constraint programming, decision support, ASP

Paweł SITEK, Jarosław WIKAREK[*]

Towards intelligent decision support for application service providing

Abstract

Decision support systems (DSS) provide decision-makers with an interactive environment for analyses of information with various models to help solve unstructured and NP-hard problems. The important aspect of DSS is a technical and technological approach to the design and implementation of the above systems. A traditional approach to DSS engineering and implementation requires a great deal of effort for its maintenance. However, the enterprises would like to concentrate on its core competitiveness instead of non-core activities like IT maintenance. As a result of this, IT outsourcing has became a very popular event. Thus there is a growing need for intelligent decision support tools capable of assisting a decision maker in many problems in SMEs (Small and Medium Sized Enterprises). In this paper we present the use of declarative programming (constraint logic programming and relational SQL database) as an environment and framework for such decision support systems in an application service providing (ASP) model.

1. INTRODUCTION

An important aspect of decision support systems studies is to develop techniques for automatic or interactive decision analysis in a complex real-world situation. Decision makers face the problem of making optimal choices in uncertain situation under given constraints with various sources of knowledge (often semi-structured or ill-structured). Provided that a decision support system is an interactive computer-based system that helps decision-makers utilize data and models to solve unstructured problems [1], an aspect of decision support systems studies is to develop techniques of modelling of decision problems and data management in a unified framework.
Another aspect of decision support systems is technical and technological approach to DSS design and implementation as an information system (IS).
A traditional approach to IS engineering and implementation requires a great deal of effort for its maintenance. Up to 70% of information technology (IT) cost is tied up in maintenance in many enterprises [2].

[*] PhD, PhD, Technical University of Kielce, Control and Management Systems Section, 1000-PP 7, Kielce, Poland, E-MAIL: sitek@tu.kielce.pl, j.wikarek@tu.kielce.pl

However, the enterprises would like to concentrate on its core/base competitiveness instead of non-core/non-base activities like IT and specially IS maintenance and management. On the other hand IT has also been changed –a rapid development of Internet, computing environments, lower costs of hardware, new programming paradigms and so on. As a result of the above premises, IT outsourcing has become a very popular event. Outsourcing is considered to be an important way in the evolution of the IT. A service scope of IT outsourcing has been extended so that the hardware, the software, the application, the network, the business process and the know-how can be covered.
Thus, there is growing need for intelligent decision support tools capable of assisting a decision maker in many problems in SMEs. In this paper we present the use of declarative programming (constraint logic programming and relational SQL database) as an environment and framework for such decision support systems in ASP model.

2. ASP MODEL

IS outsourcing has asked for decades but gained attention during the 1990s as well-known firms signed large contracts, primarily to cut costs or to concentrate on their core business [3]. Client applications were transferred to vendor machines or the vendor bought the client's systems. In the late 1990s, application service providers (ASPs) offered products, such as Enterprise Resource Planning (ERP) systems, as services available through networks such as the Internet. ASP has been defined as a single point of contact for all the telecommunications, hardware, software, and consulting necessary to deploy, run, and maintain hosted applications remotely. One would expect that the provision of an application service (AS) shares many features with conventional IS outsourcing as well as offering alternative ways of outsourcing. However, ASPs usually deliver services through the Internet, increasing uncertainty about availability and response time and, in contrast to conventional (non-ASP) outsourcing, AS provision necessitates coordination among network providers, hosting services, software vendors, and consultants.
Like any information systems (IS) marketplace development, the ASP model may enjoy great success or be replaced quickly by other market or technical offerings.

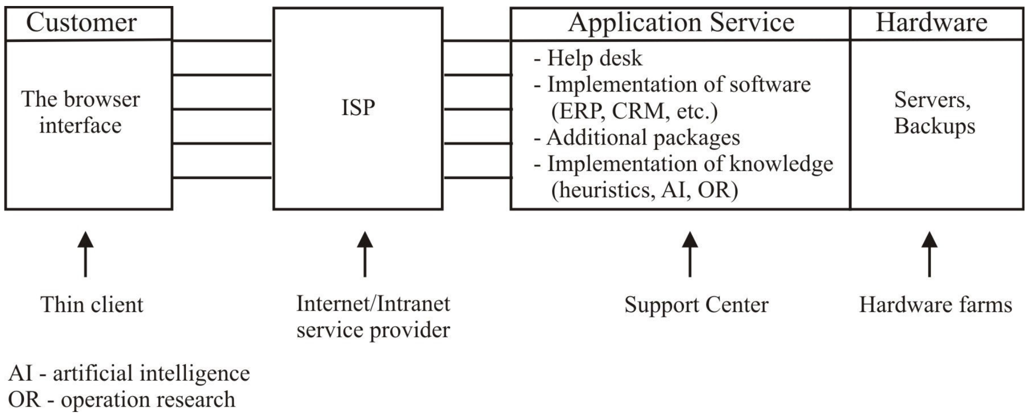


Fig. 1 The structure of ASP model

The important components of ASP's operational environment have been shown in fig. 1 [8]:

- **The browser interface** – the browser is now (during Internet revolution) the door to customer's world. Using a browser interface as the front end of an application keeps the cost of both infrastructure and technical support very low for the customer.
- **The internet service provider (ISP)** – the ability to deliver first class systems both nationally and internationally has been a challenge. For SME's the cost of dedicated lines has generally been prohibitive. SME's can now have access to bandwidth that allows complex and real time movement of information around the world. Nowadays, the central challenge of ISP is maintaining sufficient response over the WEB. For instance, some ERP systems generate multiple updates to multiple files from a single transaction entry by the user. While this updating generally occurs in the back end system (and therefore will not impact directly on response), the flow of information between the back end system and multiple users can be substantial. Response time to the user is therefore a central challenge that the ISP must address.
- **Support center**
    - **(implementation of the software)**-regardless of the size of a client, each business will seek to have his or her own needs reflected in the software. The business challenge for the ASP is to maintain enough consistency across clients in order to gain economies in the maintenance and support processes. Especially, SMEs or other customers without ERP experience will probably settle for the cheaper, more quickly implemented standard system. To deliver tailored ERP systems, the ASP will need experienced implementers who can customize the software to meet client's needs. Timing is also an essential element of success.
    - **(managing change in the client)-** Delivering new software over the Web does not vary the need to train staff in a new system or new processes. Traditional issues such as staff communication, change management, and training are therefore essential if client staff is to accept the new systems. The ASP will need to provide adequate on-site support to the client before and during live operation. Tailored courses, either face to face or via the Web will also support successful implementation.
    - **(help desk)-** A call center support is essential where an ASP is providing ``mission critical" systems to the operation of a business. The ASP must staff itself to provide a full range of services including on-site work. ASPs themselves require support from the vendor.
- **Hardware farm-** The hardware farm, as the name implies, provides capacity to the ASP for the delivery of the ERP, CRM or other IS software. The most important ingredient is the ability to grow the capacity to meet the needs of the client.

3. DECLARATIVE PROGRAMMING AND ENVIRONMENTS – SQL, CLP

Declarative programming is a term with two distinct meanings, both of which are in current use. According to one definition, a program is "declarative" if it describes *what* something is like, rather than *how* to create it. For example, HTML, XML web pages are declarative because they describe *what* the page should contain — title, text, images — but not *how* to actually

display the page on a computer screen. This is a different approach from imperative programming languages such as Pascal, C, and Java, which require the programmer to specify an algorithm to be run. In short, imperative programs explicitly specify an algorithm to achieve a goal, while declarative programs explicitly specify the goal and leave the implementation of the algorithm to the support software (for example, an SQL select statement specifies the properties of the data to be extracted from a database, not the process of extracting the data).
According to a different definition, a program is "declarative" if it is written in a purely functional programming language, logic programming language, or constraint programming language. The phrase "declarative language" is sometimes used to describe all such programming languages as a group, and to contrast them against imperative languages.
These two definitions overlap somewhat. In particular, constraint programming and, to a lesser degree, logic programming, focus on describing the properties of the desired solution (the *what*), leaving unspecified the actual algorithm that should be used to find that solution (the *how*). However, most logic and constraint languages are able to describe algorithms and implementation details, so they are not strictly declarative by the first definition.
Constraint Logic Programming as a declarative modeling and procedural programming environment is increasingly realized as an effective tool for decision support systems [4, 5, 6]. CLP is suitable for Decision Support Systems (DSS) because [1, 5]:

