Automation and Robotics in Construction XI
D.A. Chamberlain (Editor)

Providing a drilling robot with the right instructions

Ronald P. Krom[a,b]

[a]TNO-Building and Construction Research,
PO. Box 49, 2600 AA, Delft, The Netherlands

[b]Delft University of Technology, Faculty of Civil Engineering,
PO. Box 5048, 2600 GA, Delft, The Netherlands

Abstract

This paper presents the concept and implementation of a task-level instruction system for an autonomous drilling robot used in railway installation work. The research objectives are: to reduce the required programming effort, make more intelligent robot behaviour possible and bring the programming abstraction level to a task-instruction level. Relevant railway construction knowledge and task related information is stored in the drilling robot using product models. Standardised exchange formats for product models enables re-use of CAD information by the drilling robot which significantly reduces the required robot instruction effort.

1. INTRODUCTION

Since 1991 TNO Building & Construction Research has been involved in the research for, and development of an autonomous drilling robot (See also [1] and [2]). This drilling robot is specially designed to drill large series of holes in concrete floors. The most challenging application for which the robot is being used, is the drilling of series of holes for rail fixtures. These fixtures fasten railway rails to a concrete floor in a railway station tunnel at the Amsterdam airport. (see Figure 1). This application requires an accuracy in the positions of the drilled holes of ± 2 mm. The required accuracy is achieved using the CAPSY positioning system. The CAPSY system uses a rotating laser beam to measure directions to bar-code reflectors at known locations to calculate it's current location (see [3, 4] for further information about CAPSY).

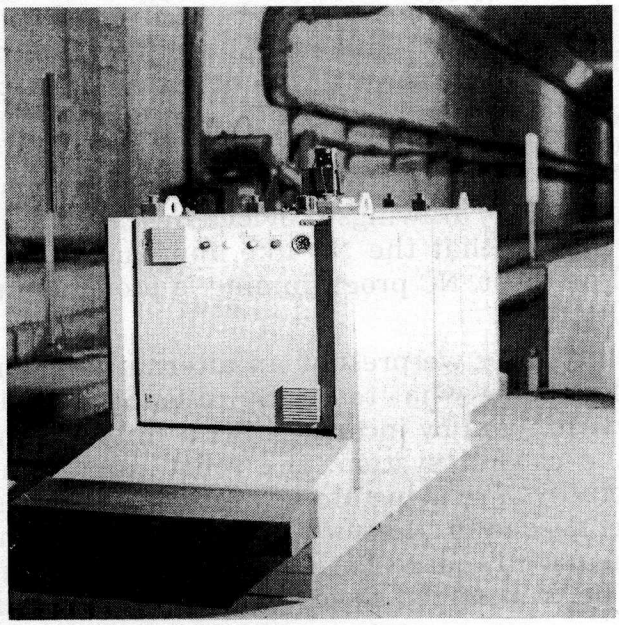

Figure 1: Picture of the drilling robot.

The rail fixtures being used, consist of a steel base plate of approximately 35 x 20 cm with two diagonally placed holes. Two glue anchor bolts fasten the base plate to a concrete slab. A double drill in the drilling robot drills the holes for

the two glue anchors in one go at correct relative positions.

The most important benefits of the use of the drilling robot are:

- The elimination of the uncomfortable manual drilling work. At the Amsterdam Airport station nearly 20 000 holes are needed for rail fixtures.
- The automation of the labour intensive surveying work required for the positioning of the rails.

Because the drilling robot is able to automatically position itself at the right locations, the tasks of the project surveyors has changed. In the current, non robotised situation, the surveyors indicate the approximate locations of the base plates on the concrete slabs. During the rail installation the surveyors measure the track position every few metres to ensure the correct position and curvature of the rails. Using the drilling robot, the surveyors' task changes: They have to position the reference reflectors used by the CAPSY positioning system and they have to provide the drilling robot with the location co-ordinates of the rail fixtures to be installed.

