Capturing Lessons Learned for Variation Reduction in an Automotive Assembly Plant

John A. Cafeo, Diane I. Gibbons, Ronald M. Lesperance, Alexander P. Morgan, Gülcin H. Sengir, and Andrea M. Simon

General Motors R&D and Planning Center
Mail Code 480-106-359
Warren, MI 48090-9055
810-986-2157, 810-986-0574 fax
alexander.p.morgan@gm.com

Keywords: Lessons Learned, Case-Based Retrieval, Automotive Manufacturing, Ontological Engineering, Knowledge Management, Variation Reduction, Quality

Abstract

Reducing variation in the assembly of a vehicle body is a critical element in improving its quality. Assembly plants have dimensional-management teams, who monitor variation, respond to "variation crises," and continually reduce variation when there are no crises. These teams have, in the past, not kept archival records and have not shared solutions with other teams. This paper describes our project to develop a lessons-learned system for variation reduction. The elements of the project include developing an ontology, designing a case structure, defining similarity, and developing a user interface for case reporting and case-based retrieval.

Introduction

Reducing variation in the early part of the assembly of a vehicle body is the critical element in improving its quality. If the variation is low, the nominal points of the build can be tuned to match their design specifications. Then, many important customer-satisfaction issues, such as wind noise, water leaks, door-closing effort, and "fit and finish," will be easily controlled. (Here, "build" refers to the vehicle while it is undergoing assembly and also to the assembly process. This "product-process" duality in language is common in manufacturing applications.) On the other hand, if the variation is high, then the build will always be "off" and impossible to tune. Assemblies and components will not fit together properly, and many difficult-to-diagnose-and-repair issues will arise, both during assembly and after the vehicle is sold.

Each assembly plant has a *dimensional-management* (DM) team, which addresses variation-reduction and other dimensional-control problems. These teams have, in the past, not kept archival records of their work and have not shared solved problems with other teams. Sometimes, issues solved one year must be solved again the next "from scratch," because both records and memory fail to recall the details of past solutions.

This paper gives an overview of our efforts to introduce a systematic knowledge-management approach to sharing and archiving lessons learned (Weber et al., 2000) by the DM teams. Elements of this work include developing an ontology for dimensional control in automotive body shops, establishing a simple "bulletin board" for daily sharing, designing a case structure for archival lessons that would satisfy reporting needs and at the same time facilitate a case-based retrieval mechanism, and designing a user interface that would be acceptable to the process engineers and technicians. Our project is in its early stages, but we have learned enough to make a preliminary report.

Problem Environment, Case Structure, and Ontology

Our project focuses on the body shop of a truck assembly plant. The body shop's role within the assembly center is to make the cab, the fenders, the hood, the doors, and the bed from stamped sheet metal parts that are produced at other facilities. Figure 1 shows an example of a cab. The body shop is organized into a number of zones. The output of each zone is a sub-assembly and the inputs can consist of a combination of stamped metal parts or sub-assemblies from an outside supplier or sub-assemblies from a previous zone. Within each zone, the input parts are loaded, positioned and then joined. Some of the zones have vision stations to measure the output dimensional characteristics of the sub-assemblies. The analyses of these data are key pieces of information used by the DM team in identifying and solving dimensional control and variation issues. Because variation reduction (VR) is the keystone of dimensional

control, we have focused on it in our project. This is a somewhat arbitrary limitation, which could be easily removed. We call our software the VRAdviser.

We need to say explicitly that in solving VR problems, there is often not an obvious link between the observed problem on the vehicle body and its root cause. However, experienced problem solvers tend to work by linking a current problem to previous similar problems, which then provide a starting point for the diagnosis. This is one of the main reasons that solving VR problems is a good potential application for a case-based approach. See, for example, (Watson, 1997) or (Leake, 1996, Chapter 2).

