Program Supervision under Resource Constraints

Sabine Moisan[1] and **Monique Thonnat**[2]

Abstract. Fields such as image processing, signal processing or scientific computing have produced a large number of powerful programs and many different libraries of data processing programs have been developed. But few support is provided to new-comers or inexperienced users. Thus, for these programs to be widely applied, there is a need for an intelligent support. To help users of program libraries, we propose a knowledge-based approach, that we call *program supervision*. A program supervision system takes in charge the management of the library use i.e. the (re)configuration of data processing programs. Depending on the context of execution, a program supervision system has to take into account various constraints on resources, e.g. hardware, software or time. A typical case is real-time applications, where data flows coming from sensors have to be processed in limited time and under hardware constraints. This paper summarizes our program supervision approach and focuses on how program supervision can take into account resource constraints, and especially time constraints. A real-time experiment in program supervision illustrates this problem.

1 RESEARCH FRAMEWORK AND MOTIVATION

Fields such as image processing, signal processing or scientific computing have produced a large number of powerful programs and many different libraries of data processing programs have been developed. But few support is provided to new-comers or inexperienced users, who need to use these programs in order to solve their application problems. The proposed programs often implement complex functionalities and their use is therefore difficult for end-users who do not have a deep understanding of program specifics. Thus, for these programs to be widely applied, there is a need for an intelligent support. An optimal use of the most efficient and accurate programs is especially necessary when resources are limited.

To help users of program libraries, we propose a knowledge-based approach, that we call *program supervision*. A program supervision system takes in charge the management of the library use i.e. the (re)configuration of data processing programs. Such a configuration may involve the chaining of many different programs. It is to note that the program codes pre-exist (in a library, for example) and the goal is not to optimize the program themselves, but their *use*. A program supervision system emulates the strategy of an expert in the selection, combination, and execution of the programs.

Program supervision provides means to capitalize the knowledge on the optimal use of various programs in different situations The knowledge-based approach allows *flexibility*, i.e. changes in the number or the characteristics of programs are easy to integrate. Moreover, for a novice user, a program supervision system takes in charge all the necessary steps to build a complex processing plan, and to control the intermediary and final results, due to the *evaluation* and *repair* mechanism. This aid is more than just an advisory guide and it can leads to fully automatic program monitoring systems. The system can adapt itself to different input data and automatically backtrack in case of bad results.

Depending on the context of execution, a program supervision system has to take into account various constraints on resources, e.g. hardware, software or time. A typical case is real-time applications, where data flows coming from sensors have to be processed in limited time and under hardware constraints. This paper summarizes our program supervision approach and focuses on how program supervision can take into account resource constraints, and especially time constraints. A real-time experiment in program supervision illustrates this problem.

2 PROGRAM SUPERVISION

Program Supervision [11, 3, 1] is a very general approach that has been applied in image processing, scientific computing, signal processing, automatic control, etc. It aims at automating (partly or completely) the configuration of data processing programs.

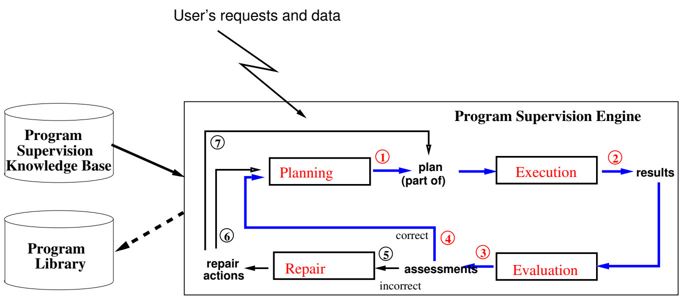

Figure 1. Program supervision engine phases

For this purpose, a program supervision system encapsulates knowledge about the best use of the programs, which may be complex for unexperienced users. Indeed, the program applicability frequently depends on the data contents, on the goal to reach, etc. Moreover, their execution often implies the precise tuning of sensitive input parameters. Knowledge-based systems provide means to express and to structure this kind of knowledge. That is why we have chosen a knowledge-based orientation to embody the advanced expertise about program use. A program supervision system is composed of a program supervision engine and a knowledge base.

