The Ford Motor Company Direct Labor Management System

John O'Brien, Henry Brice, Scott Hatfield
Ford Motor Company
17000 Oakwood Blvd.
Allen Park, MI 48101

Wayne P. Johnson, Richard Woodhead
Inference Corporation
5300 West Century Blvd.
Los Angeles, CA 90045

Abstract

The Ford Motor Company Direct Labor Management System (DLMS) is one major subsystem in a multi-phase Manufacturing Process Planning System (MPPS) intended to assist in all aspects of the manufacturing planning process. DLMS plays a central role by automatically generating detailed plant-floor assembly instructions from high level structured English descriptions and by providing consistent and accurate estimates of the associated labor times. Much of DLMS design was inspired by CONSUL [Mark 81], a system which uses classification-based reasoning in the process of interpreting natural language requests for interactive computing services.

The kernel of the DLMS system is a large knowledge base of automotive assembly expertise constructed and maintained by the end-user engineering community. The central Knowledge Base Management System (KBMS) is a classification-based terminological reasoning system implemented on the NIKL model. The KBMS is tightly coupled with a natural language component which interprets the surface language process description and maps it into a set of standard atomic assembly instructions within the knowledge base. The resulting instruction set forms the basis of a knowledge based simulation which generates any auxiliary work elements necessitated by this process script.

Introduction

The Direct Labor Management System (DLMS) is one major subsystem of a multi-phase Manufacturing Process Planning System (MPPS) designed to assist production and planning personnel in all aspects of the manufacturing process; it will eventually support several thousand users. The system is extremely ambitious in scope and has profound implications for the process planning activity at Ford Motor Company. DLMS is designed to provide the foundation for several intelligent systems aimed at improving the assembly process at Ford and has targeted several key areas in the planning process. The major objectives are:

1. To achieve standardization in process description and to improve the clarity of process sheets. The process sheet is a critical document and is the primary vehicle for conveying assembly information from the initial process planning stage through to assembly at the plant level. Process sheets and their derivatives should be an effective means of communicating information at all stages of the assembly process.

2. To support the creation of work allocation sheets by automatically generating detailed plant-floor assembly instructions from the more abstract process sheet text. The allocation sheet contains the detailed assembly instructions assigned to an individual assembly worker (several allocation sheets are typically derived from one process sheet by elaborating and reordering operations specified on the process sheet).

3. To automatically provide consistent and accurate estimates of product and non-product labor times involved in the assembly process. (Product times represent effort directly associated with product assembly. Non-product times represent any indirect effort required to carry out the direct operation.) Removing this essentially clerical task frees the industrial engineer to use his expertise effectively in analyzing the assembly processes. Consistent process descriptions help in this respect by highlighting any process inefficiencies.

4. To provide a foundation for generic processing. Process sheets are currently written at a fine level of granularity. The objective of generic processing is to provide process descriptions at a higher level and thus to promote standardization in the assembly process across vehicle types. Process sheets can now be written at any level of abstraction. Instructions can be written in terms of the micro motions made by the assembly worker (e.g., "grasp screw," "position screw," etc.) or alternatively as macro descriptions applied to complete vehicle subsystems (e.g., "install brake system").

5. To provide a foundation for machine translation. Production of automobiles is increasingly an international effort and the overhead involved in translating process instructions into different languages is considerable. The standard language is designed to facilitate this and the taxonomy is structured such that it will serve as an interlingual form for the target languages.

System Architecture

The first stage in system development was to identify a grammar for describing assembly processes. This grammar is outlined in the next section. The software system implemented to interpret process descriptions is described in subsequent sections. An illustration of the software architecture is shown in Figure 1.

