Automation and Robotics in Construction XII
E.Budny, A.McCrea, K.Szymanski (Editors)



# Object-Oriented Planning- and Controlling-System for a Wall Construction Robot

T. Bock / D. Stricker / T. Huynh / J. Fliedner

Division for construction automation
Faculty of mechanical engineering in construction
University of Karlsruhe, D-76128 Karlsruhe

## Abstract

In this article we present several important aspects of a software system control. This is designed and developed for a wall assembly robot in an European Esprit III project called ROCCO, RObot assembly system for Computer integrated COnstruction.

The system consists of an off-line program for planning of complex assembly tasks and for generating robot actions. Furthermore, an on-line command control is made possible through an adaptive user interface.

All the software is designed with object-oriented concepts and implemented in C++. The wall assembly system is organized on the base of the successive generation of different types of actions, called "Mission", "Task", and "Action". They represent different levels of complexities. Those different actions are organized in a tree structure. The control of each level is done through the user interface window.

Furthermore, the software system can be connected to a CAD-robot simulation software for checking the robot assembly motions. Added to the control system, a recovery module is implemented for all possible errors during the construction.

## Introduction

The ROCCO project intends to automate the construction process from the design to the construction on the building place. In the work preparation phase, the necessary informations for the pre-fabrication of the different assembly parts and the construction site layout are calculated. Based on the architectural CAD-representation of the building, the assembly sequence and the correlated optimal working position of the mobile robot are determined. With the informations about the position of the bricks on the palettes and in the wall and the position of the robot, the off-line robot program generates the assembly sequences.

In order to assure the exchange and the update of the datas in an on-line mode, the robot system has to have the qualities of an efficient controlling information system. For our application, it was particularly interesting to design the software with help of the object-oriented (OO) concept and to base it on an OO-database management system.

First the OO-model of the world and of robot activities will be presented. Secondly, several aspects of the algorithm will be explained and at the end we will show the strategy used for the robot motion.

## 1. OO-model for the wall construction robot

### 1.1. Worldmodel

In the case of automatic building construction, the considered elements are physical objects like stones, walls, palettes and working places, which can easily be described through two principal attributes. One is the global position, the other are the dimensions of the object, for instance, the hight, the width and the depth. These two different attributes are present in all the considered objects; they will be used to model all the world objects.

The worldmodel is build up from several classes, which are in some kind inherited from each other. The base class for all classes is the database class called "Object", from which are directly inherited the classes "frame" and "geometry". The first one describes a position and an orientation in form of a matrix. It is completed with the inherited class "position", which gives an identifier of the frame and a reference in which it is valid. The class geometry is a container for a description of the shape of an object. The attributes of the object are the width, the hight and the depth and a list of points giving if necessary, each edge-coordinates in the object frame.

These two classes are joined to create the class "physical-object", which represents the main class for describing the world. It contains all the necessary components for the model of an real object present in the construction site. The subclasses have been designed to specify and to improve the structure of the worldmodel through corresponding functionalities. For example a list of frame objects has been added to constitute the class "storage". It is adapted to model a palette or a wall composed of a list of stone positions.

The following figure gives an overview of those classes:

