# DLMS: An Evaluation of KL-ONE in the Automobile Industry

**Nestor Rychtyckyj**
Ford Motor Company
Manufacturing Quality Business Systems
P.O. Box 1586, Room B154
Dearborn MI, 48121
usfmc53B@ibmmail.com

## Abstract

Ford Motor Company's Direct Labor Management System (DLMS) utilizes a knowledge representation scheme based on the KL-ONE family of languages to represent the world of automobile vehicle assembly knowledge. DLMS is an implemented system that has been utilized by Ford personnel since 1991 and it remains an integral part of Ford's worldwide manufacturing strategy. This paper will describe the rationale for the use of KL-ONE, provide an overview of how KL-ONE was implemented into DLMS, discuss the advantages and disadvantages of using a knowledge representation scheme such as KL-ONE in a production environment and present an evaluation of how knowledge representation and reasoning systems can be successfully utilized in industry.

## INTRODUCTION

Ford Motor Company's Direct Labor Management System (DLMS) is the knowledge-based subsystem of a complex multiphase manufacturing process planning system. Since its original deployment in 1991, DLMS has been utilized by hundreds of users throughout Ford's automobile and truck assembly plants in North America. Currently DLMS is being expanded to Ford's assembly plants around the world. The knowledge that is used to drive the manufacturing assembly process in DLMS is stored in a KL-ONE knowledge representation scheme.

This paper will discuss the long-term implications of utilizing KL-ONE in a dynamic environment such as automobile assembly planning. Issues such as knowledge base validation and verification, maintenance and adaptability to changing market conditions will also be discussed.

## DLMS OVERVIEW

The Direct Labor Management System (DLMS) is an implemented system utilized by Ford Motor Company's Vehicle Operations division to manage the use of labor on the assembly lines throughout Ford's vehicle assembly plants. DLMS was designed to improve the assembly process planning activity at Ford by achieving standardization within the vehicle process build description and to provide a tool for accurately estimating the labor time required to perform the actual vehicle assembly. In addition, DLMS provides the framework for allocating the required work among various operators at the plant and builds a foundation for automated machine translation of the process descriptions into foreign languages.

The standard process planning document known as a process sheet is the primary vehicle for conveying the assembly information from the initial process planning activity to the assembly plant. A process sheet contains the detailed instructions needed to build a portion of a vehicle. A single vehicle may require thousands of process sheets to describe its assembly. The process sheet is written by an engineer utilizing a restricted

subset of English known as SLANG (Standard LANGuage). Standard Language allows an engineer to write clear and concise assembly instructions that are machine readable.

Figure 1 shows a portion of a process sheet written in Standard Language. This process sheet is written by an engineer at the Vehicle Operations General Office; it is then sent to the DLMS system to be "validated" before it can be released to the assembly plants. Validation includes the following: checking the process sheet for errors, generating the sequence of steps that a worker at the assembly plant must perform in order to accomplish this task and calculating the length of time that this task will require. The DLMS system interprets these instructions and generates a list of detailed actions, known as allocatable elements, that are required to implement these instructions at the assembly plant level. These allocatable elements are associated with MODAPTS (MODular Arrangement of Predetermined Time Standards) codes that are used to calculate the time required to perform these actions.

MODAPTS codes are utilized as a means of measuring the body movements that are required to perform a physical action and have been accepted as a valid work measurement system around the world. [IES 88]. For example the MODAPTS code for moving a small object with only a hand is M2; utilizing the arm gives a code of M3. The MODAPTS codes are then combined to describe a entire sequence of actions. MODAPTS codes are then converted into an equivalent time, which is needed to perform that action. Figure 2 shows the output generated by the DLMS system including a description of each action with its associated MODAPTS code.

The work instructions generated by DLMS are known as allocatable elements. These allocatable elements are then used by the engineering personnel at the assembly plant to allocate the required work among the available personnel. DLMS is a powerful tool because it provides timely information about the amount of direct labor that is required to assemble each vehicle, as well as pointing out inefficiencies in the assembly process. A more complete description of the DLMS system can be found in [O'Brien et al. 89].