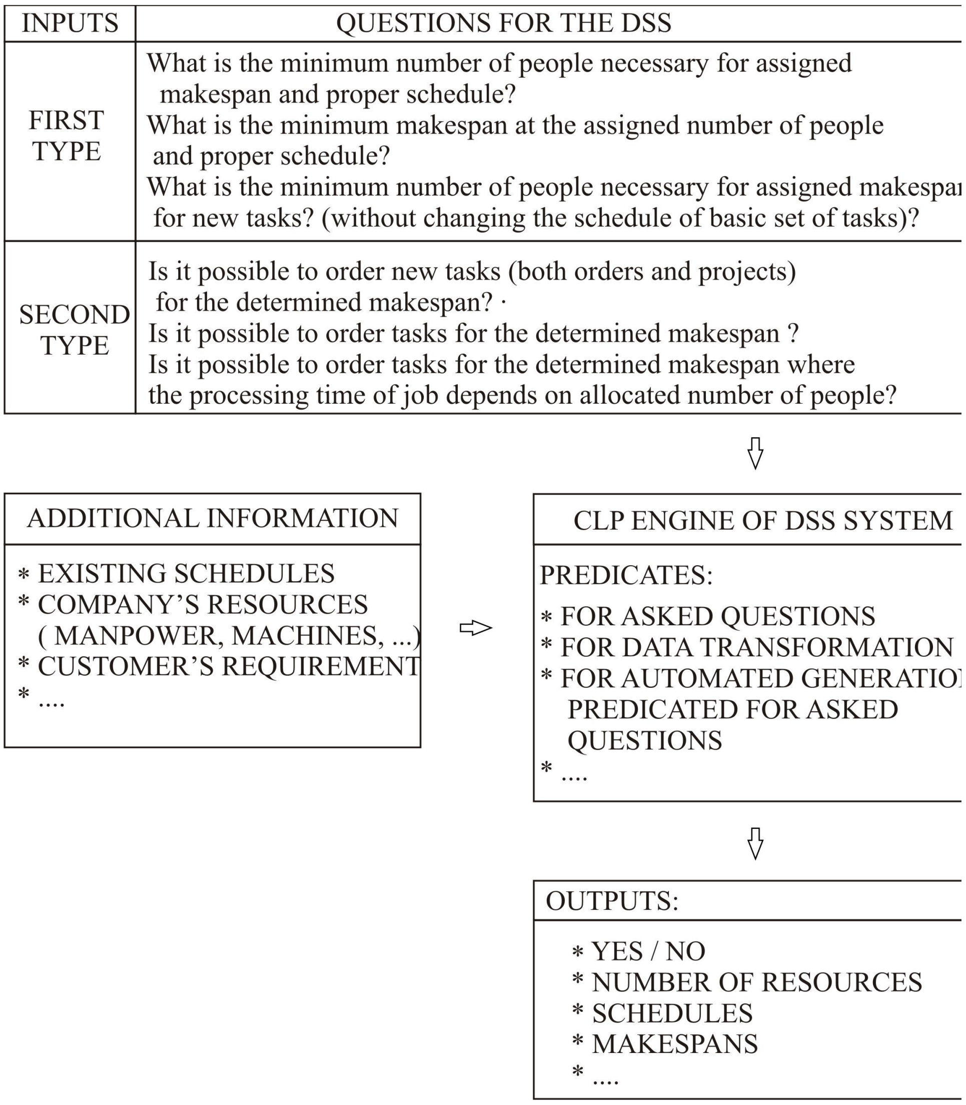
- CLP is a very good tool for the development of knowledge base that has expertise and experience represented in terms of logic, rules and constraints. This tool allows the knowledge base to be built in an incremental and accumulating way (it is suitable for ill-structured or semi-structured decision analysis problems).
- Constraints naturally represent decisions and their inter-dependencies. Decision choices are explicitly modeled as the domains of constraint variables.
- CLP can serve as a good integrative environment for the decision analysis that has different kinds of model.
- Decision analysis requires a number of computational facilities which this tool can provide.

4 CONCEPT OF DSS BASED ON DECLARATIVE PROGRAMMING FOR SCHEDULING PROBLEMS

The presented in (3) advantages and possibilities of declarative programming environment for decision support make it interesting for decision support in SMEs. The decision support system for production scheduling has been presented as an example of implementation of DSS with declarative programming. Building decision support system for scheduling, covering a variety of production organization forms, such as job-shop, flow-shop, project, multi-project etc., is especially interesting. The following assumptions were adopted in order to design the presented scheduling processes of the decision support system (see Fig.2):

- The system should possess data structures in relational model that make its use possible in different production organization environments
- The system should make it possible to schedule the whole set of tasks simultaneously, and after a suitable schedule has been found, it should be possible to add a new set of tasks later, and to find a suitable schedule for both sets without the necessity to change initial schedules.
- The system should regard:

    - Additional (external) resource types apart from machines, e.g. people, tools, etc.
    - Temporary inaccessibility of all resource types.
    - The processing times dependent on the starting time of jobs, allocated additional resources, etc.
- The decisions of the systems are the answers to appropriate questions formed as CLP predicates.



INPUTS	QUESTIONS FOR THE DSS
FIRST TYPE	What is the minimum number of people necessary for assigned makespan and proper schedule? What is the minimum makespan at the assigned number of people and proper schedule? What is the minimum number of people necessary for assigned makespa for new tasks? (without changing the schedule of basic set of tasks)?
SECOND TYPE	Is it possible to order new tasks (both orders and projects) for the determined makespan? · Is it possible to order tasks for the determined makespan ? Is it possible to order tasks for the determined makespan where the processing time of job depends on allocated number of people?

ADDITIONAL INFORMATION

* EXISTING SCHEDULES
* COMPANY'S RESOURCES (MANPOWER, MACHINES, ...)
* CUSTOMER'S REQUIREMENT
* ....

CLP ENGINE OF DSS SYSTEM

PREDICATES:

* FOR ASKED QUESTIONS
* FOR DATA TRANSFORMATION
* FOR AUTOMATED GENERATIO PREDICATED FOR ASKED QUESTIONS
* ....

OUTPUTS:

* YES / NO
* NUMBER OF RESOURCES
* SCHEDULES
* MAKESPANS
* ....

Fig.2 Concept of DSS based on declarative programming for scheduling problems

The range of the decisions made by the system depends on data structures and asked questions. Thus, the system is very flexible as it is possible to ask all kinds of questions (write all kinds of predicates). In this version of DSS the questions which can be asked are the following:

- What is the minimum number of people necessary for the assigned makespan and proper schedule? *opl_g(_,C)*
- What is the minimum makespan at the assigned number of people and proper schedule? *opc_g(L,_)*
- Is it possible to order new tasks (both orders and projects) for the determined makespan? *szu_g(L,C)*
- What is minimum makespan at the assigned number of people for new tasks? *opc_g(L,_)*
- What is the minimum number of people necessary for the assigned makespan for new tasks? (without changing the schedule of basic set of tasks) *opl_g(_,C)*
- Is it possible to order tasks for the determined makespan ? *opc_g(_,L)*
- Is it possible to order tasks for the determined makespan where the processing time of job depends on the allocated number of people? *opc_g(L,C)*

These questions are just examples of questions that the present system can be asked. New questions are new predicates that need to be created in CLP environment. Two types of questions are asked in the system:

- About the existence of the solution (eg., is it possible to carry out a new task in the particular time?, etc.)
- About a particular kind of the solution: find a suitable schedule fulfilling the performance index, find the minimum scheduling length-makespan, find the minimum number of people to carry out the task, etc.

5 ASP FRAMEWORK OF DSS WITH DELCARATIVE PROGRAMMING

We propose ECL^{i}PS^e [9] as a platform to decision support in scheduling problems. ECL^{i}PS^e is a software system - based on the CLP paradigm - for the development and deployment of constraint programming applications. It is also ideal for developing aspects of combinatorial problem solving, e.g. problem modeling, constraint programming, mathematical programming, and search techniques. Its wide scope makes it a good tool for research into hybrid problem solving methods. ECL^{i}PS^e comprises several constraint solver libraries, a high-level modeling and control language, interfaces to third-party solvers, an integrated development environment and interfaces for embedding into host environment. The ECL^{i}PS^e programming language is largely backward-compatible with Prolog and supports different dialects. It provides, however, an extended set of basic data types (byte strings, unlimited precision integer and rational numbers, double precision floats and double precision intervals).
Data structures were designed in such a way that they could be easily used to decision problems in a variety of scheduling environments, which is job-shop, flow-shop, project or multi-project. The obtained flexibility resulted from the use of relational data model. The implementation framework is shown in fig.3. All structures of DSS where implemented in XML. XML has initially designed for the exchange electronic information and documents. Now, XML is becoming the standard for data exchange among distributed applications components or co-operating applications and systems. The most widely supported technologies

for describing the schema of XML are Documents Type Definitions (DTDs) and XML-schema [10]. The DTDs files for DSS data structures have been presented in fig. 4 and fig. 5. With the use of XML, communication and information exchange can be established regardless of the underlying storage platform. However, different applications, environments and systems that communicate using XML have to transform XML to underlying information model, which is usually a relational DBMS (Database Management System). The implementation and used tools for the above DSS system are suitable and useful for the ASP model. In the ASP model the application software resides on the vendor's system and is accessed by users through a web browser using HTML or by special purpose client software provided by the vendor. Custom client software can also interface to these systems through XML APIs. These APIs can also be used where integration with in-house systems is required.