At first glance it seems that the rail fixture locations belong to a repeating pattern. However this is not the case for two reasons:

- A relative large proportion of the tracks has a non straight horizontal alignment. (The alignment is the description of the trajectory of the track.) In non-straight alignment sections the fixture locations have to be adjusted to follow the alignment.
- The concrete floor is divided into sections of varying length. In order to avoid fixtures on or too near section joints, the spacing between fixtures is reduced near the section ends.

The most straight forward instruction method for the robot would be to provide a list of co-ordinates of the locations where a hole is required. This programming method is similar to the Numerical Control (NC) method used to control machines in mechanical industries. This NC-like approach has a number of serious drawbacks:

- NC programming is not very efficient. Unlike robot applications in manufacturing industry where identical operations are performed many times, construction robot operations are repeated only once. The programming effort on construction robot applications can not be 'written off' over many products.
- The volume of data in the program becomes unmanageable. In total approx. 10 000 rail fixtures are to be installed. Putting all the co-ordinates into one list will result in a huge list (or many small lists) in which humans will not be able to detect or correct errors.
- The co-ordinates from all rail fixtures must be extracted (manually) from technical (CAD) drawings provided by the railway designers.

We believe that the NC-like approach is not a suitable method to instruct the drilling robot. NC programming is probably unsuitable for most construction robot applications.

In this paper we present an alternative, more efficient approach to instruct the drilling robot what to do. By re-using already available computerised *design information* and by incorporating a right amount of *railway construction knowledge* in the control system, the robot can be instructed with minimal effort. The effectivity of re-using already computerised design information, is influenced by the effort needed to transfer this information to the robot's control system. Using parts of the ISO STandard for Exchange of Product model data (STEP), design information can be transferred from CAD systems into the robot using a neutral form. The

availability of railway construction knowledge in the robot controller also enables the possibility for high-level instruction and intelligent behaviour of the robot.

Chapter two describes the approach used in the information transformation process in which rail fixture locations are deducted from design information using knowledge. Chapters three and four describe two sub processes of the transformation process. Chapter five describes our prototype implementation. Finally in chapter six presents some concluding remarks.

2 THE HIGH LEVEL CONTROL SYSTEM

In this chapter we look at the robot instruction system as an information processing system. The purpose of the developed high-level control system is to generate the rail fixture locations with optimal effectivity; i.e. with as little input information and as little programmer's effort as possible.

In order to reach an optimal effectivity of the drilling robot instruction method, it is necessary to find a balance between:

- *entering (new) information into the system*
  Entering information has the advantage of flexibility, but the disadvantage of being time consuming and potentially unreliable.
- *(re-)using information available from elsewhere*
  A disadvantage of using already computerised information is that it introduces extra communication interfaces with the outside world. Such interfaces need to be defined and different information translators are needed when there is more than one information supplying system.
- *deducting information from other information*
  When *knowledge* (which is information that is *understood* and can be *applied*) is present, required information can be deducted from other information. This requires knowledge to be formalised and entered into the computer. Representing knowledge in (compiled) computer programs reduces the flexibility of the system.

The drilling robot control system combines three sources of information to generate the fixture locations. Two sources can be transferred from elsewhere and one source is formed by (internal) knowledge. The three sources are:

- *information about the track alignment*
  The alignment is a three dimensional curve which describes the trajectory of a railway. The alignment design is an important part of the railway design. Dedicated CAD systems are used to design the alignments. The alignment information is available in computerised form in the dedicated CAD systems. The transfer of such CAD road alignment designs (which are identical to rail alignment designs) using STEP-standardised files has been demonstrated in the past at the Dutch ministry for public works.
- *information about the division in sections*
  The length of each of the tunnel sections is measured after the construction of the sections. This information is also available in computers.
- *knowledge about the fixture layout regulations and code of practice*
  In order to ensure end result quality, guidelines exist for the layout design and installation of rail fixtures. These guidelines are available in textual documents but the guidelines are of a procedural nature and therefore this knowledge can be represented as an algorithm.