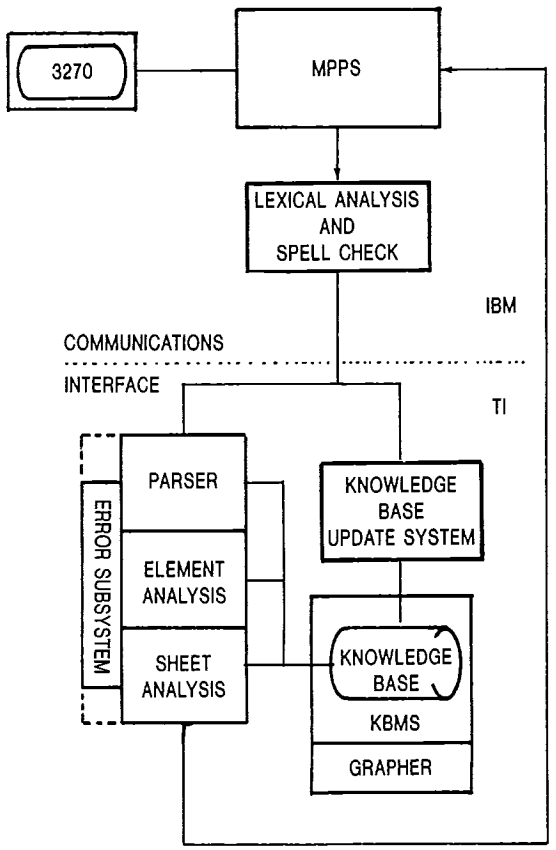

Figure 1: The DLMS Software Architecture

The Process Description Language

A clear prerequisite for system development was the identification of a standard format for writing process descriptions: these were originally written in free-form English and uninterpretable by computer.

Process sheets in the re-engineered system are described using a case grammar developed specifically to meet the requirements of automotive assembly. It provides for the expression of imperative English assembly instructions at any level of detail. The fact that the language had to be accessible to potentially thousands of engineers with different educational levels meant that the necessary coverage had to be achieved without significant loss of expressivity or "naturalness." Approximately three hundred engineers have been trained to use the language to date without experiencing any significant difficulties.

A typical process sheet is illustrated in Figure 2.

```
SHEET EJ540000000 CONNECT HEATER HOSES TO A/C EVAPORATOR
ASSY.

 10  OBTAIN LUBRICANT FROM STORED POSITION
 20  SPRAY LUBRICANT ONTO A/C EVAPORATOR ASSEMBLY NIPPLES
 30  OBTAIN 2 CLAMPS FROM STOCK
 40  POSITION 1 CLAMP ONTO OUTLET HOSE
 50  SEAT HOSE TO A/C ASSY.
 60  POSITION EXISTING CLAMP ON HOSE BETWEEN A/C
     EVAPORATOR ASSY. NIPPLE AND ASSY.
 70  SECURE OUTLET HOSE TO HEATER NIPPLE WITH 1 HOSE CLAMP
 80  POSITION 1 CLAMP ONTO INLET HOSE
 90  SEAT HOSE TO A/C EVAPORATOR ASSY.
100  POSITION EXISTING CLAMP ON INLET HOSE BETWEEN A/C
     EVAPORATOR ASSY. NIPPLE AND ASSY.
110  SECURE INLET HOSE TO HEATER NIPPLE WITH 1 HOSE CLAMP
     USING A NUTRUNNER

TOOL (70 110) 1 P AAPTCA TSEQ RT ANGLE NUTRUNNER TORQUE
SHUT OFF
TOOL (70 110) 1 S 2H12E10H16B56 TSEQ 3/8 HEXSRT BDY1/2
ODLONG LNGTH.

PROCESS-PREFIX EJ
REPEAT-OPPOSITE-SIDE N
PLANT AGO
VEHICLE CQA
```

Figure 2: A Typical Process Sheet

Process data falls into 2 categories. Process descriptions are specified in the main body of the text and in part and tools areas of the sheet. This is supplemented by contextual data: the sheet title, vehicle and plant information for example.

Certain word categories in the language possess very specific semantics: these were defined by the engineering community. Verbs in the language are associated with specific assembly actions and are modified by significant adverbs where appropriate. So, for instance, the fragments "inspect," "visually inspect" and "manually verify" all have different interpretations. Important cases in the language are denoted by specific tokens. Sentence 110 from the example sheet is illustrated with its constituent cases in Figure 3.