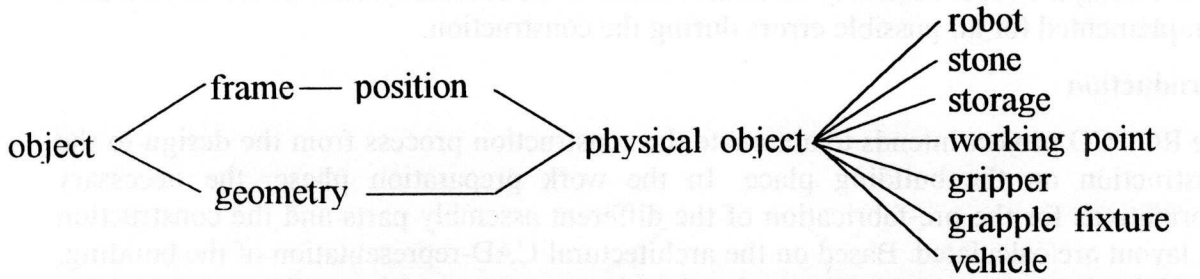



Figure 1.1: Worldmodel structure

### 1.2. Model of the robot activities

The robot activities can be modeled with help of three classes corresponding to three levels of the robot operations. The first level gives the global aim of the robot operations; the second one describes only a sequence of the global task and the last one gives a single motion of the manipulator. The three classes are called the class "mission", "task" and "action". They inherit from the root class "activity". This last one contains the main attributes used for the description of a robot operation.

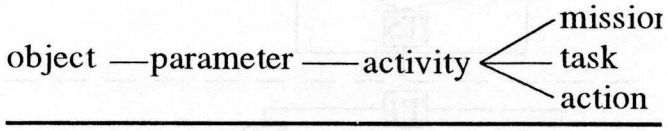



Figure 1.2: Activity model

## 2. Algorithm of the control program

### 2.1. Program architecture

In order to organize the different types of activities, other attributes like a list of children objects have been added. They subdivide an activity into several others, which describe it with more details. On this way, the three activity types correspond to three levels of action complexities. In the figure 2.1.a, a mission called "Assemble" has been divided into four tasks: Transport, Pick up, Transport and Place. And for example, the task "Pick_up" orders the actions: Approach, Grapple, Extract and Retract.

To execute an activity, the children objects and the corresponding modules have to be called. At the end of the tree, Explicit Elementary Operations (EEO) give a numerical conversion of the activity "Action". An interpreter reads the EEO-objects and sends the command to the robot or to the simulation (see figure 2.1.b).