**Process Sheet Written in Standard Language**

TITLE: ASSEMBLE IMMERSION HEATER TO ENGINE

10 OBTAIN ENGINE BLOCK HEATER ASSEMBLY FROM STOCK
20 LOOSEN HEATER ASSEMBLY TURNSCREW USING POWER TOOL
30 APPLY GREASE TO RUBBER O-RING AND CORE OPENING
40 INSERT HEATER ASSEMBLY INTO RIGHT REAR CORE PLUG HOSE
50 ALIGN SCREW HEAD TO TOP OF HEATER
60 SEAT ASSEMBLY UNTIL ENGINE BLOCK AND ASSEMBLY ARE FLUSH
70 SECURE SCREW USING POWER TOOL TO TIGHTEN HEATER ASSEMBLY
TOOL 20 1 P AAPTCA TSEQ RT ANGLE NUTRUNNER
TOOL 30 1 C COMM TSEQ GREASE BRUSH

Figure 1.

**Resulting Work Instructions Generated by DLMS**

LOOSEN HEATER ASSEMBLY TURNSCREW USING POWER TOOL
GRASP POWER TOOL (RT ANGLE NUTRUNNER) <01M4G1>
POSITION POWER TOOL (RT ANGLE NUTRUNNER) <01M4P2>
ACTIVATE POWER TOOL (RT ANGLE NUTRUNNER) <01M1P0>
REMOVE POWER TOOL (RT ANGLE NUTRUNNER) <01M4P0>
RELEASE POWER TOOL (RT ANGLE NUTRUNNER) <01M4P0>

Figure 2.

The DLMS system consists of five main subsystems: parser, analyzer, simulator, knowledge base manager, and the error checker. The input into DLMS is a process sheet; it is initially parsed to break down the sentence into its lexical components which includes the verb, subject, modifiers, prepositional phrases and other parts of speech. Since Standard Language is a restricted subset of English, the parser has a very high rate of success in properly parsing the input from the process sheets. The parser utilizes the Augmented Transition Network (ATN) method of parsing [Charniak, et. al 87]. Any process element that is not parsed successfully will then be flagged by one of the error rules that will (hopefully) suggest to the user how to correct this element. The analyzer will then use the components of the parsed element to search the knowledge base (or taxonomy) for relevant information describing that item.

For example, the input element contained the term "HAMMER". This term will then be searched for in the taxonomy; when it is found the system will then learn all of the attributes that "HAMMER" has: (it is a Tool, its size is medium, it can be used with one hand, etc.) The system will perform this analysis on all of the components of the input element in order to select what work instructions are required.

The work instructions are then found in the taxonomy based on all of the available input and are passed on to the simulator. The simulator will use the taxonomy to generate the sequence of instructions and MODAPTS codes that will describe the input element. These work instructions will then be sent back to the user. The knowledge base manager is used to maintain the knowledge base; this maintenance may be performed by the user community or by the system developers.

All of the associated knowledge about Standard Language, tools, parts and everything else associated with the automobile assembly process is contained in the DLMS knowledge base or taxonomy. This knowledge base structure is derived from the KL-ONE family of semantic network structures and is the integral component in the success of DLMS. DLMS also contains a rulebase of over 350 rules that are used to drive the validation process and perform error-checking on the Standard Language input. DLMS was implemented in Common LISP and ART (Automating Reasoning Tool from Brightware Corporation) on the Texas Instrument Explorer platform. It is currently being ported to the Hewlett Packard UNIX platform. Figure 3 describes the current DLMS architecture.