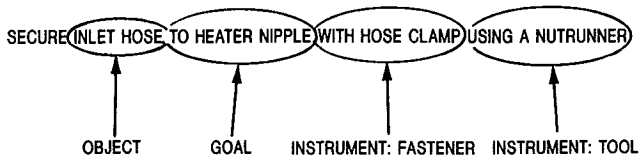

Figure 3: Example Sentence and its Constituent Cases

Other areas of the process sheet containing part, tool, vehicle and other related information are parsed to provide extra detail and also to provide context for the sheet.

The Knowledge Base and Associated Management System

At the center of the DLMS system is a large taxonomy of automotive assembly expertise. This maintains descriptions of any concepts required to interpret the surface language and to generate the implied atomic work instructions. Concepts in the taxonomy include parts, process equipment, standard operations and geometric workstation models. A portion of the current taxonomy is illustrated in Figure 4.

The DLMS knowledge representation system is implemented on the NIKL [Moser 83] [Kaczmarek, Bates and Robins 86] model. All concepts in the environment are described in terms of a Frame-based Description Language (FDL): this is a variation of the FL$^-$language described by Brachman and Levesque [Brachman and Levesque 84]. It meets all the identified requirements for terminological reasoning in this application domain and avoids the computational limitations implied by FL$^-$. (See [Brachman and Levesque 84] [Nebel 88] for formal treatments of issues involved in selecting a language to support terminological reasoning).

Concepts in the taxonomy fall into two basic categories. Primitive concepts (corresponding to NATURAL KINDS) are concepts whose structure contains necessary, but not sufficient, criteria for determining subsumption. This means the system will never automatically infer that a supplied concept is subsumed by a primitive concept, it must be specifically instructed to make the connection. The "operation" concept in Figure 4 is an example of a primitive concept. Non-primitive concepts are assumed to be fully defined by their descriptions which contain necessary and sufficient criteria for determining subsumption.

Non-primitive concepts are defined by their roles which describe properties of a concept by relating it to another concept. Roles correspond approximately to slots in a frame and form a sub-taxonomy in their own right. A role is attached to a domain concept (the most general concept for which the role has meaning) and represents a set of fillers for each instance of that concept. Role restrictions are used to constrain the set of values which a filler can take. The "Secure threaded fastener using power tool" concept from Figure 4 is a

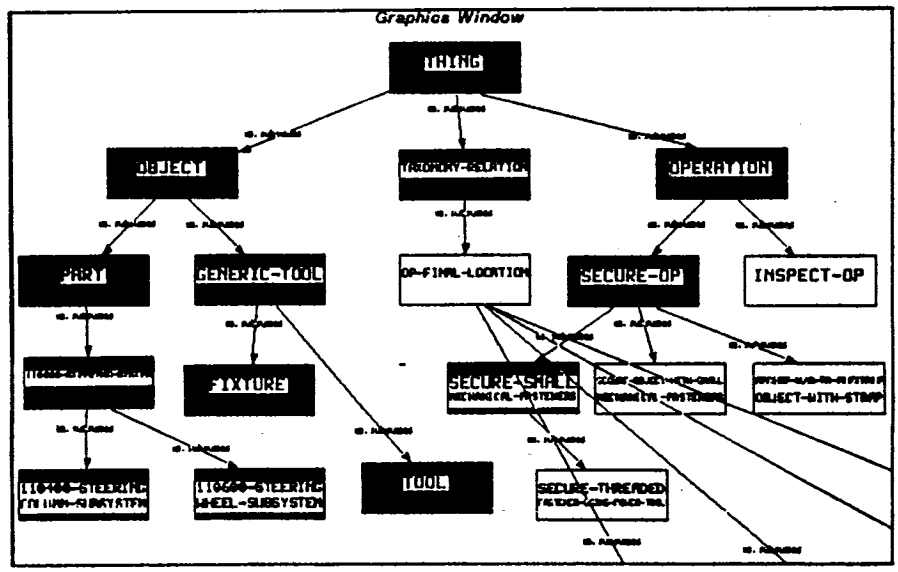

Figure 4: Part of the DLMS Taxonomy

non-primitive concept. A partial description of the concept and some of its superconcepts are shown in Figure 5 (the notation follows the example of [Moser 83]).