The IDEF$_0$ diagram in Figure 2 shows how the main control system process, called *'calculate rail fixture locations'*, is decomposed into two sub processes. The first sub process generates a fixture layout taking the section lengths into account but disregarding the rail alignment. The second sub process 'bends' the fixture layout along the alignment. In other words, the effect on the fixture locations caused by the division into sections is be handled independently of the effect of the alignment on the locations. The next chapter discusses both sub processes in more detail.

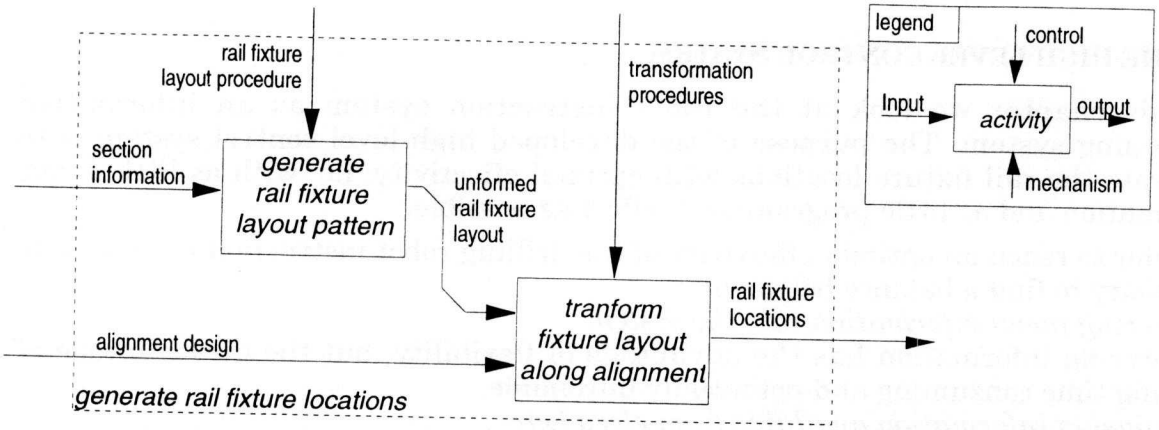

Figure 2: IDEF$_0$ diagram of decomposition of drilling robot high level control process.

An important technique used in the developed high-level instruction system is *product modelling*. The use of product models in the robot controller provides two major advantages:

- Product models provide a structured information representation to which knowledge can be related and on which the knowledge can be applied. Product models do not provide facilities to represent algorithmic or expert system knowledge.
- Existing STEP parts and software tools are available for standardised exchange of product model data.

Using the STEP technology we can provide standardised, neutral interfaces for the robot controllers input and output streams. Part 21 of STEP [6], titled *Physical File Format* (PFF), provides a specification for the file format for the exchange of product models. The STEP PFF is a computer interpretable ASCII representation (which can also be read by humans) for (product) models containing entities, attributes and relations.

The next chapter discusses the sub processes presented in Figure 2.

3 THE RAIL FIXTURE LAYOUT GENERATION

The first step in generating the rail fixture co-ordinates is to generate the fixture layout pattern using the design rules. The design rules are not very complicated; The rails are fastened every 600 mm to concrete slabs of 3450 mm length and 850 mm width. Because the length of sections (which varies between 18 and 22 m) does generally not correspond to a multiple of 3600 mm (3450 mm + 150 mm gap), the length of one or more slabs is reduced in order to fit the slabs in a section. Therefore the distance between fixtures on shorter slabs is reduced up to approx. 50 mm on shortened slabs, to ensure that the fixtures are evenly spaced and stay clear of the slab edges.

