damage to the official MA dataset, a virtual MA sandbox was created to support the experiments.

In the early experiments, the subjects were becoming familiar with operating the software. By the 3rd experiment, the subjects, with repeated exposure to the tool, were beginning to explore other aspects of the tool and were reporting undesirable behavior in the executed mission data that they wanted to fix. These subjects were able to use the model editors to make model changes. For example, during one of the trials, while analyzing the mission case base of executed plan data, one of the subjects noticed that an aircraft type that they expected would be used to fly an air interdiction mission was not being used in any of the executed missions to service that mission type. On inspection of the models, the subject discovered that the aircraft was defined with the proper weapon load for air interdiction, but the aircraft model was missing the capability marking which allows it to support the air interdiction mission. The subject was able to make this change by using the aircraft editors and was able to create a new model set for publication.

In another experiment trial, one of the trained subjects was able to create a mission case base using 2 days of test data, aggregating appropriate historical information together into one case base for easier analysis. This particular subject then wanted to look at the distribution of mission types across those datasets. He successfully used the Feature Analysis pie charts to obtain the data that he was interested in seeing.

In another experiment trial, a subject wanted to view how targets were serviced in sequence. Since the mission case base typically contains all of the runs during a particular time frame, if the entire case base is analyzed, then the results would yield insights about target acquisition over time. The subject indicated that this data was extremely difficult to see in the pie charts, so we enhanced the tool so that in a subsequent experiment (about a month later) we had a tabular view of the data. With this new view on the data, the subject reported that he was able to better understand the historical data and could see how targets were serviced over time by different missions. The subject also appreciated that he could save the tabular report of the data in a comma separated file format for later use.

There were varied reactions to the analysis provided through the Weka tool. The subjects were happy with the quantitative data about each of the features that was readily available through the Weka screen. During the course of our experiments, we discovered that the background of the individual subject influenced the choice of tool for analyzing the data. Every subject initially viewed the data from a high level, using the pie charts to help get a visual feel for the structure and composition of the data. We noticed that subjects who had a background that included work in the intelligence arena expressed interest in viewing the data in multiple, more complex formats, such as the Weka confusion matrices and decision trees.

Results

The experiments conducted to date indicate that the subjects found the analysis tools to be very useful for situation understanding. Many of the subjects wanted to be able to use the data as a way to provide briefings to higher authority.

The subjects generally reacted favorably to the editing tools. One of the problems encountered was the sequence of creating the models. For example, if one wants a particular weapon load on an aircraft, the weapon has to be defined before it can be associated with the aircraft. There was no information in the system to advise the subject on this fact.

The process of validating and publishing a model set was determined to be too complex, especially when only a simple model change, e.g., revision of a fuel burn rate, was performed. The subjects requested that a simpler model validation and publication process be provided.

When subjects wanted to build reports using data in the executed plans that were not available via any of the existing features, we showed them how to build new features. We discovered that the existing tools available for building new features was too difficult to use, especially those requiring complicated XPath queries.

We discovered that the benefits of machine learning tools are a function of how well the subject understands the output of machine learning algorithms. Only three out of seven of the subjects fell in this category. Since the JAGUAR Model Adaptor was developed for operators without expertise in machine learning, our observations of how the tools were being used during the experiments indicates that custom tools will need to be developed to enable future operational users to take advantage of the machine learning algorithms, without having to understand them in detail.

Finally, in a JAGUAR test and evaluation situation a plan tends to be run repeatedly and tends not to be completely flown out (or executed). This leads to misleading data and to repetitions that are not necessarily desirable or permanent. We found that many of these problems disappeared once we were in a richer testing and simulation environment (as was the case for experiment 3 and 5) where many of the planned missions were run to completion and not repeated.

Conclusions

In this paper, we have provided an overview of how executed plan data is stored, annotated and analyzed. We have described how CBR technology, data visualization tools, and certain machine learning algorithms have been used to facilitate data analysis. Experiments with subjects have uncovered usability issues with the tools and the JAGUAR terminology, as well as modeling problems ranging from the mechanics of using the tools to user expectations about what has been modeled about the domain. Requirements for enhancements to the tools were expressed throughout the experiments. As a result of the feedback from the earlier experiments, many of our tools were refined to allow the subjects to more easily identify, track, and report patterns in the executed plan data.

Our experience to date indicates that the MA tools can effectively assist domain experts, and potentially the operational users, as they work to determine the validity of models and understand how the model set affects overall plan generation. However, while tools such as the ones we have described in this paper can provide credible indications of requirements for model repair and can support the generation of consistent and valid models, the human operator has the final responsibility for implementing the repair.