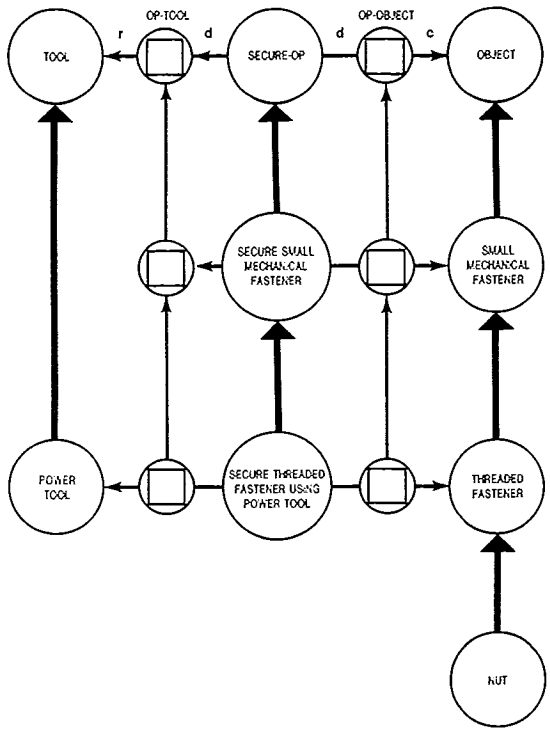

Figure 5: Taxonomy Structure for Secure Operations

Concepts can also be attributed with non-definitional information. One use of this is to associate a mini process script with each standard operation.

An important result of the precisely specified semantics of the concept description language is that a formal definition of subsumption and an associated classification procedure can be defined. The process of classification involves determining the most specific subsumers and most general subsumees of the current concept with the taxonomy.

The notion of subsumption is intuitively clear [Nebel 88]:

> "Concept 1 subsumes Concept 2
> <=> all objects which are a Concept 2 are also a Concept 1"

The test for subsumption used in this application is as follows.

Concept 1 subsumes Concept 2 iff:

- All primitive concepts that subsume Concept 1 also subsume Concept 2.
- For each roleset of Concept 1 and the corresponding roleset of Concept 2, the value description of Concept 1's roleset subsumes that of the value description of Concept 2's roleset.

More detailed presentations of classification in the KL-ONE family of systems are given in [Schmolze and Lipkis 83] and [Lipkis 81].

The DLMS classifier is incremental in nature. As concepts are added or modified, only those inferences which are required to position the current concept and to reclassify any related others are made.

Two interfaces to the KBMS are provided. A procedural interface has been developed to support reasoning within the process sheet and end-user Knowledge Base Update (KBU) subsystems. An interactive graphical interface inspired by the ISI grapher [Robins 86] supports taxonomy maintenance for system development purposes.

The number of concepts and the level of detail at which they must be represented is determined by the requirements of any standard operations and the need to discriminate between them. The set of defined operations determines the "granularity" of the resulting plan and hence the resolution of any associated time estimates. The required granularity or resolution can only be determined with respect to the processing requirements of any downstream client application. The resolution of the system increases over time as significant derivatives of existing operations are identified. This leads to a definition of operational redundancy: a concept is redundant (for the purpose of creating work instructions) if there is no associated standard operation which refers uniquely to it. To clarify this, consider the taxonomy fragment shown in Figure 5. The concept "nut" is clearly redundant as the only operation of significance refers to the concept "threaded fastener" which subsumes the concept "nut."

The Parsing Subsystem

The parser produces a structure for grammatically correct sentences and provides information about the state of the parse to the error subsystem when parsing fails. It was designed primarily to support efficient processing of syntactically correct sentences and to provide for flexible extension of the grammar. As the system is transaction oriented and no direct user interaction is possible, any errors are processed as a group at a later stage in the analysis.

The parser is implemented as an Augmented Transition Network (ATN) [Winograd 83] [Allen 87]. To improve the performance of the ATN, the implementation includes a "cut" operator which eliminates backtracking in cases where traversing a specific arc clearly excludes all other alternatives. In addition, to eliminate the cost of undoing actions during backtracking the parser defers most of its actions to a second phase which it performs once a successful parse has been achieved. This second phase generates a predicate-based representation of the parse tree used in the element analysis subsystem described later.