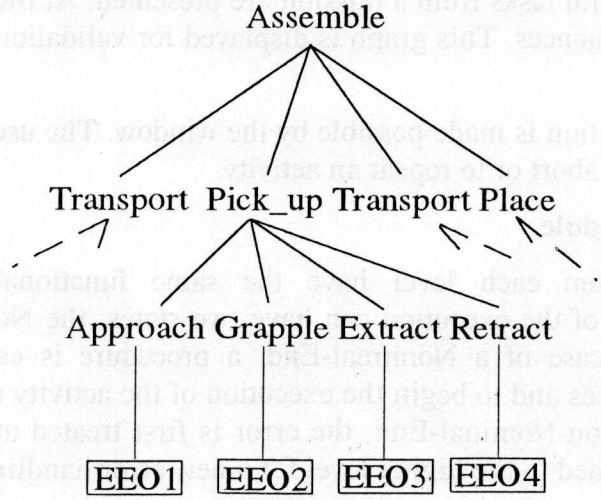



Figure 2.1a: Subdivision of an assembly mission

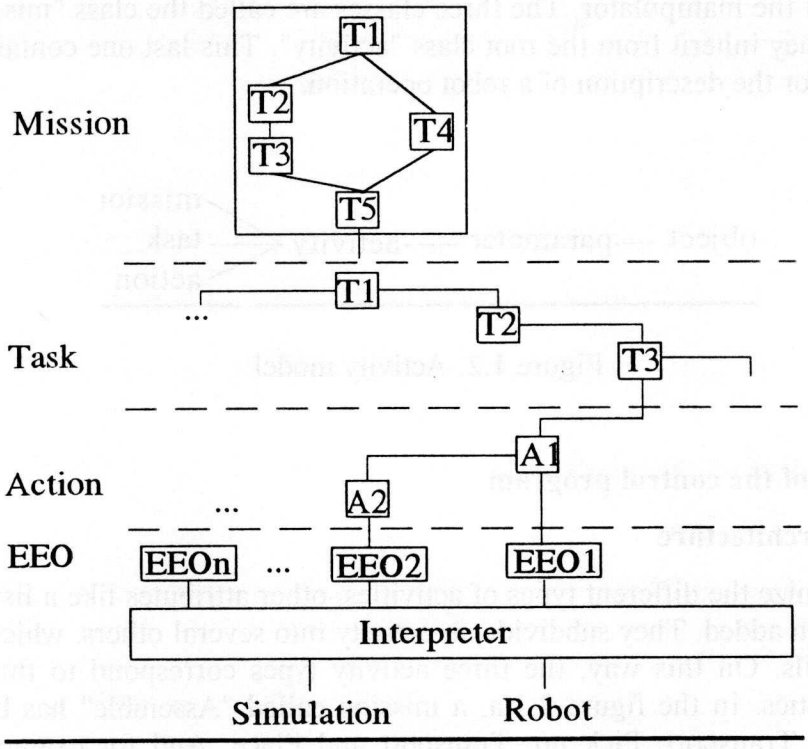



Figure 2.1b: Overview of the activities execution

In the figure above, several tasks from a mission are presented. At the first level, a graph shows the execution sequences. This graph is displayed for validation on the user screen too.

The control of the execution is made possible by the window. The user has the possiblity to confirm, to ignore, to abort or to repeat an activity.

### 2.2. Error recovery module

The activity-objects from each level have the same functional and hierarchical architecture. The result of the execution can have two states: the Nominal-End and the Non-Nominal-End. In case of a Nomimal-End, a procedure is called to update the worldmodel circumstances and to begin the execution of the activity of the next children object. In case of the Non-Nominal-End, the error is first treated at the corresponding level and then it is returned to the higher level for a new error handling.