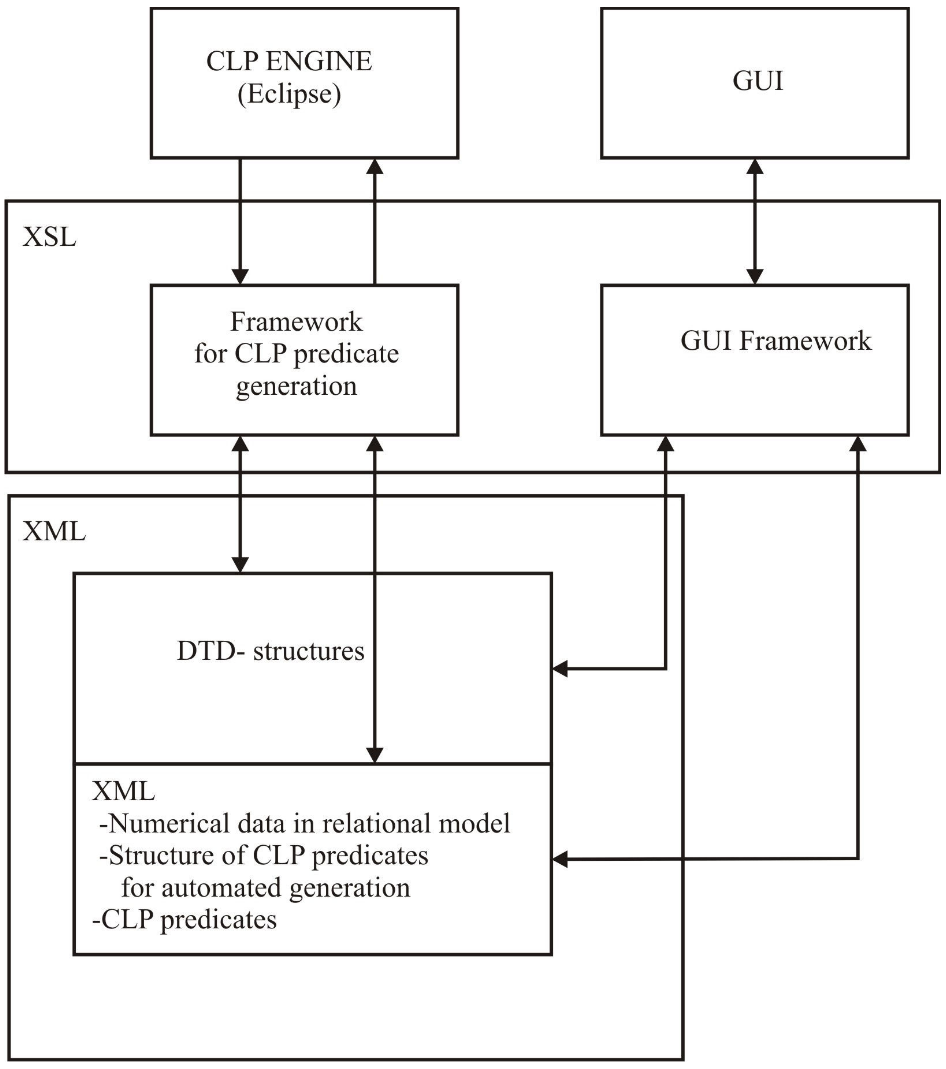


Fig. 3 Implementation framework of DSS

```
<!ELEMENT typ_operacji EMPTY >
<!ELEMENT operacja EMPTY >
<!ELEMENT operacje (typ_operacji+,operacja+) >
<!ATTLIST typ_operacji kod_typu_o ID #REQUIRED>
<!ATTLIST typ_operacji opisCDATA #REQUIRED >
<!ATTLIST operacja kod_o ID #REQUIRED >
<!ATTLIST operacja nazwa CDATA #REQUIRED >
<!ATTLIST operacja kod_typu_o IDREF #REQUIRED>
```

Fig. 4 DTD file for description of task (job).