The parser has an associated rule-based error processing component tailored specifically to match the standard grammar. It both detects errors and suggests appropriate corrections to the user. No attempt was made to implement a general mechanism, for instance of the type proposed by Weischedel and Sonhdeimer [Weischedel & Sonhdeimer 83]. Error system requirements were derived by exhaustively analyzing the types of errors made by process engineers during their initial attempts to use the process description language.

Data is passed to the error subsystem by the parser which maintains a set of states representing the progress it has made in interpreting a given sentence. This set contains those partial parses having the greatest number of edges. If the parser is blocked, it reconstructs the partial parse tree for each of these states and applies rules to select one as the most likely intended interpretation.

The error system proceeds by using information from the lexicon to identify case markers in the unparsed portion of the sentence. These case markers are then used to hypothesize any constituent phrases intended by the user from clues provided by the marker itself, the partial parse and the position of the phrase relative to other datum tokens (any verb or direct object for instance). These hypotheses are tested using the appropriate transition sub network. When a complete hypothesis for the intended sentence structure has been determined and validated it is translated into an error message and an appropriate correction strategy for return to the user.

The Element Analysis Subsystem

The objective of element analysis is to generate a set of atomic work elements representing the direct labor content implied by the sheet. This can be interpreted as a process script for performing the assembly task. Direct labor is defined to encompass all those actions which contribute directly to the product assembly process. Indirect labor constitutes all the activities which are required to effect the associated direct actions.

The process sheet illustrated in Figure 2 is used to motivate the discussion.

A preliminary task involves identifying any significant contextual data (vehicle, plant and functional subsystem of the vehicle for example) from the appropriate regions on the sheet and establishing plausible defaults.

The system continues by interpreting the parse trees supplied by the parsing subsystem. The first phase resolves any weak or implied references in the sheet (ellipsis, intra and inter-sentential, anaphora, etc.).

The next phase is to map individual statements on the sheet into the corresponding operation concept within the taxonomy. Consider statement 110 from Figure 2. Objects implied by the constituent noun phrases in the statement are identified:

— Appropriate prepositional phrases are transformed into a normal noun-noun modifier form.

— Noun-noun modifiers are analyzed and individual concepts identified using a precompiled dictionary structure.

— Concept descriptions are created and classified to determine the corresponding persistent taxonomic concept.

Significant cases in the statement are identified from the appropriate tokens and are mapped onto their corresponding roles. Any tools associated with the statement are identified as necessary instrumental cases.

An operation description is constructed and classified to identify its counterpart standard operation and the assembly script stored with the definition of this operation is retrieved. The operation description created for statement 110 is illustrated in Figure 6.

After any necessary variable bindings have been established for the script its constituent activities are in turn classified to identify their standard counterparts. This process continues until a set of terminal scripts has been identified. This set represents the direct labor content of the statement. This set is then supplemented by adding any tool handling information required to undertake the operation.

The set of direct allocatable elements generated by statement 110 are illustrated in Figure 7. Redundant data is embedded in the output to maintain contextual information when the elements are split up into operator work allocations. The codes in parentheses represent the direct labor times associated with each element.