The engine exploits the knowledge about programs which is available in the knowledge base in order to configure the best plan of

[1] Inria BP 93 06902 Sophia Antipolis Cedex, France
[2] Inria BP 93 06902 Sophia Antipolis Cedex, France

ECAI 98. *13th European Conference on Artificial Intelligence*
Edited by Henri Prade
Published in 1998 by John Wiley & Sons, Ltd.

programs, that achieves the user's goal. The reasoning of the engine may be decomposed into different phases (see figure 1). First a **planning** phase selects and assembles programs to build a (part of) plan (1), an **execution** phase then executes the actual program in the plan (after tuning of their parameter values, etc.) and produces results (2). The **evaluation** phase uses expert criteria to evaluate the results and returns an assessment (3). If the assessment is positive (4) planning can go on. In case of bad result assessment (5) a **repair** phase decides either to repair the plan (by replanning (6)) or to modify some parameter values and re-execute previously executed programs (7).

A program supervision knowledge base stores representations of programs and of typical combinations of programs, as well as criteria to guide the reasoning process. A program representation (called operator) contains its complete calling syntax, descriptions of its input and output arguments (e.g. name, type, usual parameter values, etc.) and its conditions of use. These information are used by the engine to select the programs, to initialize their parameters, to manage non trivial data-flow and to combine the programs to produce a satisfactory plan of programs, depending on the input data, constraints and request. Various criteria are attached to operators. They play important roles during the reasoning phases. They are used to choose between different alternatives, to tune program execution, or to diagnose and repair a bad execution.

When resources are limited, this general model should be adapted to take into account resource constraints. Some of the engine phases may be kept to a minimum or simply disappear if they are incompatible with the constraints to handle. An example is shown in the next section, where for reasons of efficiency, and due to the real-time nature of the application, the evaluation and repair phases are skipped.

3 A REAL-TIME EXPERIMENT

Within the framework of the Eureka European Prometheus project [7], a program supervision system has been embarked on a demonstrator vehicle. The aim was to provide an on-board driving assistance system (or co-pilot) to help the driver in negotiating road obstacles (cars, pedestrians, etc.) in realistic traffic situations.

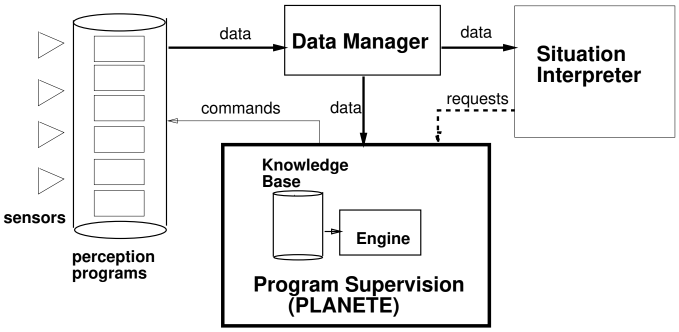

Figure 2. Co-pilot architecture

The co-pilot is an intelligent real-time system which processes the information flowing into it from a variety of sensors mounted on the vehicle (5 CCD cameras, 1 linear stereo camera pair and one telemetry-camera combination, in addition to a number of proprioceptive sensors), uses this information to interpret the traffic situation in the vicinity of the vehicle, and communicates its analysis to the driver by various means. The co-pilot is composed of three modules: a situation interpreter, a data manager, and a program supervision system, as shown in figure 2.

Compared to the generic model of program supervision described in the previous section, real-time program supervision cannot perform sophisticated repair and evaluation. That is why they do not appear in the engine, whereas a request manager appears to manage the flow of requests that must be processed, as shown in figure 3. Indeed, the role of the program supervision system, named PLANETE [8, 5, 10], is to convert the flow of symbolic requests coming from the high-level situation interpreter, into the correct low-level processing commands, under computational and time constraints. A request, expressed in terms of zones, obstacles, etc., was translated by the program supervision system into a series of low-level commands—expressed in terms of sensors, program names, data flow, etc.—to the perception programs. The results of the processing done by these programs are then communicated via the data manager to the situation interpreter, for further analysis. The duration of a complete loop is 200ms.