Using only the section length and a set of design rule parameters such as the maximum spacing reduction, a list of fixtures chainage locations can be generated (the chainage is the name of the co-ordinate along the one dimensional alignment co-ordinate system). Figure 3 shows a Pascal-like pseudo-code description of this fixture location algorithms representing the layout rules.

```
procedure generate_fixture_layout(integer: section_length);

Constants
   SLAB_LENGTH                := 3600;
   NR_OF_FIXTURES_PER_SLAB    := 6;
   MAXIMUM_SPACING_RED        := 50;
   NORMAL_FIXTURE_SPACING     := 600;

begin
   nr_of_slabs                := section_length div SLAB_LENGTH + 1;
   nr_of_fixtures             := nr_of_slabs * NR_OF_FIXTURES_PER_SLAB;
   length_reduction           := SLAB_LENGTH * nr_of_slabs - section_length;
   reduction_per_fixture      := length_reduction div (NR_OF_FIXTURES_PER_SLAB- 1);
   nr_of_shortened_slabs      := reduction_per_fixture div MAXIMUM_SPACING_RED + 1;
   red_fixture_spacing        := NORMAL_FIXTURE_SPACING - length_reduction div
                                    (nr_of_shortened_slabs * NR_OF_FIXTURES_PER_SLAB- 1);
   fixture[1].chainage        := NORMAL_FIXTURE_SPACING div 2 - length_reduction mod
                                    nr_of_shortened_slabs;

   for i:= 2 to nr_of_fixtures do begin
      if (i <= (nr_of_shortened_slabs * NR_OF_FIXTURES_PER_SLAB)) then
         spacing       := NORMAL_FIXTURE_SPACING ;
      else
         spacing       := red_fixture_spacing;
      fi
      fixture[i].chainage  := fixture[i-1].chainage + spacing;
   end;
end;
```

Figure 3: Pseudo code of fixture layout algorithm.

When the rail fixture chainages are known, X and Y co-ordinates of the fixture locations can be calculated in the alignment transformation process.

Unfortunately no standard does (yet) provide a neutral form for the exchange of (algorithmic) knowledge. Therefore the rail fixture layout procedures can not yet be represented in a neutral form. However perhaps predicate logic, a well designed object oriented programming language such as EIFFEL or a standardised language such as the STEP EXPRESS language can be used for this purpose.

4 THE ALIGNMENT DEFORMATION PROCESS

The second step in the generation of the fixture locations is the transformation of the unformed layout so that it will follow the horizontal alignment. The internal alignment model used in the drilling robot control system is based on the results of the development of a national standard road design information called Road Model Kernel [7]. The development of the Road Model Kernel is an initiative of the Dutch ministry of public works.

The horizontal alignment is a 2-dimensional curve describing the railway trajectory as if going through a perfectly flat landscape. The horizontal alignment is represented by a list of *alignment elements*. Figure 4 shows the NIAM diagram of

the horizontal alignment representation being used. Each of the alignment elements of the alignment design is either a *straight*, a *circular curve* or a *transition curve*. The purpose of the straight elements and the circular curved elements is obvious. The transition curves are used in order to obtain a smooth (linear) change of the centrifugal force in the transition between straight elements and circular curves. Every alignment element is bounded by two *tangent nodes*. These tangent nodes have a *chainage* to indicate their location on the alignment.