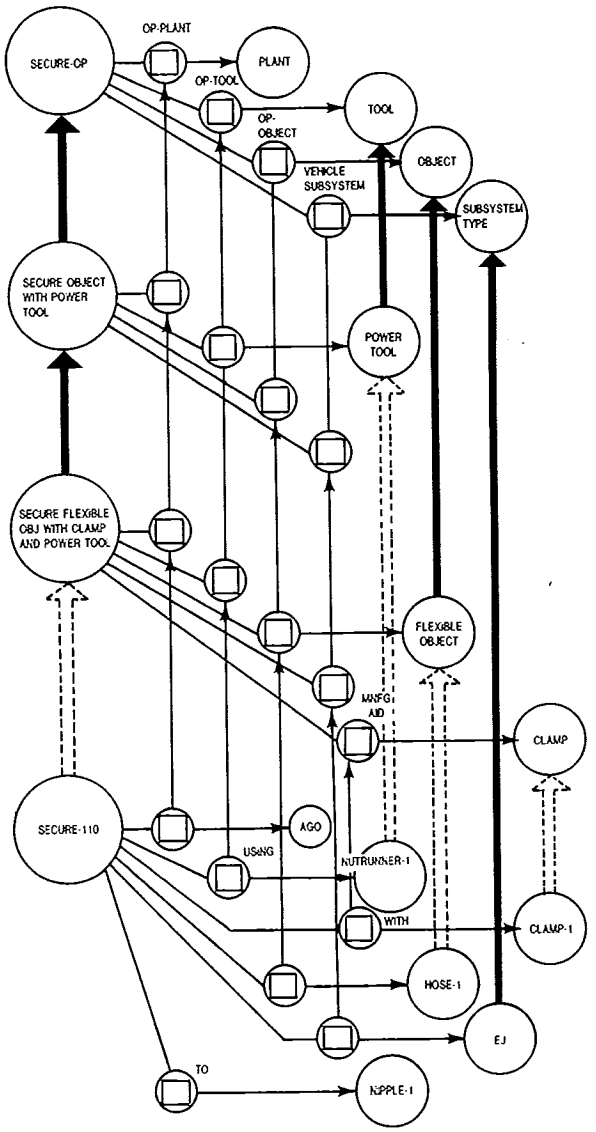

Figure 6: Operation Structure for Sentence 110:

"SECURE INLET HOSE TO HEATER NIPPLE WITH 1 HOSE CLAMP USING A NUTRUNNER"

GRASP POWER TOOL (RT ANGLE NUTRUNNER TORQUE SHUT OFF) (01M4G1)
POSITION POWER TOOL (RT ANGLE NUTRUNNER TORQUE SHUT OFF) TO HOSE CLAMP
SECURE INLET HOSE (01M4P2) TO HEATER NIPPLE USING POWER TOOL (RT ANGLE NUTRUNNER TORQUE SHUT OFF) WITH HOSE CLAMP (01M2P5M4P5M1P0)
RELEASE POWER-TOOL (RT ANGLE NUTRUNNER TORQUE SHUT OFF) (01M4P0)

Figure 7: Direct Elements for the Sentence:

The Assembly Process Simulation Subsystem

The assembly simulator elaborates the set of direct elements by generating any additional indirect (non-product) work elements required to implement the current plan. These are essentially the motions the operator needs to make to move around in the workstation.

The first stage in this process is to identify the standard workstation configuration in which the assembly operation is assumed to take place from a taxonomy of standard configurations. This is done by creating a workstation description from contextual information in the sheet and classifying it. Each work element is then processed in sequence to identify any datum locations referred to in the text. The operator is assumed to relocate between datum locations in carrying out the assembly operation. The absence of a datum location for a given element is taken to mean that the operator remains in situ for that element. Datum locations are once again identified by creating concept descriptions and classifying them. This allows a location to be associated with a concept at any level of detail.

The complete set of elements for sentence 110 of Figure 1 is illustrated in Figure 8.

WALK TO POWER TOOL
GRASP POWER TOOL (RT ANGLE NUTRUNNER TORQUE SHUT OFF) (01M4G1)
WALK TO HEATER
POSITION POWER TOOL (RT ANGLE NUTRUNNER TORQUE SHUT OFF) TO HOSE CLAMP
SECURE INLET HOSE TO HEATER NIPPLE USING POWER TOOL (RT ANGLE NUTRUNNER TORQUE SHUT OFF) WITH HOSE CLAMP (01M2P5M4P5M1P0)
RELEASE POWER TOOL (RT ANGLE NUTRUNNER TORQUE SHUT OFF) (01M4P0)

Figure 8: The Complete Set of Elements for Sentence 110:

Updating the Knowledge Base

The Knowledge Base Update (KBU) tool is designed to be accessible to any of the process or industrial engineers who use the system. The design goal was to allow the user unrestricted access to maintain that part of the taxonomy which directly relates to his area of assembly expertise. This was necessitated by the fact that there is no single source of assembly expertise: in fact there are several hundred experts in the assembly community who all contribute parochial expertise to the final objective.

The KBU is divided into 2 basic areas of functionality. The process engineering community maintains a common dictionary and adds descriptions of parts, tools

and other assembly equipment to the taxonomy. The industrial engineering groups are responsible for maintaining that part of the taxonomy which contains standard operations. A query facility is universally available to facilitate knowledge base interrogation.