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT przydzial EMPTY>
<!ELEMENT kolejność EMPTY>
<!ELEMENT przy_zas EMPTY>
<!ELEMENT technologia (predykaty,zasoby,operacje, przydzial+, kolejnosc+,przy_zas+)>
<!ATTLIST przydzial kod_o IDREF #REQUIRED>
<!ATTLIST przydzial kod_m IDREF #REQUIRED>
<!ATTLIST przydzial czas CDATA #REQUIRED>
<!ATTLIST przydzial typ CDATA #REQUIRED>
<!ATTLIST kolejnosc kod_o_p IDREF #REQUIRED>
<!ATTLIST kolejnosc kod_o_d IDREF #REQUIRED>
<!ATTLIST przy_zas kod_o IDREF #REQUIRED>
<!ATTLIST przy_zas kod_z IDREF #REQUIRED>
<!ATTLIST przy_zas ile CDATA #REQUIRED>
<!ATTLIST technologia predykat IDREF #REQUIRED>
<!ENTITY % zas SYSTEM "zasoby.dtd">
<!ENTITY  % oper SYSTEM "operacje.dtd">
<!ENTITY  % pre SYSTEM "predykat.dtd">
%zas;
%oper;
%pre;
<!ENTITY  lis_operacja SYSTEM "operacje.xml">
<!ENTITY  lis_zasoby SYSTEM "zasoby.xml">
<!ENTITY  lis_pred SYSTEM "predykat.xml">
```

Fig. 5 DTD file of relationship among entities of DSS

6 ILLUSTRATIVE EXAMPLES

After the complete implementation of the DSS into ECL^{i}PS^e and XML environments, computation experiments were carried out. The job-shop scheduling problem with manpower resources (Example 1) and project –building house (Example 2) were considered.

The proposed illustrative examples cover a wide range of scheduling problems encountered in the SMEs. The examples are selected in such a way that they how two extremely different forms of production organization; repetitive production in the job-shop environment and the unique production including the project. The presented methodology makes solving scheduling problems possible also in indirect methods of production organization. Moreover, the examples are larded with problems of constrained resources (e.g. manpower, specialized machines, etc.) and the dependence of particular jobs processing time on the amount of the allocated resources, for instance.

6.1 Example 1- the job shop scheduling.

In the classical scheduling theory job processing times are constant (Example_1a). However, there are many situations where processing time of a job depends on the starting time of the job in queue or the amount of allocated additional resources (e.g. people) (Example_1b) etc. The parameters of computational examples are presented in table 1. The job data structures are shown in Fig. 6a and Fig. 6b.

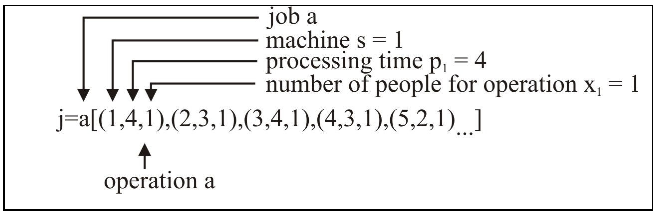

job a
machine s = 1
processing time $p_1 = 4$
number of people for operation $x_1 = 1$
j=a[(1,4,1),(2,3,1),(3,4,1),(4,3,1),(5,2,1)...]
operation a

Fig.6a. Description of task (job) data structure for job-shop computational example (Example_1a)

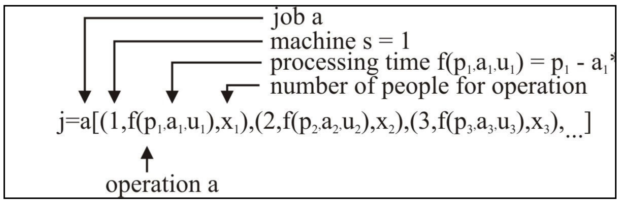

job a
machine s = 1
processing time $f(p_1,a_1,u_1) = p_1 - a_1$
number of people for operation
j=a[(1,f(p_1,a_1,u_1),x_1),(2,f(p_2,a_2,u_2),x_2),(3,f(p_3,a_3,u_3),x_3),...]
operation a

Fig.6b. Description of task (job) data structure for job-shop computational example (Example_1b)-the processing times depend on allocated number of people.

The computational Example_1b was carried out with job processing times of jobs dependent on the allocated additional resource (people). The parameters of this example are presented in tab.1 without processing times and number of allocated people. The processing time is a function of allocated people $f(p_j,a_j,u_j)$ fig. 7.

$$f(p_j,a_j,u_j) = p_j - a_j*u_j \text{ and } f(p_j,a_j,u_j) > 0 \text{ and } a_j = 1$$

where :

- p_j - processing time from Example_1.
- u_j- additional number of allocated people.
- a_j - acceleration factor

Fig 7 Processing time for Example_2b

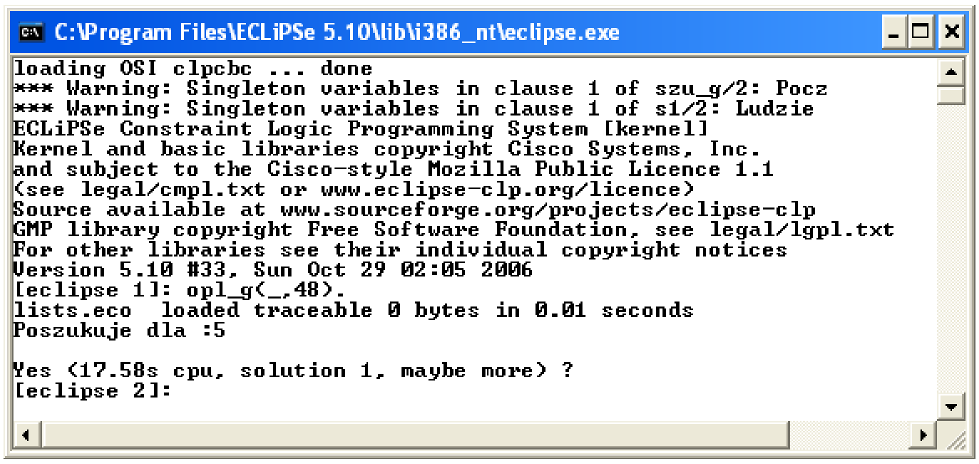
Table 1. Parameters of computational examples (Example_1a, Example_1b)

j∈{a,b,c,d,e,f,g}, o∈{a,b,c,d,e,f}, s∈{s1,s2,s3,s4,s5,s6}
j=a [(4,1,2), (4,2,1), (3,3,1), (8,4,1), (3,5,1),(3,6,1)]
j=b [(2,5,1), (3,4,1), (5,3,1), (4,2,1), (4,1,2),(8,6,1)]
j=c [(8,1,2), (3,5,1), (4,2,1), (4,3,1), (8,4,1),(4,6,1)]
j=d [(4,1,2), (4,2,1), (5,3,1), (3,4,1), (3,5,1),(3,6,1)]
j=e [(3,5,1), (3,4,1), (3,3,1), (4,2,1), (2,1,2),(4,6,1)]
j=f [(6,5,1), (4,4,1), (6,3,1), (6,2,1), (4,1,2),(3,6,1)]
j=g [(4,3,1), (3,5,1), (4,1,2), (5,2,1), (4,4,1),(2,6,1)]

The time constrained resources availability and manpower limitation were modeled as a list of parameters. The resource occupancy can be interpreted as a job with the fixed start times for all their operations and fixed manpower requirements. For the computational example the following questions (write following predicates) were asked (see section 4):

- *opl_g(_,48)* (see fig.6)
- *opc_g(5,90)* (see fig.7,8)
- *szu_g(5,60)* (see fig. 9)
- *szu_g(5,45)* (see fig. 10).

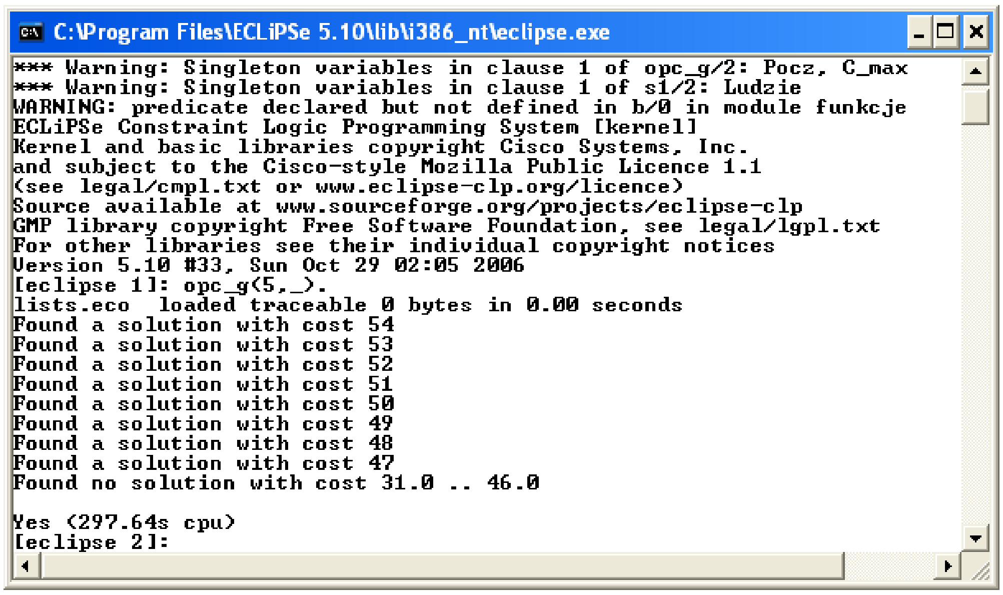
Computation experiments were started on the computer PIV 1,4 GHz, RAM 512 under Windows XP.



```
C:\Program Files\ECLiPSe 5.10\lib\i386_nt\eclipse.exe
loading OSI clpcbc ... done
*** Warning: Singleton variables in clause 1 of szu_g/2: Pocz
*** Warning: Singleton variables in clause 1 of s1/2: Ludzie
ECLiPSe Constraint Logic Programming System [kernel]
Kernel and basic libraries copyright Cisco Systems, Inc.
and subject to the Cisco-style Mozilla Public Licence 1.1
(see legal/cmpl.txt or www.eclipse-clp.org/licence)
Source available at www.sourceforge.org/projects/eclipse-clp
GMP library copyright Free Software Foundation, see legal/lgpl.txt
For other libraries see their individual copyright notices
Version 5.10 #33, Sun Oct 29 02:05 2006
[eclipse 1]: opl_g(_,48).
lists.eco  loaded traceable 0 bytes in 0.01 seconds
Poszukuje dla :5

Yes (17.58s cpu, solution 1, maybe more) ?
[eclipse 2]:
```

Fig. 8 Answer to the question implemented in predicate *opl_g(_,48)*–result L_{min}=5 (Example_1a)



```
C:\Program Files\ECLiPSe 5.10\lib\i386_nt\eclipse.exe
*** Warning: Singleton variables in clause 1 of opc_g/2: Pocz, C_max
*** Warning: Singleton variables in clause 1 of s1/2: Ludzie
WARNING: predicate declared but not defined in b/0 in module funkcje
ECLiPSe Constraint Logic Programming System [kernel]
Kernel and basic libraries copyright Cisco Systems, Inc.
and subject to the Cisco-style Mozilla Public Licence 1.1
(see legal/cmpl.txt or www.eclipse-clp.org/licence)
Source available at www.sourceforge.org/projects/eclipse-clp
GMP library copyright Free Software Foundation, see legal/lgpl.txt
For other libraries see their individual copyright notices
Version 5.10 #33, Sun Oct 29 02:05 2006
[eclipse 1]: opc_g(5,_).
lists.eco   loaded traceable 0 bytes in 0.00 seconds
Found a solution with cost 54
Found a solution with cost 53
Found a solution with cost 52
Found a solution with cost 51
Found a solution with cost 50
Found a solution with cost 49
Found a solution with cost 48
Found a solution with cost 47
Found no solution with cost 31.0 .. 46.0