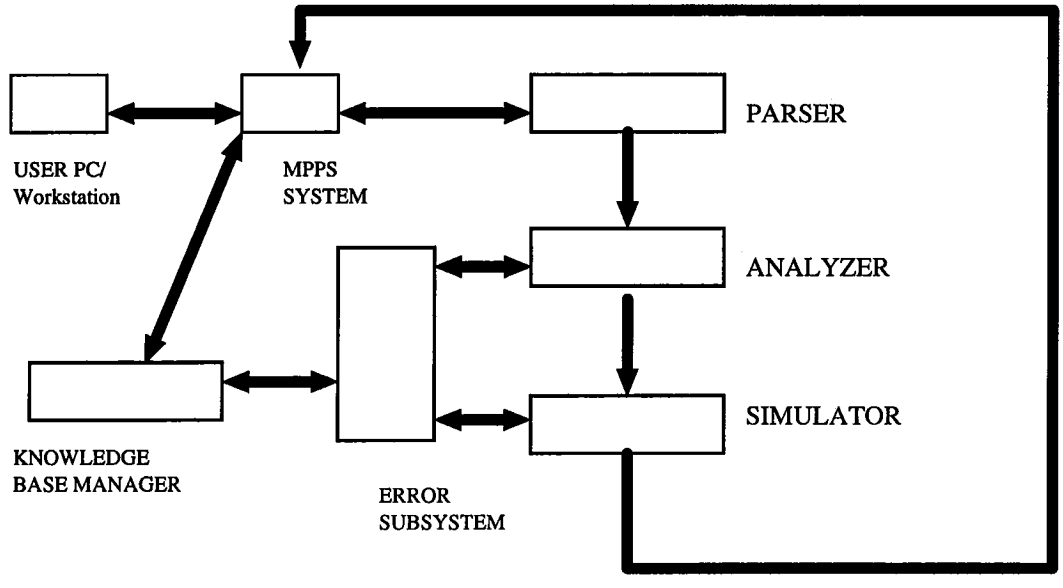


Figure 3: Current DLMS System Architecture

## KL-ONE OVERVIEW

The KL-ONE knowledge representation system [Brachman, Schmolze 85] was first developed at Bolt, Baranek and Newman in the late 1970's as an outgrowth of semantic net formalisms. KL-ONE was selected for use on the DLMS project because of its adaptability for many diverse applications as well as the power of the KL-ONE classification algorithm. KL-ONE is derived from research done on semantic networks.

The principal unit of information is the "concept". Each concept has a set of components or attributes that is true for each member of the set denoted by that concept. The main form of relation between concepts is called "subsumption". Subsumption is the property by which concept A subsumes concept B if, and only if, the set denoted by concept A includes the set denoted by concept B. The KL-ONE knowledge base as used in DLMS can be described as a network of concepts with the general concepts being closer to the root of the tree and the more specific concepts being the leaves of the tree. A concept in a KL-ONE knowledge base inherits attributes from the concepts that subsume it. The power of the KL-ONE system lies in the classification scheme. The system will place a new concept into its appropriate place in the taxonomy by utilizing the subsumption relation on the concept's attributes. A much more detailed description of the KL-ONE classification scheme can be found in [Shmolze, Lipkis 81].

## KL-ONE USAGE BACKGROUND

The requirement for a system to automate process planning in automobile assembly at Ford Motor Company was very evident since the early 1980's. Previously, process sheets were written in free-form English and then sent to the assembly plants for implementation. The quality and correctness of process sheets differed greatly based upon which engineer had written a particular sheet. There was no standardization with similar process sheets and industrial engineers at the assembly plants would be forced to implement work instructions based on differing process sheets. The process sheets could not describe the amount of labor required and the assembly plants were not able to accurately plan for labor requirements. Work usage instructions were written manually and the time required to accomplish a particular job would have to be measured manually.

These manual "stopwatch" time studies suffered from several major disadvantages. A time study consisted of an industrial engineer watching an assembly line worker doing their job and measuring how long each job would take. These measurements would vary from worker to worker, so that multiple time studies were required for each particular job. Since there may be hundreds or even thousands of jobs in an assembly plant, the time studies were very expensive and time-consuming. Time studies also have a very adverse effect on worker morale and are a source of resentment among the assembly personnel.

Since labor is a very significant portion of the cost of producing an automobile, there was a very strong incentive to develop a system that could both standardize the process sheet and create a tool for automatically generating work instructions and times from these process sheets. The first attempts to create DLMS were done utilizing standard third generation programming languages (COBOL) and existing IBM mainframe databases (IMS). The sheer complexity of the knowledge required to accurately generate reliable work instructions could not be represented in either a database or in a program. A database could easily store the amount of data required, but the relationships between the various components in the database could not be adequately represented. A program could be written that could explicitly list all of the inputs and desired outputs, but this program would quickly become obsolete and be impossible to maintain.

At this time expert systems and Artificial Intelligence were beginning to be accepted into industry, so Ford contracted Inference Corp. to develop a prototype of the DLMS system. A rule-base approach was also considered, but the complexity and future maintainability of a system containing explicit knowledge about a dynamic domain such as automobile assembly ruled this approach out. A requirement for the DLMS knowledge base included the ability to make frequent and complex changes without affecting other components of the knowledge base. This required that the objects in the taxonomy be stored in classes that were analogous to the real world of automobile assembly planning. This approach led to a semantic network representation of the automobile assembly world where classes and subclasses corresponded to their appropriate equivalents in the real world. This type of semantic network representation was very similar to the KL-ONE representation language. It was decided to model the Ford automobile manufacturing knowledge base utilizing KL-ONE in order test the feasibility of this approach. This prototype proved very successful and the basic KL-ONE model proved to be

both robust and flexible as the knowledge base evolved over the years. Changes were made for processing and memory efficiency (i.e. the use of a hash table to store the list of concepts), but the KL-ONE logical design has been successful in terms of our problem domain.

