PACOS: an integrated planning and control system

Andrea Caccamese

ing. C. Olivetti & C. S.p.A.
Olivetti Systems & Networks
Ivrea (Italy)

October 15, 1987
1. Introduction

This paper presents the functionalities and the architecture of PACOS.

PACOS is a system designed to support the management to plan and control activities in a very complex industrial environment, that of a company involved in several critical hardware/software projects.

Unlike other similar packages available in the market place, PACOS presents some unique features.

First, it is an integrated system, not just a single tool with several functionalities.

It manages a data base in which information is gathered and maintained.

A set of tools can then access the data base to produce different views as needed by different managers, for different purposes, in different stages of the production process.

Second, it is a cooperative system, in that it reverses the ordinary approach of personal planning used by most of the packages currently available.

The data base is built with the cooperation of the entire organisational structure, in that the plans must be built top-down, this forcing any organisational level to plan its activities, at the correct abstraction level.

Data base manipulation is made with a complete set of consistency checks that can not be avoided and that ensure that schedule, resources and cost constraints are maintained consistent at any level of the structure.

Double link dependencies are allowed, this avoiding loss of control in a complex organisation where several process are independently running towards different goals, while needing synchronisation in some critical ocurrences.

Third, it is a complete system, in that it does not only include well known management tools like a Gantt charts reporter and a Pert analyser.

Specific tools for administrative cost control, based on the needs of the company, were added.

Moreover, emphasis is given to software cost estimation, in that the system includes implementations of SLIM and COCOMO cost models.

PACOS is thus much more than a planning and control system; it is a proposal for a structural organisation of a modern company dealing with both a turbulent market and with a very dynamic technological environment.

October 15, 1987
2. Generalities and motivation

The production of software became in the near past a very complex process needing an adequate counterpart on the side of project control.

This applies especially in the case of companies involved in many different projects at the same time. The need to develop different products covering different marketing sectors is one of the reasons. It is often necessary to offer product lines in place of simple, not correlated, products.

This raises the need of project control techniques able to keep track of a large number of separate items.

Moreover, the process is made more complex because in many cases the project does not only involve software, but also hardware developments.

In these cases, there are dependencies between separate items in the process that cannot be avoided, since hardware and software together form the product to be delivered, then a tight coordination between separate production teams has to be ensured.

This raised the need of project control techniques able to keep under control a large number of dependencies between many distinct items.

There is another reason that contributes to make the production process even more complex, in that many companies currently are switching from the simple, well-understood, role of production centers, to the new role of system integrators.

In other words, there is now a tendency to privilege buy on make, i.e. in many cases the company prefers to buy from an external supplier one or more components of a complex product.

This happens for different reasons, some of which being time constraints evaluation, specific know-how owned by an external producer, and similar.

This solution of the make-or-buy dilemma is in many cases also applied without reference to external producers.

Different groups inside the same big company may be in total charge of different products, this involving all the life cycle, including sales.

So, each independent group could decide to buy a component from another group, belonging to the same company, which is under separate development, and to include it in the product itself.

This raises the need for project control techniques able to cope with a complex network of dependencies not only related to external events, but also involving internals (inside the company) events.
The above needs simply add to the ordinary needs of any industrial process, i.e. an efficient accounting control organisation, and the need to track the project progress.

3. Features

PACOS is a system designed to cover the complete set of the above needs.

In particular, PACOS offers:

a) an integrated solution.
   The features of PACOS solve the problem to standardize the different project views offered by different tools, to give to the user the right level of visibility of the project status, and to share the same data for different reasons;

b) a cooperative solution.
   The features of PACOS force the involvement of each team member in the activities of planning. The work breakdown structure is prepared top-down by all of the people responsible, each at his level of visibility and responsibility, with the use of tools allowing de-centralisation of planning activities;

c) a systematic solution.
   PACOS is indeed not a collection of separate, independent, tools, but on the contrary it is a system of cooperating tools, sharing the same data base for different reasons ranging from the simple project status control to project cost modeling.

3.1. Integrated solution

This is one of the main features of PACOS, in that it was felt that the use of different tools for project control in different groups was a major reason of chaos. Each tool produced its own output, based on its own specific functionalities and conventions. This led very rapidly to situations in which it was very difficult for people that were not directly involved in the project to recognise what project the plan was referring to.