Yes (297.64s cpu)
[eclipse 2]:
```

Fig. 9 Answer to the question implemented in predicate *opc_g(5,_)*–result C_{max}=47 (Example_1a)

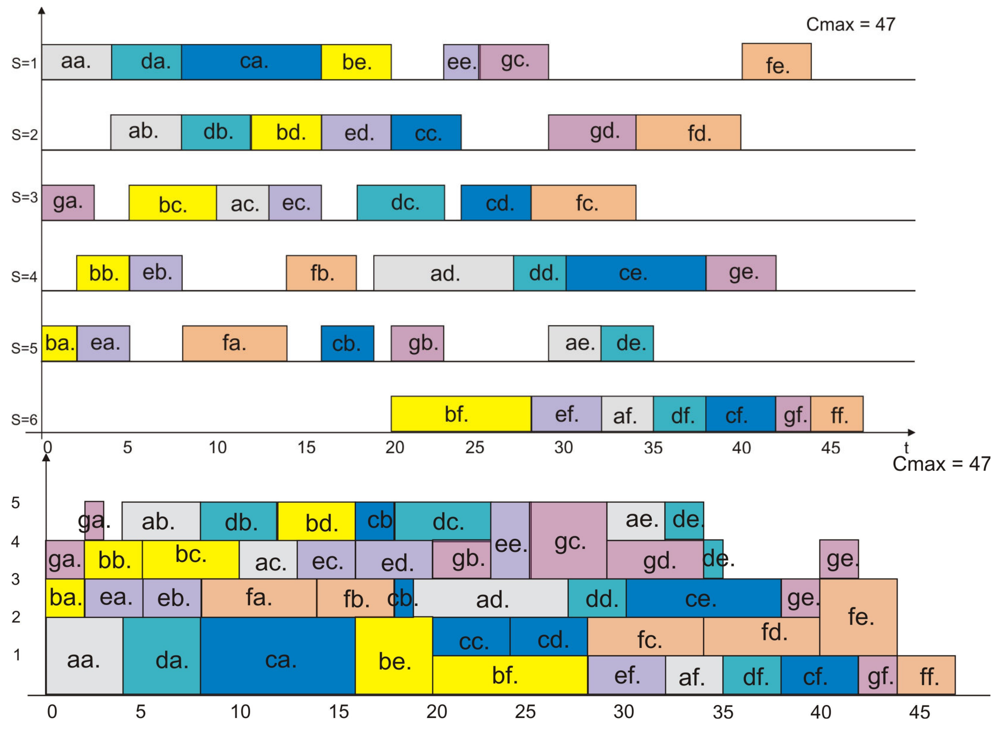
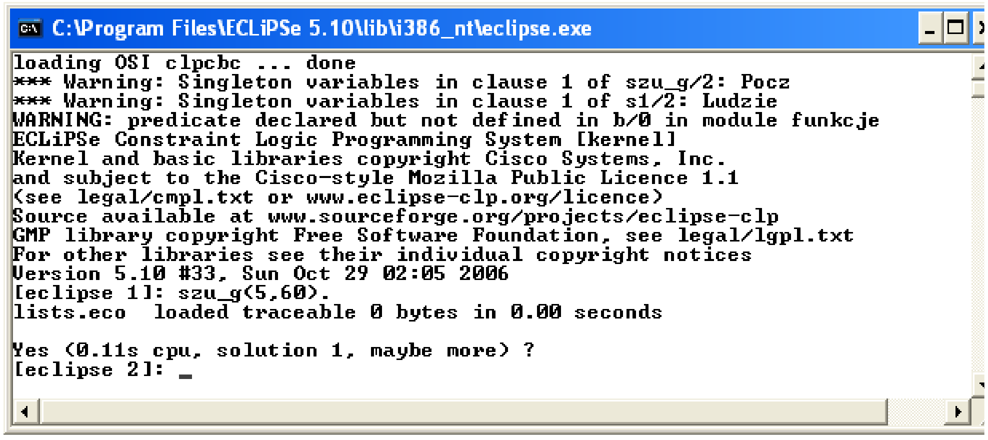


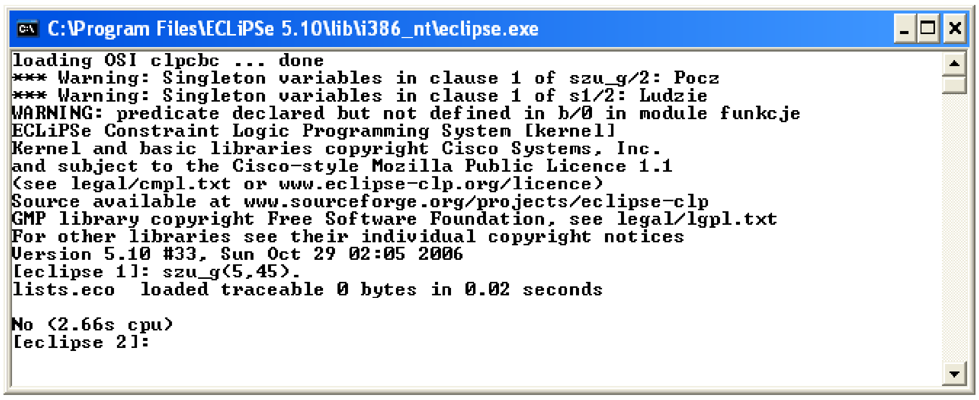
Fig. 10 Gantt's chart for decision from fig.9 (Example_1a)



```
loading OSI clpcbc ... done
*** Warning: Singleton variables in clause 1 of szu_g/2: Pocz
*** Warning: Singleton variables in clause 1 of s1/2: Ludzie
WARNING: predicate declared but not defined in b/0 in module funkcje
ECLiPSe Constraint Logic Programming System [kernel]
Kernel and basic libraries copyright Cisco Systems, Inc.
and subject to the Cisco-style Mozilla Public Licence 1.1
(see legal/cmpl.txt or www.eclipse-clp.org/licence)
Source available at www.sourceforge.org/projects/eclipse-clp
GMP library copyright Free Software Foundation, see legal/lgpl.txt
For other libraries see their individual copyright notices
Version 5.10 #33, Sun Oct 29 02:05 2006
[eclipse 1]: szu_g(5,60).
lists.eco   loaded traceable 0 bytes in 0.00 seconds

Yes (0.11s cpu, solution 1, maybe more) ?
[eclipse 2]: _
```

Fig. 11 Answer to the question implemented in predicate *szu_g(5,60)* – Yes (Example_1a)

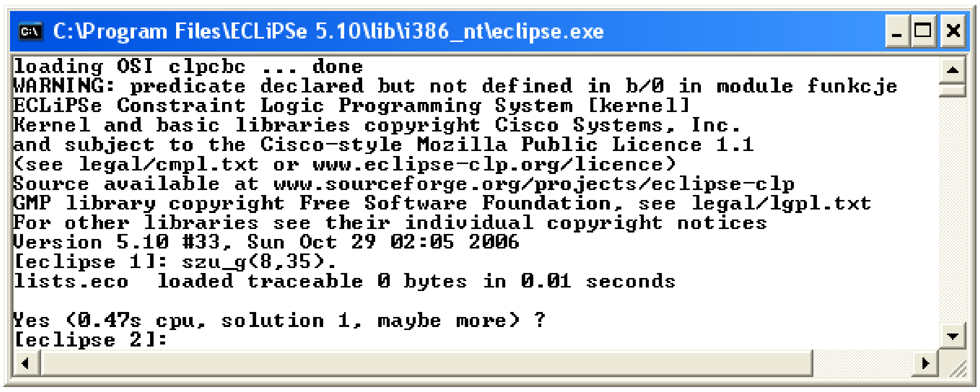


```
loading OSI clpcbc ... done
*** Warning: Singleton variables in clause 1 of szu_g/2: Pocz
*** Warning: Singleton variables in clause 1 of s1/2: Ludzie
WARNING: predicate declared but not defined in b/0 in module funkcje
ECLiPSe Constraint Logic Programming System [kernel]
Kernel and basic libraries copyright Cisco Systems, Inc.
and subject to the Cisco-style Mozilla Public Licence 1.1
(see legal/cmpl.txt or www.eclipse-clp.org/licence)
Source available at www.sourceforge.org/projects/eclipse-clp
GMP library copyright Free Software Foundation, see legal/lgpl.txt
For other libraries see their individual copyright notices
Version 5.10 #33, Sun Oct 29 02:05 2006
[eclipse 1]: szu_g(5,45).
lists.eco   loaded traceable 0 bytes in 0.02 seconds

No (2.66s cpu)
[eclipse 2]:
```

Fig. 12 Answer to the question implemented in predicate *szu_g(5,45)* – NO (Example_1a)

The results of computational experiments (Example_1b) have been shown at Fig. 13, Fig.14.