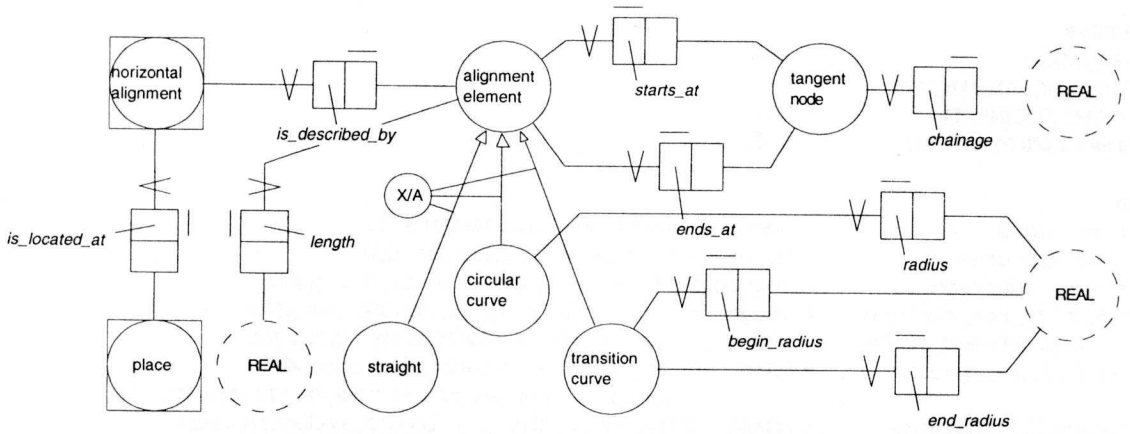

Figure 4: NIAM diagram of the entities and relations of alignment model.

An example of such an alignment design is shown in Figure 5. This figure shows a top view on the horizontal alignments of three tracks near the platform area at the Amsterdam airport railway station.

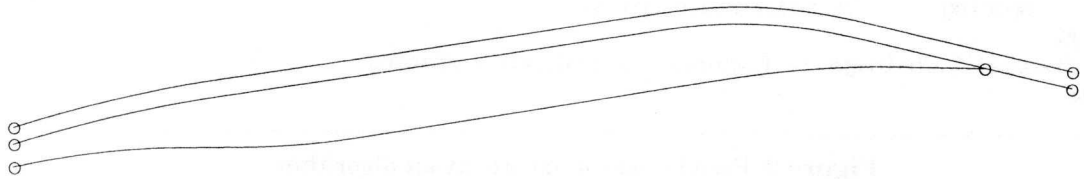

Figure 5: Horizontal alignment of three tracks in the station near the platforms.

Using the Road Model Kernel representation for alignment designs, alignment data can be transferred from a CAD system into the drilling robot in a neutral form.

The transformation process

In the transformation process the information from the alignment model is used to construct a list of transformation operations. Each of these transformations transforms a straight line into a curve following the alignment (such as shown in Figure 5). Each transformation operation is only valid in small domain which is bounded by the chainages of the begin and end tangent nodes of the alignment element.

The list of transformation operations, derived from the list of alignment elements, can now be applied to the unformed rail fixture layout. Figure 6 shows a generated fixture layout transformed along an exaggerated curve to show the effect of the transformation.

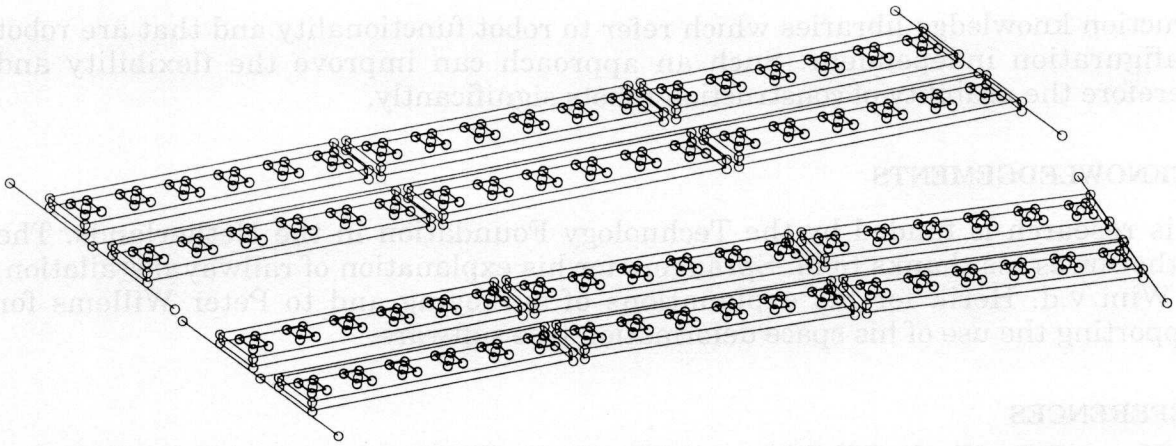

Figure 6: Perspective view on visualisation of generated fixture layout in exaggerated curve.