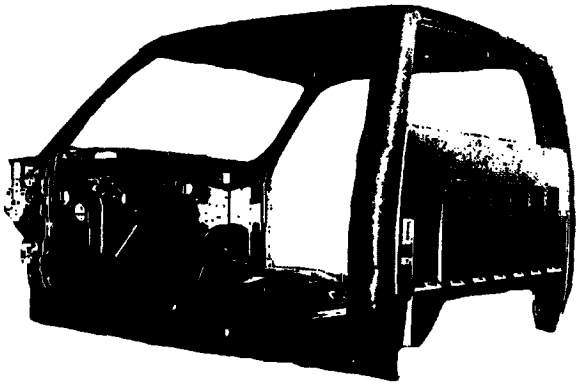

Figure 1 Example of a truck cab manufactured at the truck assembly plant.

A typical VR "case history"

- begins with the first symptom,
- proceeds to generate further "symptoms" through the statistical analysis of measurement data (including correlation studies, drill down analyses, principal-component analysis animations),
- goes on to generate a list of "possible root causes" at the approximate midpoint of the case,
- moves to physical tests and inspections, and
- is completed by final tests that confirm the root cause and cure.

Let us note an important point for the definition of similarity. As is well understood in fault analysis, problems cascade "downstream" until a symptom is observed, while diagnosis proceeds by tracking symptoms "upstream" until the root cause is revealed. It follows that two cases can be similar, but have entirely different root causes. Suppose case A begins with fault X, which then causes fault Y, itself causing fault Z, triggering fault W, which directly creates the initial symptom. Imagine case B beginning with fault Y. The two cases might be identical, except that X is the root cause of A and Y is the root cause of B. When we want to solve B, we will be particularly interested in seeing A. This means that our similarity function will need to consider sub-case similarity in a way that strongly favors sub-case matching, while not penalizing the parts that don't match. This is not a typical "similarity function," as described for example in (Watson, 1997).

There are other aspects of this environment that lead us not to expect to identify a fixed set of attributes whose values define a definite problem and unique solution. The diagnostic process can involve many data tests and physical tests, and the technicians want the freedom to decide which tests to run and the order in which they are done. [This point was made in (Hamilton, 2000), where a case structure with some elements in common with ours is described.] Further, a good deal of the "knowledge content" of the case history is in the chain of causation and the chain of discovery. Consider: a weld robot programming error causes excessive weld tip pressure, which leads to excessive weld tip wear in station 5 at the same time as a warping of the assembly that exits station 5, which leads to a locating pin in station 23 becoming loose, which causes an assembly-to-assembly variation which is observed via analysis of measurement data further downstream. It is the combination of the loose pin in station 23 with excessive weld-tip wear in station 5 that signals the proper diagnosis. These faults might occur separately, but then the root causes might be different. Representing every possible fault in a fixed set of attributes or organizing diagnosis around a small number of questions linked to attribute values [as described in (Watson, 1997, Chapter 7)] is not reasonable here. In particular, each case may have a different number of relevant attributes, and the approach to comparing cases for similarity must take this into account. The "information completion" framework described in (Lenz, 1998) in a context of "case retrieval nets" does provide a formal structure which we might use, but the practical issues of dealing with "mismatching cases" and developing a workable similarity function remain. We should also note that similarity is not our only concern. The lessons-learned case structure must be capable of generating reports which express the problem and solution in terms that are meaningful to the DM team members. Thus, some items of data which might be discarded to generate a case index must be preserved in the cases themselves.

In the light of these (and other) considerations, we have developed the following case structure. A case consists of an arbitrary number of "observations." There are 24 types of observations. Each type has a fixed structure, essentially a fixed set of attributes, although we allow lists of values for some attributes. Thus, two observations of the same type from two different cases may be compared with each other in a one-to-one manner, as long as some way of comparing lists of values for the same attribute is available. Only a few types of attributes are important for similarity. However, all are significant for case capture and reporting.