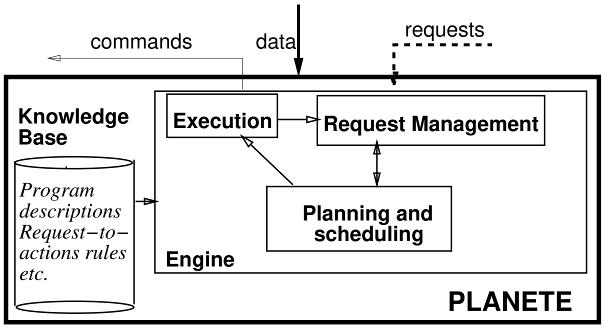

Figure 3. Zoom on PLANETE main modules

The knowledge base of PLANETE contains the representations of perception programs, with attached criteria that guide the supervision process. Criteria are represented by inference rules. In a real-time context the number and complexity of rules are kept to a minimum, because their execution must be rapid enough and must not disturb the perception process. An example of program description is shown figure 4. A program description includes its name, its input tunable parameters (with their range of variation), its ouput, etc. Tunable parameters are explicitly described and some rules on how to set their values are attached. This example also shows a simple repair rule that triggers when the result of the program are bad. In the Prometheus application, programs communicate via a shared memory, for efficiency reasons. That is why the input and ouput data are implicit in the first version of PLANETE program descriptions.

We now detail an example of the engine reasoning. Having detected that the driver intends to overtake (because the direction indicator is on), the situation interpreter emits a "detection" request, to detect possible obstacles around the car. In response to this detection request, the *planning and scheduling* module of PLANETE triggers the following expert rule to generate a list of actions to be executed, these actions mainly consist in changing the mode of execution of some programs:

ECAI 98. *13th European Conference on Artificial Intelligence*
Edited by Henri Prade
Published in 1998 by John Wiley & Sons, Ltd.

Name	3D_telemetry	
Input Parameters		
	execution_mode	[scanning,tracking]
	sub_image	
	num_lines	[0-80]
	obstacle_number	
	contout_threshold	[10-30]
	obstacle_position	
Output		
	target_lost	[true,false]
Repair Criteria		
	if (target_lost) then mode = scanning	
Parameter Setting Criteria		
	if (mode == tracking) then mode = scanning every 5 seconds	

Figure 4. Example of a perception module description in PLANETE knowledge base, with some criteria.

```
if (request_type == detection &&
    (traffic_context == road
     || traffic_context == motor-way))
{
   change_mode (3D_program,
                       continuous_scanning);
   change_mode (Math_Morph_program,
                        obstacle_detection);
   deactivate(Motion_Segmentation_program);
}
```

This rule (in C-like syntax) states that if the request is a detection and the current context traffic is either road or motorway, the modes of two programs must be changed, while another program has to be deactivated.

In this case, the 3D telemetry program computes obstacle 3D positions. It must work in continuous scanning mode to detect all obstacles. The mathematical morphology program performs @D shape analysis. In obstacle detection mode it searches for possible obstacles. The two programs return complementary information about surrounding obstacles. This information is merged by the data manager, to produce an improved environment map in terms of a list of perceived obstacles together with their position and velocity. This map is then available to the situation interpreter to make further decision for the next cycle: either to warn the driver if some obstacles pose a safety threat, or to terminate the request if there is no obstacle. It is to note that several different kinds of request may run in parallel and compete for software resources. It is also of PLANETE's responsibility to (de)activate sensors or to allocate a software resource to a program rather than to another one.

The task of PLANETE was to make the perceptual processing optimal in the sense of efficiency, timeliness and responsiveness to the rapidly-varying informational requirements of the co-pilot. For this purpose, PLANETE encapsulated all the problem-solving knowledge relating to the perception programs, and to the possible driving scenarios. The prototype vehicle has been demonstrated in October 1994 at the end of the Prometheus project [6].

From this experiment we concluded that program supervision can be adapted to different resource constraints and that providing a generic model is an help for this adaptation. But PLANETE must be improved on several points, and in particular the lack of evaluation and repair is a weakness. It is possible to introduce these important phases, while keeping them to a minimum, *i.e.* using them for occasional monitoring rather than continuous result optimization.

4 CONCLUSION AND FUTURE WORK

In program supervision systems the emphasis is put on selecting the most pertinent programs with respect to the current situation and data, rather than on increasing the speed of the data processing. In very critical real-time situations, it is not possible to evaluate all the results. We consider improvements in our current evaluation mechanism to allow experts to specify when evaluation should take place and with which frequency.