## DLMS KNOWLEDGE BASE STRUCTURE

As mentioned previously, the DLMS taxonomy or knowledge base contains all of the relevant information that describes the vehicle assembly process at Ford Motor Company. This includes all of the lexical classes included in Standard Language such as verbs, nouns, prepositions, conjunctions and other parts of speech, various tools and parts utilized at the assembly plants, and descriptions of operations that are performed to build the vehicle. Currently the DLMS taxonomy contains over 9000 such concepts.

The organization of the knowledge base is based on the KL-ONE model. The root of the semantic network is a concept known as THING which encompasses everything within the DLMS world. The children of the root concept describe various major classes of knowledge and include such things as TOOLS, PARTS and OPERATIONS. Each concept contains attributes or slots that describe that object. The values of these attributes are inherited from the concept's parents. Ranges of valid values can be given for any particular attribute. Any attempt to put an invalid value in that attribute will trigger an error. All of the information dealing with the organization and structure of the taxonomy is also contained in the taxonomy itself. There are four types of links that describe the relationship between any two concepts: subsumes, specializes, immediately-subsumes and immediately-specializes.

The subsumption relation describes a link between a parent concept and all of its children, including descendants of its children. The "immediately-subsumes" relation describes only the concepts that are direct children of the parent concept. The "specializes" and "immediately specializes" relations are inverses of the subsumption relation. A concept "immediately specializes" its direct parent concepts and "specializes" all of the concepts that are ancestors of its parents. These relationships are stored as attributes of any given concept and can be utilized as a tool to trace any concept through the entire taxonomy. Figure 4 shows how the DLMS taxonomy is organized.

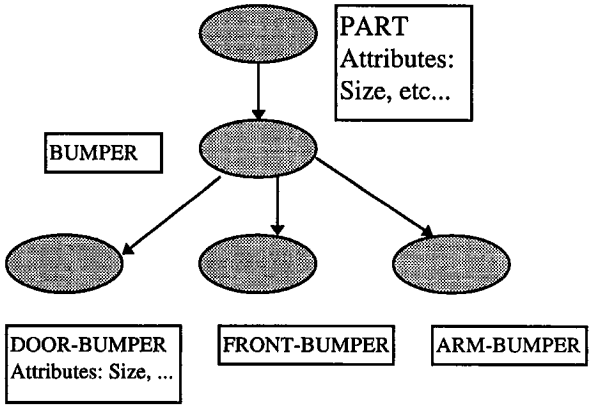


FIGURE 4: A Sample from the DLMS Taxonomy

## KNOWLEDGE BASE MAINTENANCE

The DLMS Knowledge Base is maintained through the use of two different tools: the Knowledge Base Manager (KBM) and the Knowledge Base Update facility (KBU). The Knowledge Base Manager is a graphical tool that is used by the system developers to make important changes to the knowledge base that will affect the actual output generated by the system. Since this output will have a major impact on the assembly process any such change must be approved by a committee representing all of the interested parties. All changes made to the knowledge base are logged by the system to keep a record of the system's modification history.

The Knowledge Base Update (KBU) facility is used by system users to make minor modifications to the knowledge base. A minor modification is a change that will not impact the output produced by the system. Examples of minor modifications include the addition of new words into the taxonomy. The KBU facility allows users to incorporate these changes directly into the taxonomy without any kind of system developer intervention. All changes made through the KBU facility are also logged for future reference.

## KNOWLEDGE BASE VALIDATION AND VERIFICATION

The DLMS system has been in production for over five years. In the highly competitive automobile industry, five years is a relatively long time as the cycle for bringing new models to market is steadily decreasing. Therefore, the DLMS knowledge base is being constantly updated and modified. It was necessary to develop tools that would prevent errors from being introduced into the DLMS knowledge base when making modifications to it.

One such tool is the use of an automated regression testing tool that runs a suite of test cases against the knowledge base. The results of the test cases are compared against a baseline file and all discrepancies are then flagged. These discrepancies are then examined manually in order to find out if this change is correct. The suite of test cases is constantly being updated to incorporate as many different test scenarios as possible.