```
loading OSI clpcbc ... done
WARNING: predicate declared but not defined in b/0 in module funkcje
ECLiPSe Constraint Logic Programming System [kernel]
Kernel and basic libraries copyright Cisco Systems, Inc.
and subject to the Cisco-style Mozilla Public Licence 1.1
(see legal/cmpl.txt or www.eclipse-clp.org/licence)
Source available at www.sourceforge.org/projects/eclipse-clp
GMP library copyright Free Software Foundation, see legal/lgpl.txt
For other libraries see their individual copyright notices
Version 5.10 #33, Sun Oct 29 02:05 2006
[eclipse 1]: szu_g(8,35).
lists.eco   loaded traceable 0 bytes in 0.01 seconds

Yes (0.47s cpu, solution 1, maybe more) ?
[eclipse 2]:
```

Fig. 13 Answer to question implemented in predicate *szu_g(8,25)* – Yes (Example_1b)

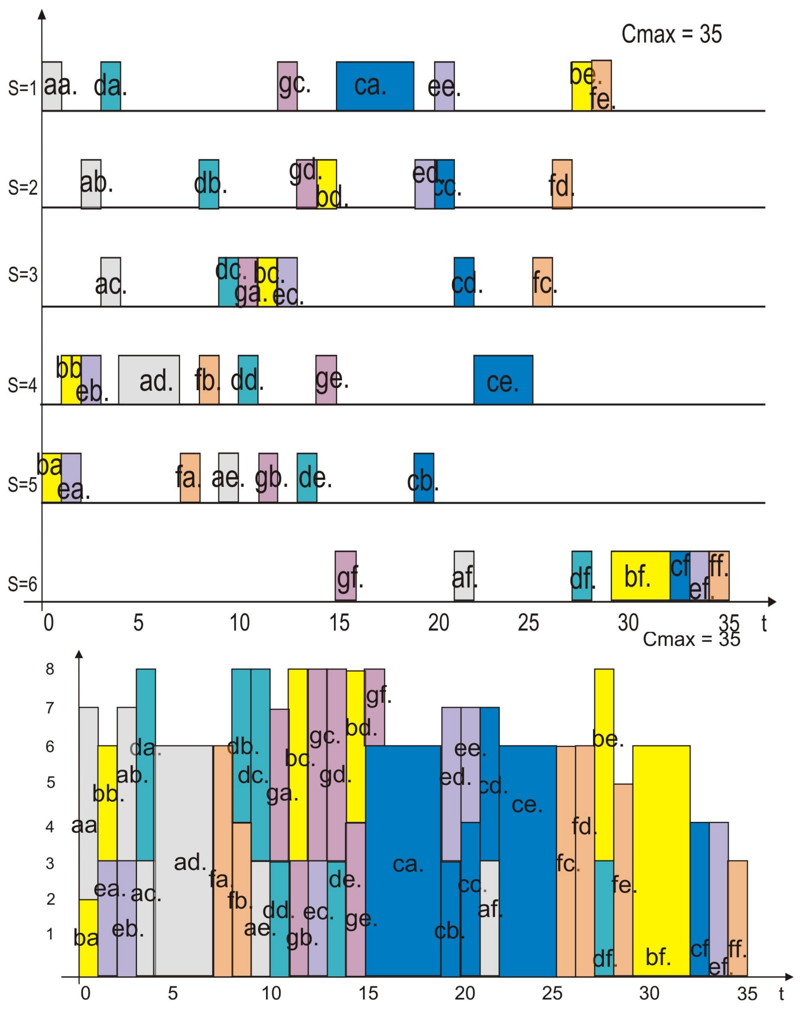


Fig 14.Gantt's charts for decision from fig. 13 (Example_1b).

6.2 Example 2 –building house-project

A typical modern-day project has a variety of complications not considered in the original PERT/CPM methodology. There are three particular situations:

- You may be able to accelerate the completion of a project by speeding up or "crashing" some of the activities in the project.
- Your ability to finish a project quickly is hindered by limited resources (e.g., two activities that might otherwise be done simultaneously, in fact have to be done sequentially because they both require a crane and you have only one crane on site).
- How long it takes to do each activity is a random variable.

In table 2, we list the activities involved in a simple, but nontrivial, project of building a house. An activity cannot be started until all of its predecessors are finished. The network activity for this project has been shown in fig.15. To solve this example the DSS with declarative programming (section 4) was used. In this example the processing times of activities depend on allocated manpower resource.

Table 2 Parameters of Example_2

On.	Activity Time	Min_MAN	Max_MAX	Name of activity
1	10	2	2	Dig Basement
2	12	4	6	Pour Foundation
3	6	1	3	Pour Basement
4	6	2	3	Install Floor Joists
5	6	1	3	Install Walls (ext)
6	4	2	8	Install Rafters
7	4	2	4	Install Walls (int)
8	4	2	2	Install Roof
9	16	4	8	Install Windows, Doors (ext)
10	12	4	8	Install Networks
11	12	6	8	Interior Plastering
12	4	2	4	Painting (int)
13	6	2	3	Finish Interior
14	18	6	9	Finish Terrace
15	4	2	4	Garden Arrangement
16	18	6	12	Exterior Plastering

MIN_MAN –	minimum manpower for activity
MAX_MAN –	maximum manpower for activity

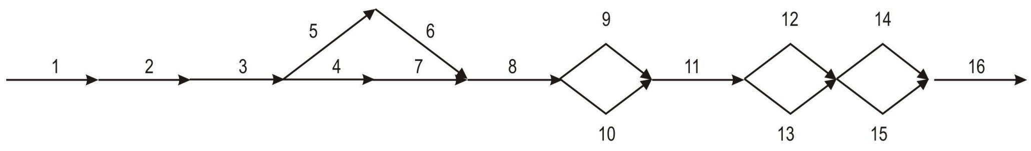
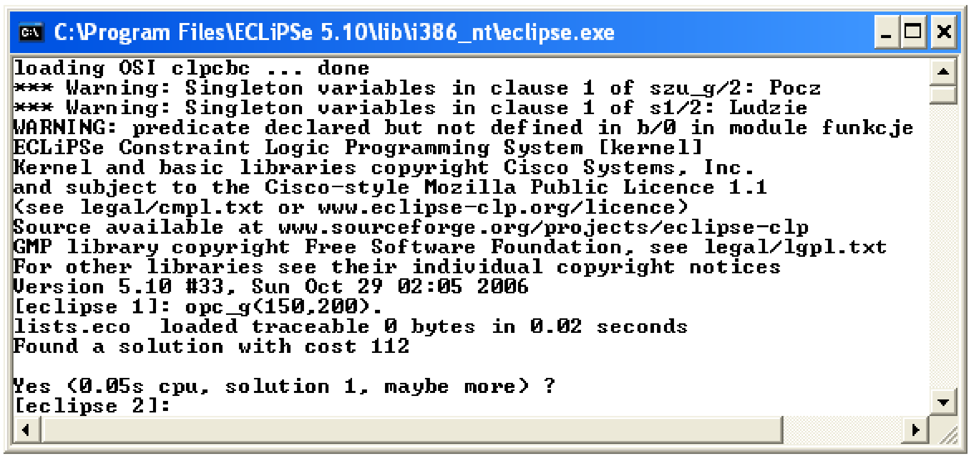


Fig. 15 Activity network

For the computational example the following questions (write following predicates) were asked (see section 4):

- *opc_g(150,200)* (see fig. 16).
- *opc_g(5,400)* (see fig. 18).
- *opc_g(7,200)* (see fig. 19, 20).
- *opc_g(12,200)* (see fig. 21, 22, 23) - processing times of jobs dependent on the allocated additional resource (people).

Computation experiments were started on the computer PIV 1,4 GHz, RAM 512 under Windows XP.