PACOS establishes a common basis of understanding between different groups, not only forcing to use the same tools, hence the same output in the same format, but also forcing to share the same name data base for all the projects that are handled by the system.

There is another feature of PACOS that contributes to provide an integrated solution, in that the different tools belonging to the system all share the same plan data base, in which all the relevant information to project control is contained.

October 15, 1987
This means that the plan database not only contains data like activities name, responsible name and schedule (which are needed to keep track of the project progress), but also contains data like the current and planned expenses for internal and external resources (consultancies, hardware material, travelling expenses), which are needed to exploit accounting control activities.

This information is all stored in the plan database, this allowing and forcing separate control activities to share the same data, thus ensuring consistent views of the project in different groups.

3.2. Cooperative solution

In a certain sense PACOS reverses the classical approach of planning.

In the classical approach, a unique staff group was always considered to be necessary to handle the job to draw plans, perform project advance control and report to the management.

The functionalitites of PACOS can not be used to substitute a staff group, whose presence is still necessary, but indeed it provides to such a group a powerful tool to improve its degree of confidence on the accuracy and level of update of the plans, as well as its responsiveness to the top management requests.

PACOS requests to draw the plans in a top-down approach, this involving that all the members of the organisation that are involved in a project are in turn requested to provide the right information at their level of visibility.

The plan database is built with the cooperation of the entire organisational structure, this forcing any individual to plan the activities at its level.

For example, when the process to build a plan is started, the top management will be asked to provide a description, schedule and other information associated to the activities as seen at that macroscopic level.

This is what is ususally called master plan.

The organisation could survive with this level of description, however to exploit an effective project progress and administrative control it is necessary to proceed to further refinements.

These refinement are supported by PACOS, which allows to refer to the already planned activities as a basis to draw a plan at a further level of refinement.

In this phase, a full set of consistency checks is implemented.

This mechanism can be iterated to reach the requested level of refinement, each time requiring the involvement of the people directly interested in the activity, and each time guaranteeing consistency.

For this reason, the approach of PACOS may be considered a cooperative planning.

October 15, 1987
This approach does not only apply to plans drawn from scratch, but it applies as well to modifications made to existing plans.

This leads to a state in which any planned activity does belong to a plan of a higher level of abstraction. As a result, no activity may be scheduled if it is not part of a complete project. As a result, too, modifications to the plan structure, i.e. adding or removing activities, schedules modifications and so on, are subject always to consistency checks. The failure of a consistency check may be overcome only through an adequate replanning. PACOS, then enforces the adjacent level of the organisational structure to discuss and to find a solution able to fulfill the organisation's requirements.

There is another main feature of this approach. PACOS allows, as many other systems do, the establishment of dependencies between activities. In PACOS, however, this is allowed only when the conditioning activity is already planned. The link can then be established only between activities contained in the data base. This avoids the phenomenon of plans drawn on the basis of unplanned activities. Moreover, since PACOS establishes double links, this avoids to cancel activities that are considered crucial for the completion of other projects. The system checks for this occurrence and refuses the cancellation.

The cooperative approach of PACOS is also stressed by the decentralisation of the data entry subsystem, as it will be explained below.

2.3. Systemistic solution

PACOS offers a solution to the company problems which is much more than a collection of standard useful tools. In the market place it is possible to find a huge set of planning control tools available. The differences between them are in the interface, easiness of use, more than in the functionalities.

It is easy, therefore, to find packages able to handle projects, subdivided in sub-projects, offering import/export functionalities to other, more complex, planning tools. However, all these packages deal with a standard, widespread interpretation of what are the needs of a planning system, so they usually are limited to the preparation of Gantt or bar charts, or to the execution of more or less complicated PERT analysis.

None of these packages can give an answer to some specific needs of the company. This is somewhat in the logic, since some needs are too specific to be incorporated in a generally available
package. For example, the accounting procedures vary from one company to another, as well the laws to which they are conformant can vary from one country to another. PACOS then incorporates some tools specifically designed to meet specific needs of the company, and, what is more important, these tools become part of an integrated system, so sharing data with other "standard" tools, avoiding the need for dangerous and expensive data duplication inside different environments.