Another approach that we are taking toward knowledge base validation is the use of automated tools that analyze the taxonomy. This analysis is based on knowledge of the problem domain as well as information about the structure of the DLMS taxonomy. For example, one requirement of the DLMS system is that all physical objects must have an attribute that describes their size. A scan of the taxonomy can then search for concepts in the class of objects that do not have a valid size. All concepts that may have an error are then flagged for future investigation by the systems organization. These types of utilities are very useful in finding errors which may contribute to decreased system accuracy. Our future plans for DLMS include moving a large portion of the knowledge base maintenance activity to the user community, which will not have a technical background. To accomplish this we are also developing various automated knowledge base verification and validation routines that will point out errors as they are being introduced into the knowledge base.

This utility is based on the concept of a "knowledge base metric", that is roughly analogous to software metrics that are used as a quantitative measure of software quality. Software metrics are computed by measuring various attributes of code in order to develop a value that can be used to predict the complexity and maintainability of this program [Khoshgoftaar, Oman 94]. Our goal is to develop a knowledge base metric that will be computed automatically when the knowledge base is modified. This number is computed by counting the number of discrepancies that have been found by applying the specific DLMS utilities to the knowledge base and using that number as a baseline. After each modification this number is recomputed and then compared to the baseline. If the new metric is greater than the baseline this change will be flagged as it may have introduced additional errors in the knowledge base. This metric will provide us with an additional tool of verifying the knowledge base correctness.

## CLASSIFICATION IN DLMS

The DLMS system utilizes a classification algorithm to create concepts and place them into their appropriate position in the taxonomy. The classifier utilizes various attributes of the concept in order to place it into its correct position. These "classifiable" attributes are slot values that play a major role in determining where this concept belongs. For example, the attribute "size" is very important in classification, while the "output format" slot has little value in classification. This classification is performed by finding the appropriate subsumers, linking the concept in and then locating all the concepts that should be subsumed by the new concept. The system narrows this search procedure considerably by selecting the appropriate node in the concept to begin the classification process. The concept which is to be classified is placed at the starting node; the system then tries to push the new concept node as far down the tree as possible. The classifiable attributes are used as a objective measure to determine if the concept is in its proper place. Within DLMS this classification algorithm is applied to all of the instances of the input string that describe the process element. In a simple element this may include a verb, an object and an associated tool. When the classifier is complete, each of the above instances will inherit necessary values from the knowledge base in order to build the appropriate operation to describe required actions. For a more complete explanation of the DLMS classifier see [Rychtyckyj 94]. Figure 5 illustrates a diagram of a simple classification procedure.