```
loading OSI clpcbc ... done
*** Warning: Singleton variables in clause 1 of szu_g/2: Pocz
*** Warning: Singleton variables in clause 1 of s1/2: Ludzie
WARNING: predicate declared but not defined in b/0 in module funkcje
ECLiPSe Constraint Logic Programming System [kernel]
Kernel and basic libraries copyright Cisco Systems, Inc.
and subject to the Cisco-style Mozilla Public Licence 1.1
(see legal/cmpl.txt or www.eclipse-clp.org/licence)
Source available at www.sourceforge.org/projects/eclipse-clp
GMP library copyright Free Software Foundation, see legal/lgpl.txt
For other libraries see their individual copyright notices
Version 5.10 #33, Sun Oct 29 02:05 2006
[eclipse 1]: opc_g(150,200).
lists.eco  loaded traceable 0 bytes in 0.02 seconds
Found a solution with cost 112

Yes (0.05s cpu, solution 1, maybe more) ?
[eclipse 2]:
```

Fig. 16 Answer for the question implemented in predicate *opc_g(150,200)* – Yes (Example_2)

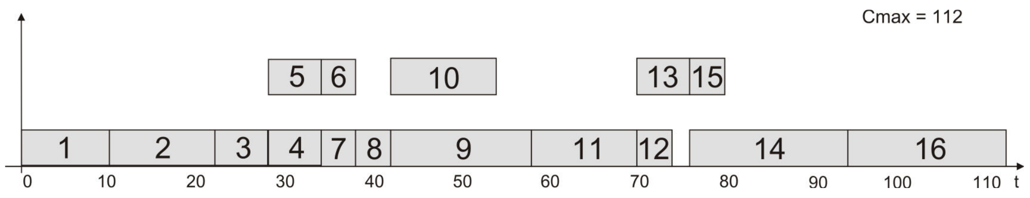
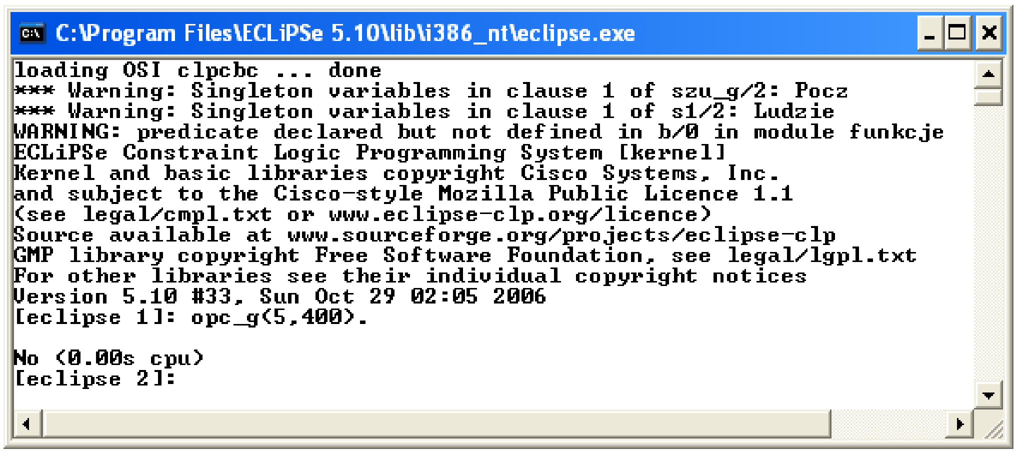


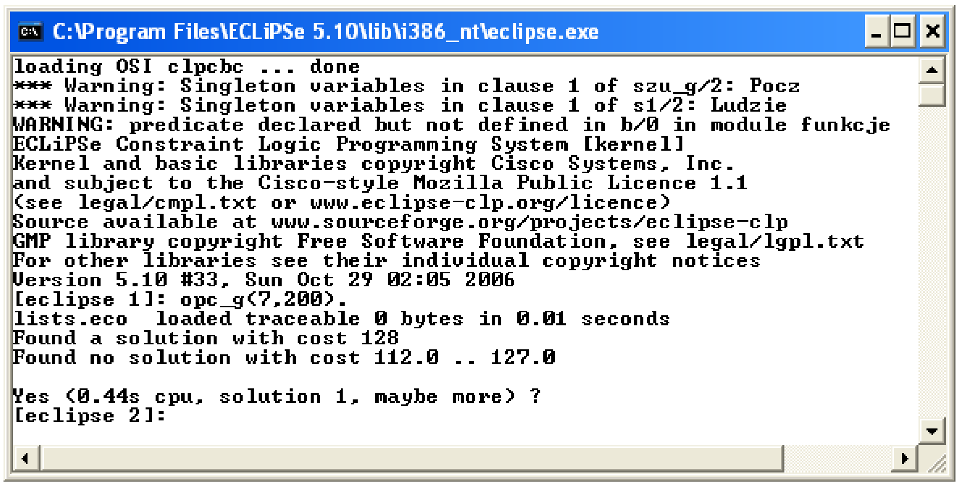
Fig. 17 Gantt's charts for decision from fig. 16 (Example_2).



```
loading OSI clpcbc ... done
*** Warning: Singleton variables in clause 1 of szu_g/2: Pocz
*** Warning: Singleton variables in clause 1 of s1/2: Ludzie
WARNING: predicate declared but not defined in b/0 in module funkcje
ECLiPSe Constraint Logic Programming System [kernel]
Kernel and basic libraries copyright Cisco Systems, Inc.
and subject to the Cisco-style Mozilla Public Licence 1.1
(see legal/cmpl.txt or www.eclipse-clp.org/licence)
Source available at www.sourceforge.org/projects/eclipse-clp
GMP library copyright Free Software Foundation, see legal/lgpl.txt
For other libraries see their individual copyright notices
Version 5.10 #33, Sun Oct 29 02:05 2006
[eclipse 1]: opc_g(5,400).

No (0.00s cpu)
[eclipse 2]:
```

Fig. 18 Answer for the question implemented in predicate opc*_g(5,400)* – No (Example_2)



```
loading OSI clpcbc ... done
*** Warning: Singleton variables in clause 1 of szu_g/2: Pocz
*** Warning: Singleton variables in clause 1 of s1/2: Ludzie
WARNING: predicate declared but not defined in b/0 in module funkcje
ECLiPSe Constraint Logic Programming System [kernel]
Kernel and basic libraries copyright Cisco Systems, Inc.
and subject to the Cisco-style Mozilla Public Licence 1.1
(see legal/cmpl.txt or www.eclipse-clp.org/licence)
Source available at www.sourceforge.org/projects/eclipse-clp
GMP library copyright Free Software Foundation, see legal/lgpl.txt
For other libraries see their individual copyright notices
Version 5.10 #33, Sun Oct 29 02:05 2006
[eclipse 1]: opc_g(7,200).
lists.eco   loaded traceable 0 bytes in 0.01 seconds
Found a solution with cost 128
Found no solution with cost 112.0 .. 127.0

Yes (0.44s cpu, solution 1, maybe more) ?
[eclipse 2]:
```

Fig. 19 Answer for the question implemented in predicate opc*_g(7,200)* – Yes (Example_2)

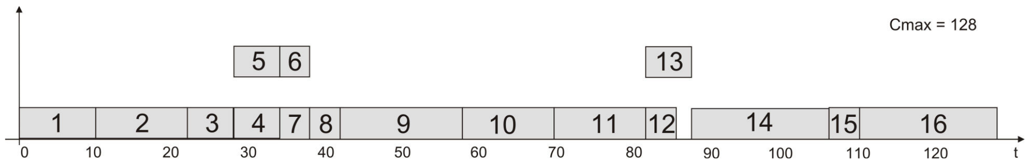
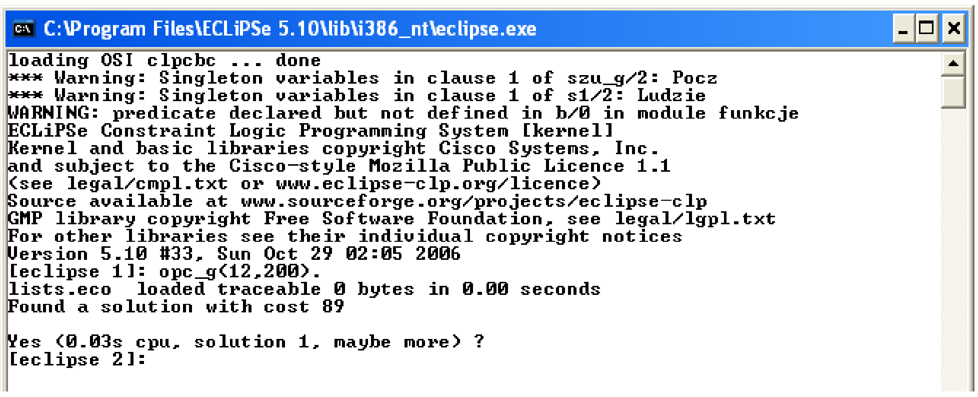


Fig. 20 Gantt’s charts for decision from fig. 19 (Example_2).