There isn't space in this brief paper to consider the ontology in detail. Because we want the observation templates to be organized around "menu selection" rather than "authoring" and because the VR world is relatively limited, the ontology consists of a rather complete set of plant-floor object names with verbs and verb phrases which express process elements, faults with process elements, tests, results of tests, and other such objects. These are partially organized taxonomically, somewhat like the hierarchical domain model for the diagnostic support application described in (Bergmann et al., 1999, chapter 9). There is also a library of pictures showing, in particular, the relevant parts, components, sub-assemblies, and assemblies. These pictures are mostly design diagrams generated in the course of planning the build, but made active for point-and-click selection of specific parts and assemblies. They have a natural taxonomic classification derived from the parts breakdown of the build. A small number of the pictures describe the body-shop layout. Their taxonomy follows the factory-floor organization by zones, stations, and machines. All of these pictures can be used to annotate observations. This is extremely important, because a great deal of the dimensional-management knowledge is expressed in terms of pictures. Observations can be annotated by text as well as pictures, but we have set out to create the VRAdviser so that text annotations are optional. Some multimedia objects would also be relevant for diagnosis and might be attached to observations in the future; for example, principal-component analysis animations, virtual reality presentations showing how the assemblies should fit into the tooling and interact with the machines, the sound of specific machine faults, etc.

One interesting element of manufacturing applications is the product-process duality of expression which is commonly used. In other words, process problems and their corresponding effects on the product are often referred to interchangeably and with a certain ambiguity:

- "The Station A build is off in the down direction."
- "The pins in Station A are low."
- "The left front fender needs to be moved up 1mm."
- "The pins in station A need to be moved up 1mm."

are (hypothetically) all saying essentially the same thing. We see that an additional element of ambiguity can be added by referencing a process fault and an action to correct the fault interchangeably. We can imagine cases with these kinds of variations of expression which we would want to match during similarity search. Part of the motivation for using observation templates and a limited ontology is to control the number of ways of saying the same thing. We have not tried to eliminate the product-process duality of language, however, as this is too much a part of standard assembly-plant usage. Rather, we have tagged dual expressions to be similar.

Analytical cases were created to seed the VRAdviser, in an exercise with the DM team in which we visited each station, talked about what could go wrong with process elements in that station and what effects on the build each fault might cause and where these faults would most likely be detected. This information was synthesized into a set of about 50 analytical cases, which serve as an independent resource (e.g., they can be browsed) as well as being seed cases for similarity search.

User Interface

We were told when we began the VRAdviser project that the DM team members would have to be able to enter a case in two minutes and not have to do any writing at all. Otherwise, they would not have the patience to use the tool. Our compromise is that an experienced user can enter an observation in two minutes. The 24 types are organized under 8 tabs: data test, data analysis, correlation, product problem, process problem, action, comparison, interpretation. The observations are essentially fill-in-the-blank sentences, which can be completed using pull-down menus that select objects from the ontology. Choices are limited to what makes sense for the particular slot being filled in. The DM team is fairly picky about language and generally dislikes generic nouns and verbs. Behind the scenes, however, generic information (e.g., the inner leaves of taxonomies) are used in the way the pull-down choices are organized, for similarity computations, and for generating summary statistics over all cases. The pictures are organized for selection using the taxonomies noted above.

At the request of the DM team we provided an additional element, a "shift log," in which random unstructured comments can be made, organized by date and shift and zone. These notes are read by one shift to see what the last few shifts have been working on. They contain direct messages from one shift to the next. Thus, they perform the function of a simple bulletin board for the team. Shift log entries can be used to identify problems that will become cases, but many of them do not.

The DM team uses a specialized in-house statistical package to analyze the data from the vision stations. This data analysis generates statistical information including graphs and charts, which can be cut and pasted into observations as annotations.

In the spirit of fast prototyping and easy integration, we used Microsoft Access and Visual Basic to build the VRAdviser. The similarity function is written in C.