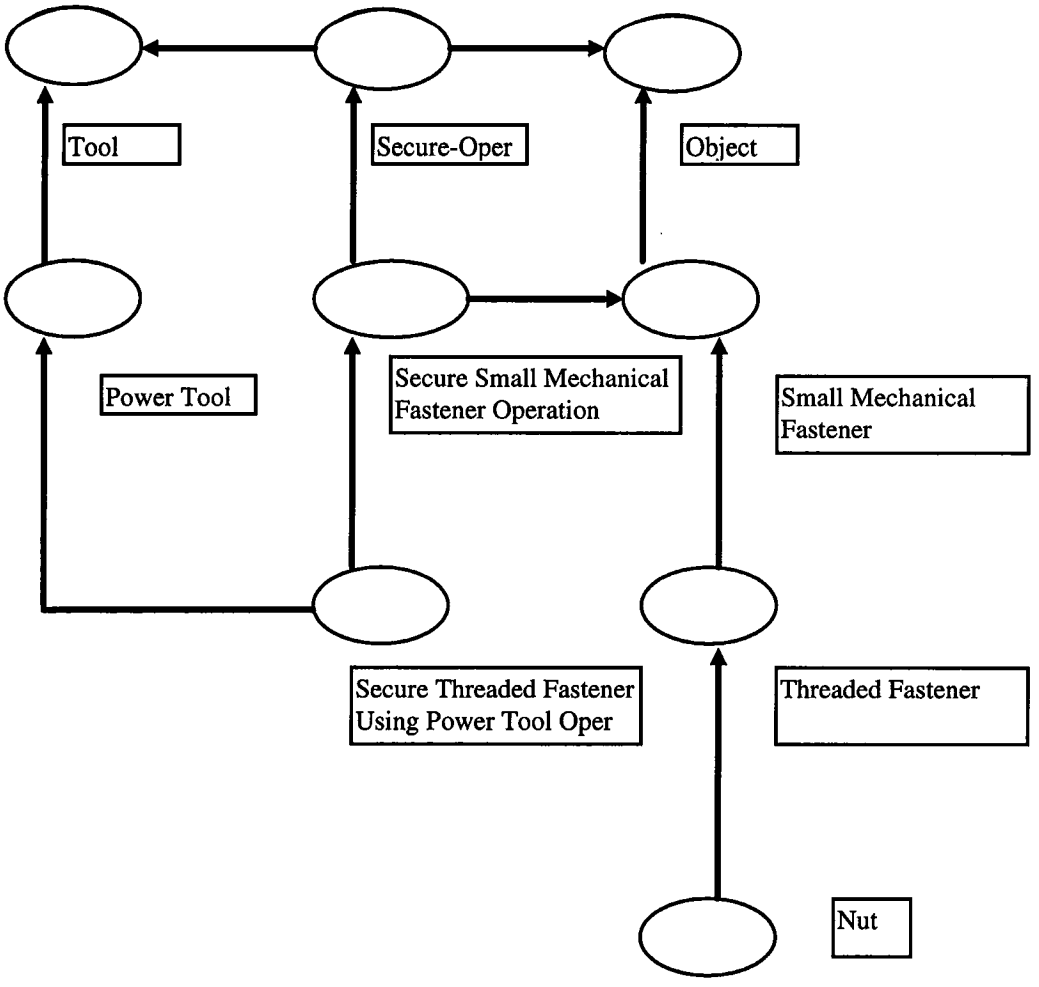


Figure 5. Taxonomy Classification Structure for Secure Operation

This figure describes the classification procedure for an operation of type "Secure Nut Using Power Tool". The concept "Nut" belongs to the class of "Threaded Fasteners". The analyzer subsystem also finds the class of Power Tools and then attempts to find which operation most closely matches the derived objects. In this case, the classifier has identified the power tool, the "Secure" verb and the object "nut". It will now try to find the "Secure Operation that most closely matches these parameters. In this case the Secure Threaded Fastener Using A Power Tool is the closest match and the classifier will then use this operation to start building its output solution.

## EVALUATION OF KL-ONE IN DLMS

This section will describe the advantages and disadvantages of the KL-ONE representation scheme as it has been utilized within DLMS. Along with Classic [Brachman, et al. 91] DLMS has been one of the few KL-ONE-based systems that has been utilized in production for a considerable amount of time. KL-ONE was selected as the appropriate tool for this project because of the requirement for a flexible and powerful knowledge management tool. The knowledge base required to store information about the vehicle assembly process is both complex and dynamic. Over the last two years Ford Motor Company has replaced or changed more than half of its product line in North America, including its two top selling vehicles: the Taurus and F-Series pickup truck. Such a rate of

change requires that the DLMS system be able to incorporate continuous updates to its knowledge base. These changes also included major modifications to the structure of the taxonomy, as well as those related to maintaining the actual automobile assembly knowledge itself.

The most complex part of the DLMS taxonomy is the class of concept known as an OPERATION. An operation is a concept that describes all of the individual actions that must be done at the assembly plant. A simple operation, such as "Hammer Object" contain the following slot values: Tool (tool required), Modapts (Modapts codes that describe this operation and an Actions slot. In this case the Actions slot will be empty, because no additional actions are required. However, the vast majority of operations require a multitude of actions, so this value will contain a list of the other required actions. The system will then process each of these actions in turn, and it is quite likely that the secondary actions may also contain subordinate actions. Therefore, the output generated from a single operation may be accessed from many different places in the taxonomy. This flexibility makes it possible to model the real world in DLMS, but it also makes it very difficult to trace where an error may have occurred. This is very analogous to tracing the execution of a large rulebase to find a single rule that has incorrectly fired. It is also possible to introduce cycles into the taxonomy; an indirect cycle of the form A -> B -> C -> A will usually not be noticed until an actual run goes into an infinite loop. This tradeoff between flexibility and ease of maintenance is common as the complexity of the world we are trying to model increases. As described previously in the section on knowledge base validation and verification, our approach is to build more sophisticated tools that incorporate intelligence about both KL-ONE and the automobile assembly process in order to improve the overall maintenance.