```
loading OSI clpcbc ... done
*** Warning: Singleton variables in clause 1 of szu_g/2: Pocz
*** Warning: Singleton variables in clause 1 of s1/2: Ludzie
WARNING: predicate declared but not defined in b/0 in module funkcje
ECLiPSe Constraint Logic Programming System [kernel]
Kernel and basic libraries copyright Cisco Systems, Inc.
and subject to the Cisco-style Mozilla Public Licence 1.1
(see legal/cmpl.txt or www.eclipse-clp.org/licence)
Source available at www.sourceforge.org/projects/eclipse-clp
GMP library copyright Free Software Foundation, see legal/lgpl.txt
For other libraries see their individual copyright notices
Version 5.10 #33, Sun Oct 29 02:05 2006
[eclipse 1]: opc_g(12,200).
lists.eco   loaded traceable 0 bytes in 0.00 seconds
Found a solution with cost 89

Yes (0.03s cpu, solution 1, maybe more) ?
[eclipse 2]:
```

Fig. 21 Answer for the question implemented in predicate opc*_g(12,200)*–Yes (Example_2)

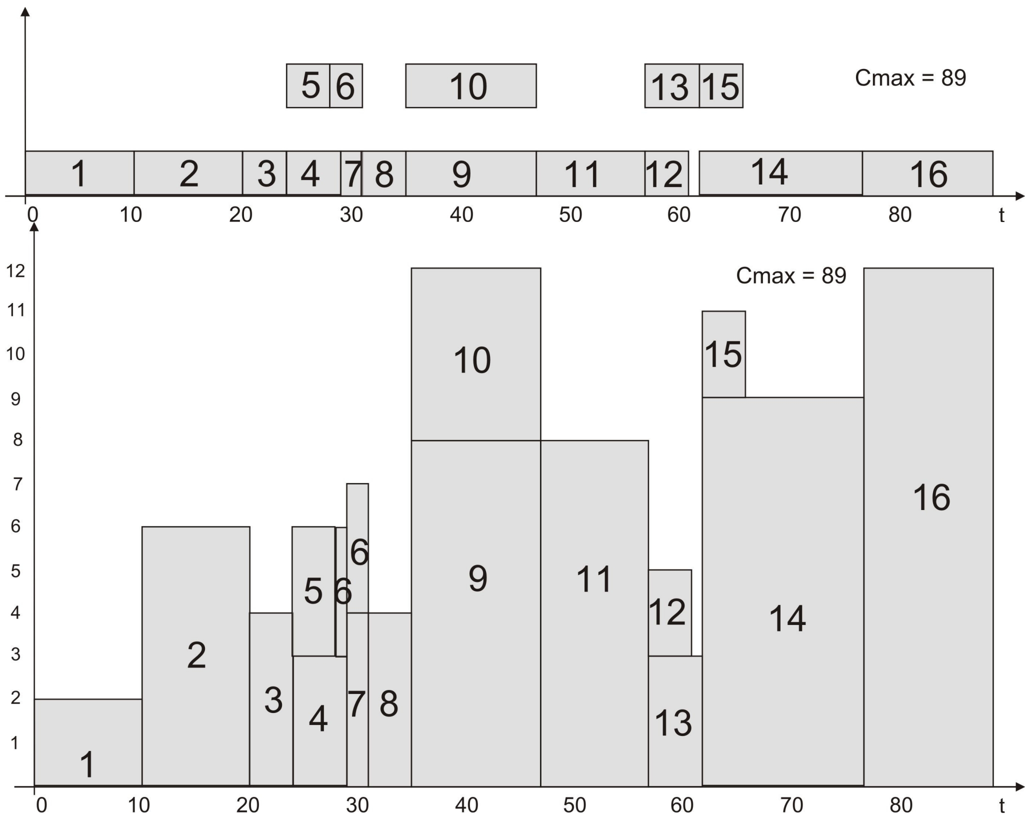
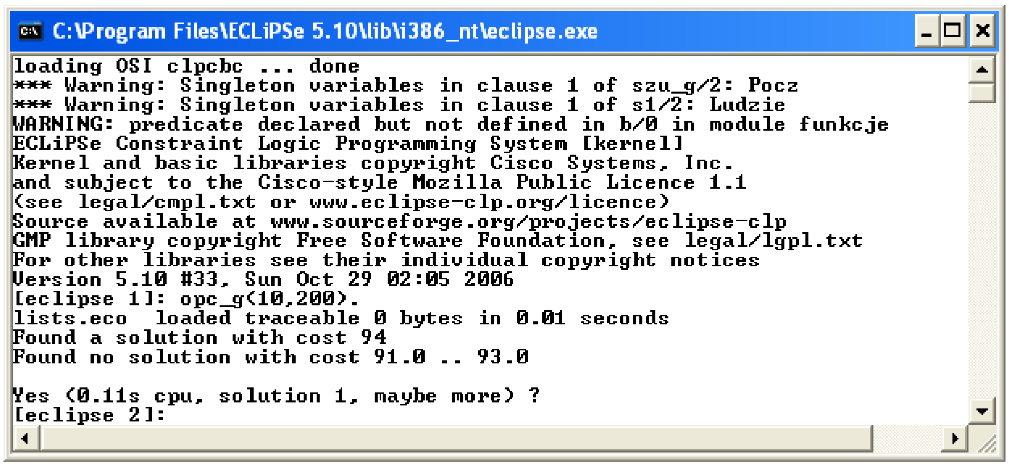


Fig 22.Gantt's charts for decision from fig. 21 (Example_2).



```
loading OSI clpcbc ... done
*** Warning: Singleton variables in clause 1 of szu_g/2: Pocz
*** Warning: Singleton variables in clause 1 of s1/2: Ludzie
WARNING: predicate declared but not defined in b/0 in module funkcje
ECLiPSe Constraint Logic Programming System [kernel]
Kernel and basic libraries copyright Cisco Systems, Inc.
and subject to the Cisco-style Mozilla Public Licence 1.1
(see legal/cmpl.txt or www.eclipse-clp.org/licence)
Source available at www.sourceforge.org/projects/eclipse-clp
GMP library copyright Free Software Foundation, see legal/lgpl.txt
For other libraries see their individual copyright notices
Version 5.10 #33, Sun Oct 29 02:05 2006
[eclipse 1]: opc_g(10,200).
lists.eco   loaded traceable 0 bytes in 0.01 seconds
Found a solution with cost 94
Found no solution with cost 91.0 .. 93.0

Yes (0.11s cpu, solution 1, maybe more) ?
[eclipse 2]:
```

Fig. 23 Answer to the question implemented in predicate opc_*g(10,200)*–Yes (Example_2)

7. CONCLUSIONS

Proposing declarative environments (CLP, SQL) for the building and implementing of the decision support system for scheduling, as well as suggesting the ASP model for the provision of the application service to SMEs seem to be as interesting and promising approach. Above all, declarative environments offer splendid possibilities of modeling and simple implementation of the decision support system. Advantages of this solution include easy decision support for scheduling of literally any method of production organization, and also considering additional resource constraints, for instance, manpower or specialized machines, and their effect on the way the jobs are performed, e.g., shortening the processing time.

Further, the application of the ASP model provides not only the decision support system but also both the know-how and the follow -up service to the enterprise.

The proposed approach can be considered as a contribution to scheduling problems with external/additional resources applied in SMEs, where this kind of resources can have influence on production and delivery schedules. That is especially important in the context of cheap, fast and user friendly decision support in SMEs. Great flexibility of the proposed approach (ASP model) and practically unlimited possibilities of asking questions through creating predicates cannot be overestimated. What is more, the whole decision system can be built in one modeling and programming declarative environment and deliver to customer by ASP, which lower costs and adds to the solution effectiveness.

REFERENCES

1. LIAO S.Y., WANG H.Q., LIAO L.J.: *An extended formalism to constraint logic programming for decision analysis*, Knowledge-based Systems 15, 2002 , pp 189-202.
2. PEABODY G.: *Interpath connects Customer to SAP Applications via World-class Communications*, Data Center and Support Infrastructure Aberdeen Group 2000.
3. LACITY M.C., HIRSCHHEIM L, WILLCOCKS: Realizing outsourcing expectations, Information Systems Management 11(4), 1994, pp 7-18.
4. BISDORFF R., LAURENT S. *"Industrial linear optimization problem solved by constraint logic programming",* European Journal of Operational Research 84 (1), 1995, pp 82-95.
5. LAMMA E., MELLO P., MILANO M. "*A distributed constrained-based scheduler"*, Artificial Intelligence in Engineering 11,1997, pp 91-105.
6. LEE H.G., LEE G. Yu., "Constraint logic programming for qualitative and quantitative constraint satisfaction problems", Decision Support Systems 16 (1), 1996, pp 67-83.
7. RYU U. Young ."*Constraint logic programming framework for integrated decision supports"* Decision Support Systems 22, 1998, pp 155-170.
8. BENNETT Ch. TIMBRELL G.: *Application Services Providers: Will They Succeed* ?, Information Systems Frontiers, pp 195 – 211, Kluwer Academic Publishers, 2000.
9. http://www.cs.kuleuven.ac.be/
10. "XML Schema part 0", W3C Working Draft, 2000 (www.w3.org/TR/xmlschema-0)