The goal of the Model Adaptor is to develop, publish and refine models that allow the Plan Generator to react and adapt in the environment. This requires tools that provide a multitude of perspectives on the data. As JAGUAR transitions to an operational environment, an intelligent user interface must be provided to the users so that they do not have to labor to understand the underlying technology and can develop trust in the results generated by the automated tools.

References

[1] Ghallab, M., Nau, D., and Traverso, P. *Automated Planning: Theory & Practice*, Morgan Kaufmann, 2004.

[2] Mulvehill, A. M., Benyo, B., Cox, M., and Bostwick, R., Expectation Failure as a Basis for Agent-Based Model Diagnosis and Mixed Initiative Model Adaptation during Anomalous Plan Execution, *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, January 6 - 12, 2007, Hyderabad, India, AAAI Press, Menlo Park, CA, 2007; pp. 489-494.

[3] Christophe Giraud-Carrier; A Note on the Utility of Incremental Learning. *AI Communications*, Vol. 13 (4): 215-223, December 2000.

[4] Caragea, D., Silvescu, A., and Honavar, V., Agents That Learn from Distributed Dynamic Data Sources, *Proceedings of the ECML 2000/Agents 2000 Workshop on Learning Agents*. Barcelona, Spain, 2000.

[5] Hullermeier, El, Creedible; Case-Based Inference Using Similarity Profiles, *IEEE Transactions of Knowledge and Data Engineering*, Vol. 19, No. 6, pp. 847- 856, June 2007.

[6] Watson, Ian; *Applying Case-based Reasoning: Techniques for Enterprise Systems*. Morgan Kaufmann. 1997.

[7] Mulvehill, A. and Cox, M., Using mixed initiative to support force deployment and execution, M. T. Cox (ed.), *Proceedings of the AAAI-99 Workshop on Mixed-Initiative Intelligence*, Menlo Park, CA: AAAI Press 1999, pp. 119-123.

[8] Kolodner, J. L.; *Case-based Reasoning*, San Francisco: Morgan Kaufmann 1993.

[9] Leake, D. B. (Ed.); *Case-Based Reasoning: Experiences, Lessons, and Future Directions*. Menlo Park, CA: AAAI Press/MIT Press 1996.

[10] Schmidt, R. and Montani, S. and Bellazzi, R. and Portinale, L. and Gierl, L.; Case-Based Reasoning for Medical Knowledge-based Systems, *International Journal of Medical Informatics*, Vol. 64, Number 2-3, Elsevier Science 2001, pp. 355-367.

[11] Veloso, Manuela M.; Planning and Learning by Analogical Reasoning, *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Berlin, 1991.

[12] http://jakarta.apache.org/tomcat/

[13] http//xml.apache.org/xindice/

[14] Staken, Kimbro. *Introduction to Native XML Databases*.http://www.xml.com/pub/a/2001/10/31/nativexmldb.html, October 2001.

[15] http://jakarta.apache.org/lucene/docs/queryparsersyntax.html

[16] Hayes, C. & Cunningham, P., Shaping a CBR View with XML, *Proceedings of the Third International Conference on Case-Based Reasoning*, ICCBR'99, Seeon Monastery, Germany. *LNCS*, Vol. 1650. Althoff, K.D., Bergmann, R., Branting, L.K. (Eds.) Springer-Verlag Berlin /Heidelberg , pp.468-481, 1999.

[17] Coyle, L., Doyle, D., and Cunningham, P.; Representing Similarity for CBR in XML, *Proceedings of the Seventh European Conference on Advances in Case-Based Reasoning*, ECCBR 2004.

[18] Gardingena, D and Watson, I., A Web based CBR system for heating ventilation and air conditioning systems sales support, *Knowledge-based Systems*, Vol. 12, Issue 5-6, pp. 207-214, October 1999.

[19] Coyle, L., Hayes, C., and Cunningham, P.; Representing Cases for CBR in XML, *Proceedings of 7thUK CBR Workshop*, Peterhouse, Cambridge, UK, 2002.

[20] Hayes, C. & Cunningham, P.; Shaping a CBR View with XML, *Proceedings of the Third International Conference on Case-Based Reasoning* (ICCBR'99). Seeon Monastery, Germany. LNCS Vol. 1650. Althoff, K.D., Bergmann, R., Branting, L.K. (Eds.). Springer-Verlag, Berlin /Heidelberg, pp.468-481, 1999.

[21] http://www.w3.org/TR/XPath

[22] Holmes, G., Donkin, A., and Witten, I.H., Weka: A Machine Learning Workbench, *Proceedings of the Second Australia and New Zealand Conference on Intelligent Information Systems*, 357–361, 1994.

[23] Witten, I. H and Frank, E., *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann Publishers, 2005.