Concerning resource limitation, we want to pursue in the direction of embarked and real-time program supervision systems. Our goal is a generic real-time program supervision tool, adapted to planning, scheduling, and control of execution in real-time, and capable to handle complex requests on data flows. We will enrich our general program supervision model with ideas from existing real-time knowledge-based systems in planning and scheduling [12, 4, 2]. Predictions about the expected duration and result quality of different programs (using for example statistical techniques) would improve the planning phase. We also intend to use a priori information about the duration of programs, given by an expert. The plan generation could take into account this information to design a plan respecting given time constraints.

But time is not our only concern. Memory space, data, hardware, and software are also resources that programs may share and their use should also be optimized. More and more of our application for program supervision are distributed ones (e.g. telemedecine). In this framework, full real-time is less important than constrainst involved by sharing nodes in the net and large data.

We are also currently working on extensions of the expressiveness of expert evaluation criteria. On the one hand, we will define a general vocabulary to describe curves in symbolic terms, because curve characteristics are a customary way for expert to evaluate result quality. On the other hand, we investigate a case-based evaluation mechanism, where the experts could compare current results with known references.

REFERENCES

[1] R. Bodington, 'A Software Environment for the Automatic Configuration of Inspections Systems', in *Proceedings of KBUP'95*, pp. 99 – 108. INRIA, Sophia Antipolis, INRIA, (November 1995).

[2] F. Charpillet, 'A Look on the Issue of Building Real-Time Knowledge-Based Systems', in *AAAI Fall-Symposium on Flexible Computation in Intelligent Systems*, (November 1996).

[3] S. A. Chien, 'Automated Synthesis of Image Processing Procedures for a Large-scale Image Database', in *First IEEE International Conference on Image Processing*, Austin, TX, (November 1994).

[4] M. Ghallab and H. Laruelle, 'Representation and Control in IXTET, a Temporal Planner', in *AIPS*, (1994).

[5] D. Hutber, S. Moisan, C. Shekhar, and M. Thonnat, 'Perception-Interpretation Interfacing for the Prolab2 Road Vehicle', in *7th Symposium on Transportation Systems: Theory and Application of Advanced Technology*, Tianjin, Chine, (August 1994).

[6] S. Moisan, C. Shekhar, and M. Thonnat, 'Real-Time Perception Program supervision for Vehicle Driving Assistance', in *ICRAM'95 International Conference on recent Advances in Mechatronics*, eds., Okyay Kaynak, Mehmed Ozkan, Nurdan Bekiroglu, and Ilker Tunay, pp. 173–179, Istanbul, Turkey, (August 1995).

[7] Groupe PROART, 'Rapport de fin de contrat de recherche PROMETHEUS-PROART', Technical report, Projet PROMETHEUS, (December 1994).

ECAI 98. *13th European Conference on Artificial Intelligence*
Edited by Henri Prade
Published in 1998 by John Wiley & Sons, Ltd.

[8] C. Shekhar, V. Gaudin, S. Moisan, and M. Thonnat, 'Real-time Supervision of Perception Programs for Prolab2', in *IMACS International Symposium on Signal Processing, Robotics and Neural Networks*, eds., P. Borne, T. Fukuda, and S.G. Tzafestas, pp. 11–14, Villeneuve-d'Ascq, France, (April 1994).

[9] C. Shekhar, S. Moisan, and M. Thonnat, 'Towards an Intelligent Problem-Solving Environment for Signal Processing', *Mathematics and Computers in Simulation*, **36**, 347–359, (1994).

[10] C. Shekhar, S. Moisan, and M. Thonnat, 'Use of a Real-Time Perception Program Supervisor in a Driving Scenario', in *Intelligent Vehicle Symposium'94*, Paris, France, (October 1994).

[11] M. Thonnat and S. Moisan, 'Knowledge-Based Systems for Program Supervision', in *Proceedings of KBUP'95*, pp. 3–8, Sophia Antipolis, France, (November 1995). INRIA.

[12] M. Wozniak, P. Taillibert, and P. Vanheeghe, 'TCLP: a temporal causal link planner with disjunctive temporal relations', in *UK Workshop on Planning and Scheduling*, (December 1997).

ECAI 98. *13th European Conference on Artificial Intelligence*
Edited by Henri Prade
Published in 1998 by John Wiley & Sons, Ltd.