5. A PROTOTYPE COMPUTER IMPLEMENTATION

Using the TNO developed case tool 'PMshell' (Product Modelling shell) [8] a prototype implementation is made. PMshell is a tool with which conceptual models such as described with NIAM diagrams can be transformed into working implementations. PMshell generates object oriented C++ programs in which every entity is implemented as a C++ class. Because PMshell has knowledge about the models being handled, it automatically generates STEP PFF interfaces for the program being constructed.

In the implementation of the prototype drilling robot control system, the rail fixture layout procedure is represented in a (parameterised) C++ Class *member function*.

Alignment designs created in the CAD system MOSS can be transferred to the drilling robot via a prototype MOSS translator which produces RMK-standardised STEP PFF-files.

CONCLUSION

The one-of-a-kind character of the construction robot's task requires effective robot instruction methods. The re-use of already available, computerised information is an effective approach to reduce construction robot instruction efforts. Available STEP technology provides neutral interfaces for exchange of information between robot control systems and their environment consisting of CAD and planning systems.

Major advantages of the use of high-level product models in robot control systems are:

- availability of high-level product and production information enables more intelligent behaviour of robots. Robot control software has more information available to analyse circumstances to adjust the robot's behaviour.
- availability of high-level information enables improved operator dialogue. Instead of instructing the robot in terms of primitive operations the robot can be instructed in the same terms as used for human instruction.

Although product models enable incorporation of construction knowledge in the control system, there is not yet a suitable neutral representation for construction knowledge. With such a representation it is perhaps possible to build neutral con-

struction knowledge libraries which refer to robot functionality and that are robot configuration independent. Such an approach can improve the flexibility and therefore the usability of construction robots significantly.

ACKNOWLEDGEMENTS

This research is funded by the Technology Foundation in the Netherlands. The author owes his thanks to M. Sprangers for his explanation of railway installation, to Wim v.d. Herik for his explanations of surveying and to Peter Willems for supporting the use of his space deformation tree software.

REFERENCES

1. Krom, R.P., Kloek, R.B.P.M., and Vos, Ch.J., *The Development of a Drilling robot for the installation of railway tracks*. in: *The 10th International Symposium on Automation and Robotics in Construction (ISARC)*, 1993, Houston, Elsevier Science Publishers, p. 347-354.
2. Kloek, R.P.W.J., Bos, J., and Marck, R.M.S. van der, The development and Testing of an Autonomous Drilling Robot. in: The 11th International Symposium on Automation and Robotics in Construction, 1994, Brighton, England, Elsevier Science Publishers.
3. Vos, L.B.C. de and Hasara, B., *Field Applications with CAPSY*. in: *The 10th International Symposium on Automation and Robotics in Construction (ISARC)*, 1993, Houston, Elsevier Science Publishers, p. 463-470.
4. Vos, L.B.C de and Schouten, J.N.Th.M., *The Computer Aided Positioning SYstem (CAPSY)*. in: *The 6th International Symposium on Automation and Robotics in Construction*, 1989, San Francisco, .
5. ISO/TC184, *Part 1: Overview and fundamental principles,* in: *Industrial automation systems and integration - Product data representation and exchange.*, 1993, ISO, Geneva, Switzerland.
6. ISO/TC184, *Part 21: Clear Text Encoding of the Exchange Structure,* in: *Industrial automation systems and integration - Product data representation and exchange.*, 1993, NIST, Gaithersburg, USA.
7. Willems, P.H., *The Road Model Kernel, Version 0.2.*, 1990, TNO Building and Construction Research.
8. Luijten, B.F.M. and Luiten, G.T., *A layered-modelling tool for the integration of computer applications,* in: *Object-oriented applications,* Meyer, B. and Nerson, J.M., Editor(s), 1993, Prentice Hall, Hertfordshire, UK