Another problem we have encountered is the requirement from various user organizations to produce documents describing the various taxonomy outputs. The complexity of the taxonomy makes any general reports extremely long and difficult to follow. Our approach has been to educate the user community about the structure of the DLMS taxonomy in contrast to a spreadsheet or a database, which are well understood by most people. Another issue that we must be careful with is the fact that taxonomy modifications may change the output from a previous run. The user community is not very receptive when a process sheet that worked last year is now flagged as invalid, because we have improved the error-checking capability of the system.

To forestall most of these problems we disseminate any such changes to the user community when they are implemented in order to prevent any future misunderstandings. Overall, based on our years of experience with the KL-ONE representation system, we are quite satisfied that KL-ONE has proved to be a very capable tool and has contributed greatly to the success of the DLMS system.

## CONCLUSIONS

In this paper we have tried to describe our experience at Ford Motor Company with the KL-ONE representation system. The DLMS system is a production system that is utilized by hundreds of users throughout the company. Therefore, the reliability of the system is of the highest priority and any new technology must be robust enough to operate successfully in this environment. We have discussed the ongoing problems of knowledge base maintenance and knowledge base validation and verification. Our solutions to both problems include automated tools and manual intervention by the system developers as required. Our goal includes the development of more sophisticated tools that would eventually pass all knowledge base maintenance activity to the user community. We have also discussed the ramifications of using a complex knowledge representation system in the realm of a corporate business environment and the steps we have taken to improve this situation. Our extensive use of KL-ONE within DLMS has convinced us that the KL-ONE representation system is an excellent tool for modeling the complex world of vehicle assembly planning and we are planning to expand our usage of KL-ONE into other problem domains in the automobile industry.

**Acknowledgments**

The Direct Labor Management System was a product of the work of many people that have contributed to the success of the system over the years; therefore I would like to give credit to the following people: John O'Brien, Tom Kaszamarek, Scott Hatfield, Wayne Johnson, Richard Woodhead, Alan Turski, Henry Brice, Tom Vitale, Jay Zaback and Rick Keller.

I would also like to thank Dr. Robert Reynolds of Wayne State University for all of his valuable insights and suggestions. I am also indebted to Alan Turski, whose work on the DLMS system was also instrumental for its success. Thanks are also due to the anonymous referees, whose comments and suggestions I have tried to incorporate into this paper.

**References**

Baade, F., (1990), "Terminological Cycles in KL-ONE-based Knowledge Representation Languages" in Proceedings of the Eighth National Conference on Artificial Intelligence, vol. 2., pp. 621-626.

Brachman, R., McGuiness, D., Patel-Schneider, P., Resnick, L., Borgida, A., (1991) "Living With Classic: When and How to Use a KL-ONE-Like Language" in Principles of Semantic Networks, ed.
J. Sowa, pp. 401-456, Morgan Kaufmann Publishers.

Brachman, R., Schmolze, J.(1985), "An Overview of the KL-ONE Knowledge Representation System", Cognitive Science 9(2), pp. 171-216.

Charniak, E., Riesbeck, C., McDermott, D., Meehan, J., (1987), Artificial Intelligence Programming, pp. 304-336, Lawrence Erlbaum Publishers.

Industrial Engineering Services (1988), Modapts Study Notes for Certified Practitioner Training.

Khoshgoftaar, T., Oman, P., (1994), "Metrics in Software", IEEE Computer, Vol. 27, No. 9, pp. 13-15.

Lipkis, T., "A KL-ONE Classifier", (1981), Consul Note #5, USC/Information Sciences Institute.

O'Brien, J., Brice, H., Hatfield, S., Johnson, W., Woodhead, R. (1989), "The Ford Motor Company Direct Labor Management System: in Innovative Applications of Artificial Intelligence, ed. Schorr & Rappaport, pp. 331-346, MIT Press.

Rychtyckyj, N. (1994), "Classification in DLMS Utilizing a KL-ONE Representation Language" in Proceedings of the Sixth International Conference on Tools With Artificial Intelligence, pg. 339-345, IEEE Computer Society Press.

Schmolze,J., Lipkis, T.,(1983), "Classification in the KL-ONE Knowledge Representation System" in Proceedings of the Eighth International Joint Conference on Artificial Intelligence, pp. 330-332, Morgan Kaufmann Publishers.

Woods, W., Schmolze, J.,(1992), "The KL-ONE Family" in Computers & Mathematics with Applications, Vol. 23, No. 2-5, pp. 133-177.