There is another point in which the systemistic approach given to PACOS becomes apparent. Some "standard" planning tools do not cover completely the needs of a real company. For example, PERT analysis is executed by most tools available assuming a hierarchical structure of the graphs. The structure of the graphs which come from the organisation of a real company may be, however, not strictly hierarchical. This implies the need for suitable PERT analysers which, again, become an integrated part of PACOS.

PACOS is then much more than a collection of tools, but indeed it offers an integrated environment made of a set of tools, some of them simply being equivalent to what is currently available in the market place, some of them specifically designed to cope with the company needs.

4. Structure

PACOS is logically subdivided into two logical subsystems:

a) a data entry subsystem. This subsystem is oriented to collect the planning data from the community of the users. Note that the planning data are to be considered as planning proposal, since they need the validation of the system to be inserted in the data base;

b) a data enquiry subsystem. This subsystem is oriented to extract from the plan data base the information which is needed. It is then oriented to a restricted community of users, that of the planning staff. However, it may also be accessed by the community of the users, if the appropriate authorization level is provided.

October 15, 1987
4.1. Data entry subsystem
user
+
V
+------------------+
| UDE             |
|                 |
|                 |
+------------------+

(LAN/WAN)
V
+------------------+
| plan           |
| data           |
| file          | 1--+
|               |
+------------------+

=================================================
planning staff system
=================================================

+------------------+
| UDE             |
|                 |
V
+------------------+
| plan           |
| data file      |
+------------------+

+------------------+
| NUCLEUS         |
|                 |
|                 |
+------------------+

+------------------+
| plan           |
| data           |
| base          |
+------------------+

figure 1

October 15, 1987
Figure 1 represents the structure of the data entry-subsystem, which is composed of two parts:

a) **NUCLEUS**

b) **UDE**

4.1.1. **NUCLEUS**

4.1.1.1. **PLAN DATA BASE ARCHITECTURE**

**NUCLEUS** is essentially a database manager and implements storing and retrieval of records representing planned activities. For each planned activity there is a record in the plan database. The structure of the database is similar to a tree, in that activities are hierarchically organised. Each activity possesses a unique father, that represents the higher level of abstraction plan of which it is a refinement. Each activity may have sons. If they exist, they represent the planning refinement of the activity itself, otherwise the activity itself is the lowest level of abstraction available on the plan.

The format of the records is described in figure 2 (Pascal-like description is used):
const
MAXDESCR = 50;
MAXLUNG = 14;
MAXRESP = 4;
MAXLIV = 8;
MAXREL = 5;

type
casseDiCosto = (prodottiMateriali, prestazioni, viaggi,
progettiEsterni, consulenzaCostoFisso);
tipolavoro = (interno, tempoMateriali);
peso = (globale, locale, normale, baseline);
stato = (orario, ritardo, anticipo, completo);
frase = packed array [1..MAXDESCRI] of char;
parola = packed array [1..MAXLUNG] of char;

personale = record
  nm: parola;
  pc: 0..100;
  tv: tipolavoro
end;

responsabili = array [1..MAXRESP] of personale;
codici = record
  linguaggio: (hllc, lllc, hlll, llll);
  ambiente: (interattivo, batch);
  available: 0..255
end;
driversNames = (reply, data, cplx, time, stor, virt, turn,
  acap, aexp, pcap, vexp, lexp, modp, tool, sced);
driversValues = (extrahigh, veryhigh, high, nominal, low,
  verylow);
calendario = record
  gg: 1..31;
  mm: 1..12;
  aa: 80..99
end;
periodo = record
  di: calendario;
  df: calendario
end;

October 15, 1987
preven = record
  minss: longinteger;
  exss: longinteger;
  maxss: longinteger
end;
consun = record
  ss: longinteger;
end;
costDrivers = array [driversNames] of driversValues;
modelli = record
  dp: preven;
  dc: consun;
  dr: costDrivers;
  cod: codici
end;
legame = record
  percorso: array [1..MAXLIV] of integer;
  lag: integer
end;
condizionatori = array [1..MAXREL] of legame;
condizionati = array [1..MAXREL] of legame;
descrCosto = record
  cst: longinteger
end;
costi = record
  g_u_interni: longinteger;
  g_u_tempoMat: longinteger;
  altrE: array[classeDiCosto] of descrCosto
end;
costiAssociati = record
  budget: costi;
  ultimaRilevazione: costi;
  cumulativo: costi;
  consuntivo: costi
end;
attivita = record
  name: frase;
  rs: responsabili;
  dt: periodo;
  pl: modelli;
  condDa: condizionatori;
  condVs: condizionati;
  ct: costiAssociati;
  ot: peso;