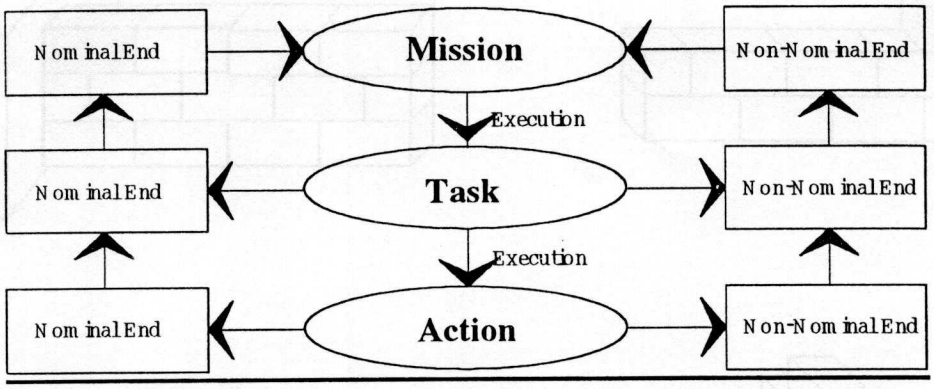


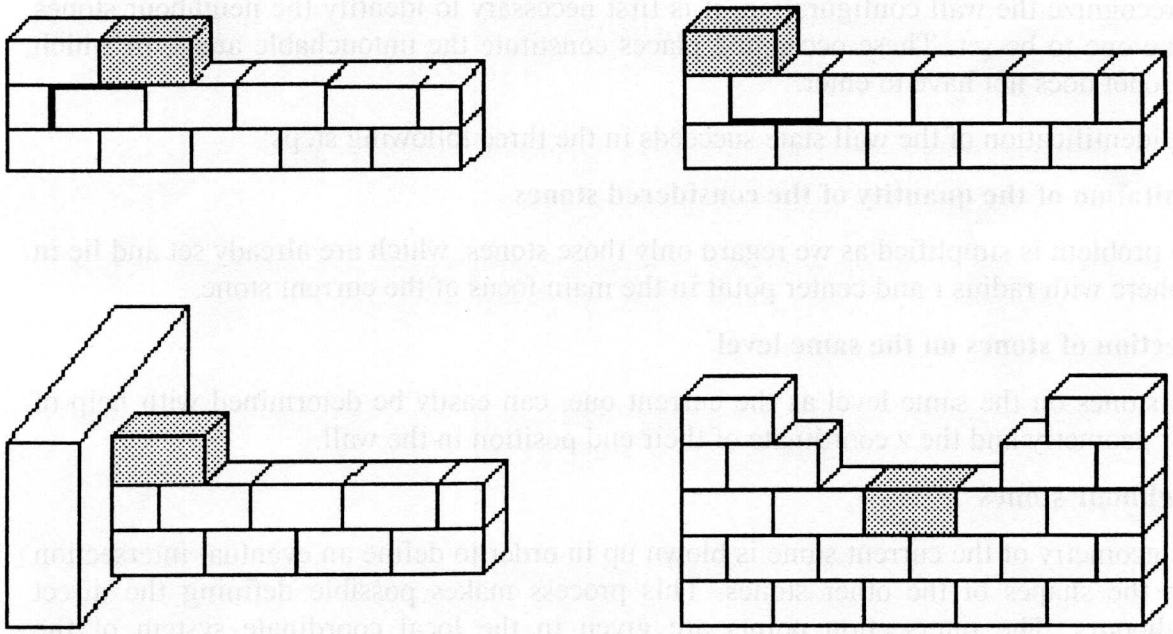

Figure 2.2: Nominal and Non-Nominal feed back

## 3. Wall assembly strategy

To achieve the assembly robot motions, all the different configurations of the wall have to be established. A recognition module identifies the current state of the wall and changes the trajectory, for instance, the way to set the stone in the wall.

### 3.1. Possible configurations of the wall

The wall configurations for a given stone can be limited to the following:



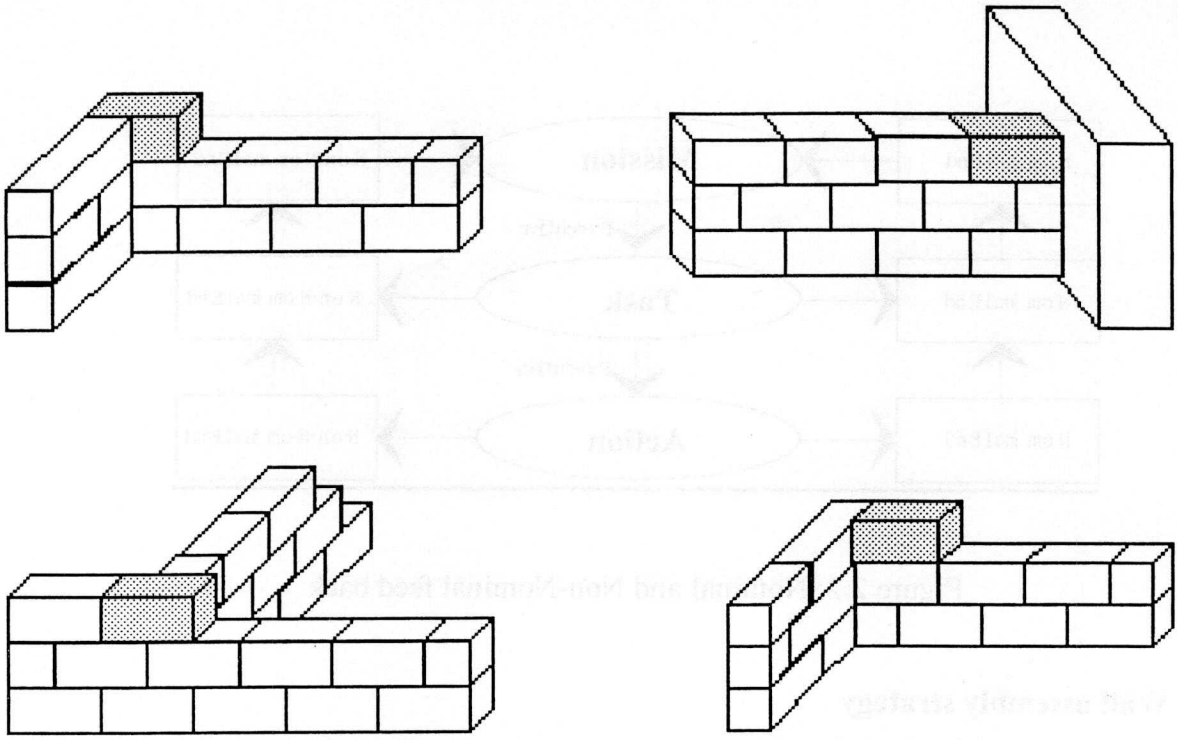


Figure 3.1: Possible insertion cases

### 3.2. Identification of the wall configuration

To recognize the wall configuration, it is first necessary to identify the neighbour stones of the one to be set. These occupated places constitute the untouchable areas, in which the robot does not have to enter.

The identification of the wall state succeeds in the three following steps:

**Limitation of the quantity of the considered stones**

The problem is simplified as we regard only those stones, which are already set and lie in a sphere with radius r and center point in the main focus of the current stone.

**Selection of stones on the same level**

The stones on the same level as the current one, can easily be determined with help of their geometry and the z coordinate of their end position in the wall.

**Neighbour stones**

The geometry of the current stone is blown up in order to define an eventual intersection with the shapes of the other stones. This process makes possible defining the direct neighbours. The intersection points are given in the local coordinate system of the current stone. With these informations, it is possible to know for each stone the relative positions to its neighbours.

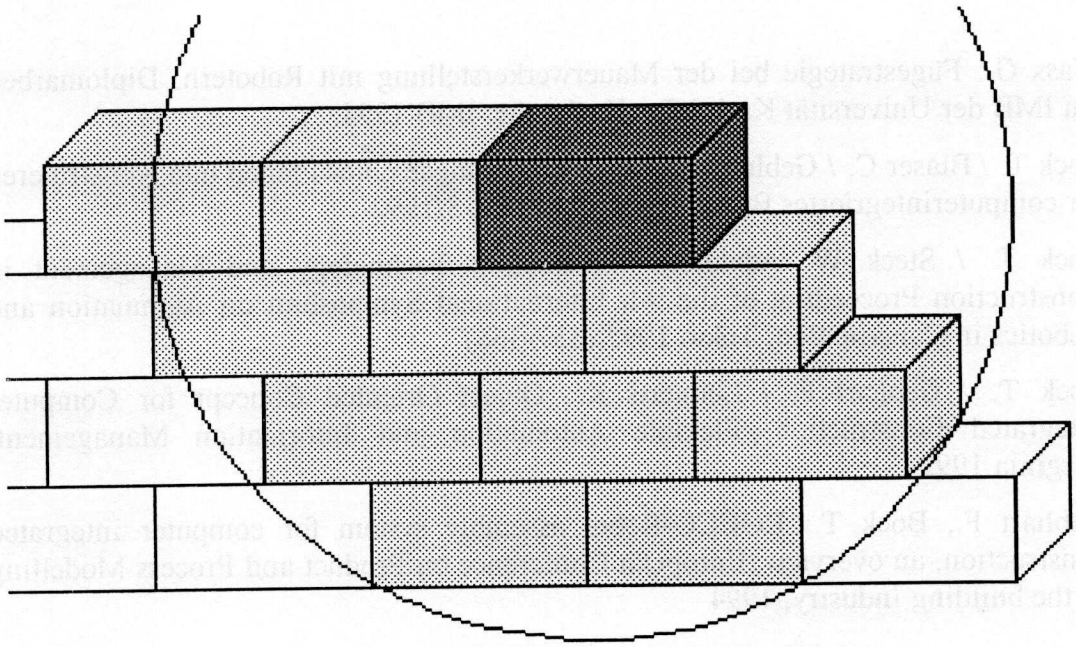


Figure 3.2: Limitation of the number of regarded stones

### 3.3. The insertion vector

With the datas explained above we defined the insertion vectors. They are perpendicular to the current stone surfaces and are oriented outside of the surfaces, like shown in figure 3.3. In this way, the contact surfaces and their orientations is known for each stone. The trajectory points of the insertion action can then be calculated with help of the end position and an offset in the direction given from the insertion vectors.