The Deployment Architecture

The DLMS system is installed within the existing Ford Motor Company systems network and is accessible worldwide. An overview of the "Back end Processor" architecture is shown in Figure 9.

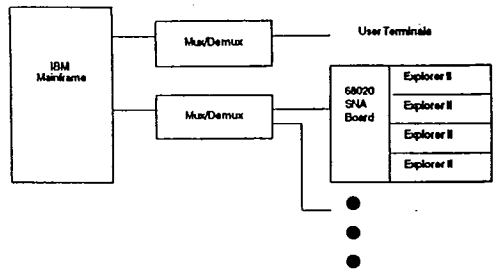

Figure 9: DLMS Deployment Architecture

The hardware platform is the Texas Instruments Multi-Processor (MP). The MP utilizes a 16-slot Nubus architecture and allows 4 Explorer II processors to share the same chassis and console. System software supports inter-processor communication and individual processor rebooting. It also allows processors to share a common load band.

Communication with the IBM host is effected via the TI SNA II processor which acts both as a 3274 terminal controller and as a terminal emulator.

Software Tools

The rule-based system components were implemented in ART from Inference Corporation. All other software was written in Common Lisp.

Current Status and Future Development

The system was developed with continuous input from the process and industrial engineering communities. Because the potential consequences for the organization are significant, the system is being deployed incrementally. This is critical as the quantity of knowledge required to competently handle all Ford vehicles is huge and the integrity of the knowledge base is of paramount importance. The system was introduced experimentally six months ago and is being used by a small subset of the process and industrial engineering community. The system is part of a major initiative at Ford Motor Company and will continue to be built on for a number of years to come.

Acknowledgements

The authors would like to thank Thomas S. Kazmareck for his invaluable contribution to the initial specification and design of the DLMS system. Thanks are also due to Philip Klahr and Philip C. Jackson, Jr. who were extremely helpful in criticizing early drafts of this paper.

References

[Allen 87] James Allen, "Natural Language Understanding," pp. 81-89, pub. Benjamin Cummings 1987, ISBN 0-8053-0330-8.

[Brachman and Levesque 84] Ronald J. Brachman and Hector J. Levesque, "The Tractability of Subsumption in Frame-based Description Languages," Proceedings AAAI-84, Austin, TX, pp. 34-37.

[Kaczmarek, Bates and Robins 86] Thomas S. Kaczmarek, Raymond Bates and Gabriel Robins, "Recent Developments in NIKL," Proceedings AAAI-86, Philadelphia, PA, pp. 978-987.

[Lipkis 81] Thomas Lipkis, "A KL-ONE Classifier," Consul Note #5, USC/Information Sciences Institute, 4676 Admiralty Way, Marina Del Rey, CA 90291.

[Mark 81] William Mark, "Representation and Inference in the Consul System," in Proceedings of the Seventh Joint Conference on Artificial Intelligence, IJCAI, 1981.

[Moser 83] M. G. Moser, "An Overview of NIKL, the New Implementation of KL-ONE," Bolt Beranek and Newman Inc., Cambridge, MA, Rep. 5421, 1983.

[Nebel 88] Bernhard Nebel, "Computational Complexity of Terminological Reasoning in BACK," Artificial Intelligence 34 (1988) pp. 371-383.

[Robins 86] Gabriel Robins, "The ISI Grapher and the ISI NIKL Browser," Internal memo, USC/Information Sciences Institute, 4676 Admiralty Way, Marina Del Rey, CA 90291.

[Schmolze and Lipkis 83] James G. Schmolze and Thomas A. Lipkis, "Classification in the KL-ONE Knowledge Representation System," IJCAI, 1983.

[Weischedel & Sondheimer 83] Ralph M. Weischedel and Norman K. Sondheimer, "Meta-rules as a Basis for Processing Ill-formed Input," American Journal of Computational Linguistics, Volume 9, numbers 3-4, pp. 161-177.

[Winograd 83] Terry Winograd, *Language as a Cognitive Process,* Vol. 1, pp. 195-267, pub. Addison Wesley 1983, ISBN 0-201-08571-2.