Similarity and Case-Based Retrieval

Given the complexity of the case structure, it is natural to generate an index for each case and define similarity in terms of the indices. Similarity is developed by considering case relevance from the point of view of the specific DM team problem-solving process. In other words, when the team is considering a new case, what sorts of reference cases would it want to be reminded of? There are several natural focal points:

- Similarity of location, in terms of the geography of the plant; e.g., cases may involve the same zone, the same station, or contiguous zones or stations.
- Similarity of location, in terms of the geography of the body; e.g., involve the roof line, involve the left front part of the cab.
- similarity of structure, in terms of the vehicle body; e.g., the roofline and the door ring are both boundary regions whose dimensional control is critical.
- similarity of process function, e.g., two welding processes or (more generally) two joining processes.
- similarity of tools, e.g., pins and clamps, or robots, or glue guns.

We have not done much field testing of our similarity function, and we expect to refine and simplify it as it is used. Our current thinking is that if case A *is similar in any way* to case B, then the DM team would want the cases and the nature of their similarity brought to its attention. The VRAdviser is not for a "naïve user." Rather it is to aid the diagnostic reasoning of human experts. As a new case is "being completed," additional information will be entered but "... the additional information may not come directly from the most similar cases, [but rather these similar cases] might suggest tests to be performed in order to obtain [the] new information." (Lenz et al., 1998, p. 65) Note, too, that this is an application which will always have only a "few" cases, as opposed , say, to certain web-based or e-business applications. Significant VR problems occur at the rate of about one a week in a body shop. Even multiplying this over 20 assembly plants and 10 years, the number of cases is on the order of 10,000. With proper case-base management, the number should remain considerably smaller than this.

Current Status

Most of our time last year has been spent in learning how to approach the ontological engineering and in building a good working relationship with the plant. We have entered a fairly complete starter set of analytical cases, and we are getting a good deal of feedback from the DM team. The comments are positive, but the number of modifications requested is large. This is as expected in prototype development. The number of cases captured so far makes premature any assessment of our approach to similarity.

Summary and Future Work

We have launched a prototype variation-reduction adviser. This includes

- developing an ontology which makes sense in terms of case reporting and case retrieval requirements, and
- seeding the system with a set of analytical cases.

When the number of cases grows to a large enough number, we will test our similarity strategy. We expect to modify and simplify, but we do not anticipate this to be a roadblock to completion of the project.

The long term intent is to link all assembly plants to the same case base, so that lessons learned in plant A can benefit plant B. There are (essentially) no ontological issues for several plants building the same product. For different products, the ontological issues have to do with "lifting" the specific language favored by the DM teams to generic language which could suggest solutions across platforms. This is our eventual goal.

References

Bergmann, R.; Breen, S.; Göker, M.; Manago, M.; and Wess, S. (eds.) 1999. *Developing Industrial Case-Based Reasoning Applications: The INRECA Methodology.* Lecture Notes in Artificial Intelligence 1612, Springer-Verlag, Berlin.

Hamilton, A. and Gomes, B. 2000. Failure Analysis of Semi-conductor Products. In Innovative Customer Centered Applications, Trento, Italy, Sept. 2000, oral only.

Leake, D. (ed.) 1996. *Case-Based Reasoning: Experiences, Lessons, & Future Directions.* AAAI Press, Menlo Park, California and The MIT Press, Cambridge, Massachusetts.

Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S. (eds.) 1998. *Case-Based Reasoning Technology.* Lecture Notes in Artificial Intelligence 1400, Springer-Verlag, Berlin.

Watson, I. 1997. *Applying Case-Based Reasoning: Techniques for Enterprise Systems.* Morgan Kaufmann, San Francisco.

Weber, R.; Aha, D. W.; Munoz-Avila, H.; and Breslow, L. A. 2000. Active Delivery for Lessons-Learned Systems. In Blanzieri, E. and Portinale, L. (eds.) *Advances in Case-Based Reasoning: 5th European Workshop, EWCBR 2000, Trento, Italy, September 2000, Proceedings*, 322-334. Lecture Notes in Artificial Intelligence 1898, Springer-Verlag, Berlin.