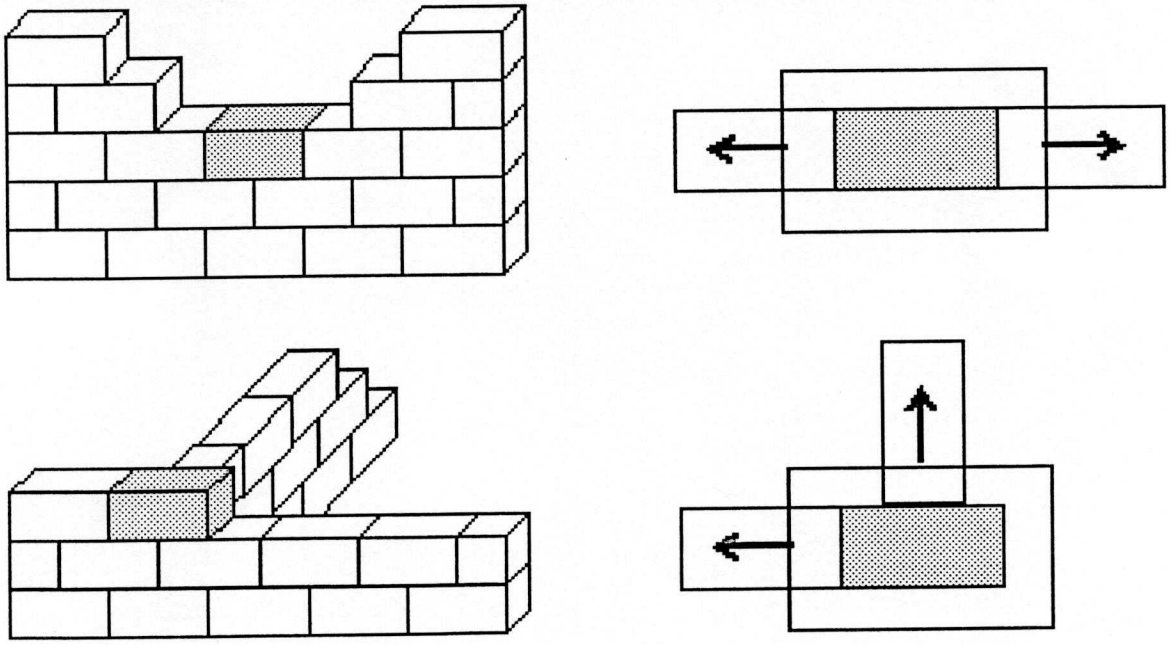


Figure 3.3: Examples of insertion vectors

## References

[1] Wass G.: Fügestrategie bei der Mauerwerkerstellung mit Robotern, Diplomarbeit am IMB der Universität Karlsruhe, Karlsruhe: IMB 1994

[2] Bock T. / Blaser C. / Gebhart F.: Automatisierungsrechtes Planen und Konstruieren für computerintegriertes Bauen, Bautechnik 69, 3/1992

[3] Bock T. / Steck W: Advanced Information Technology and Management in Construction Proceeding of the 9th International Symposium on Automation and Robotics in Construction, Tokio 1992 (357-366)

[4] Bock T. / Gebhart F. / Lennerts K.: Object Oriented Concept for Computer Integrated Construction, Flexible Automation and Information Management, Virginia 1992

[5] Gebhart F., Bock T.: ROCCO-Robot assembly system for computer integrated construction, an overview, European Conference on Product and Process Modelling in the building Industry, 1994