October 15, 1987
The meaning of the various fields is as follows:

classiDiCosto qualifies the cost of resources associated to an activity. It is used by accounting tools to divide the costs according to the company procedures;

tipoLavoro qualifies the cost of the human effort associated to an activity. It is used by accounting tools to divide the costs according to the company procedures;

peso qualifies the role of the activity in the global economy of a plan. It is used by planning tools to execute specific checks on the most important activities;

stato qualifies the current status of the activity in front of the planned schedule. It is used by planning tools to execute specific checks on the most important activities;

frase this is the actual name of the activity. It is up to the planner to correctly describe the activity with the available characters;

parola this field is used to store the name of the responsible of the activity. It is up to the planner to correctly describe the responsible with the available characters;

personale this is a complex field fully describing the responsible of the activity. Besides the *nm* field (the name), *pc* indicates the percentage of the individual that is spent on the activity while *ty* is a qualification of the cost of the human effort involved;

responsabili this field describes the responsibilities of the activity;

codici its fields qualify the environment in which the production is executed, and are used for software cost modelling purposes. *linguaggio* qualifies the abstraction level of the programming language involved,
ambiente qualifies the development environment which is made available to the programmer, available is a field for future use;

driversNames this field is simply the list of the names of the COCOMO model cost drivers;

driversValues this field is the set of the possible values for the COCOMO model cost drivers;

calendario represents the date associated to an activity.

gg indicates the day of the month, mm indicates the month of the year and aa indicates the year;

periodo indicates the schedule for the activity.

di indicates the start date, df indicates the ending date;

pr even this record indicates the estimated size of the project represented by the activity, expressed in source code lines.

minss indicates the minimum estimated size, exss indicates the average estimated size and maxss the maximum estimated size.

It is used for software cost models purposes;

consun simply indicates the counted number of source code instructions associated to the activity.

This value refers to the real number of source code lines that are counted when the activity is terminated and is used for software cost model purposes;

costDrivers this is an array describing the cost drivers associated to the activity, for software cost model purposes;

modelli this field fully describes all the information which is needed to use cost models;

legame this field is used to represent both dependencies from other activities and dependencies that were established by other activities on the current activity.

percorso locates the activity in the plan data base tree structure.

lag represents the kind of dependency. A positive lag involves an overlap relationship between the schedules of the current activity and the one indicated by Percorso. A negative lag involves a separation relationship between the two schedules.

October 15, 1987
A null lag involves a stop-start relationship. All lags are expressed in working days;

**condizionatori**  this field contains the full description of the activities which are conditioning the current activity;

**condizionati**  this field contains the full description of the activities which are dependent on the current activity;

**descrCosto**  describes the cost of resources associated to the activity;

**costi**  this field is a full description of the costs that may be associated to the activity.

- _g _u interno_ represents the number of man-days of internal personnel,
- _g _u tempoMat_ represents the number of man-days of external (consultants) personnel working in the project group,
- _altri_ represents costs other than the man-days, that may be associated to the activity;

**costiAssociati**  this field is a full description of all the costs associated to the activity.

- _budget_ expresses the costs allocated at the moment of budgetary evaluation,
- _ultimaRilevazione_ expresses the costs collected at the moment of the last accounting control check. These are executed at fixed periods, usually monthly.
- _cumulativo_ expresses the cumulation of the ultimaRilevazione figures from the beginning.
- _consuntivo_ expresses the actual costs as measured at the end of the activity itself. It will be the same as the last value assumed by cumulativo;

**attivita**  this is the record describing the activity itself.

- _name_ is the identification of the activity,
- _rs_ identifies the responsible(s) designed for the activity,
- _dt_ identifies the schedule associated to the activity,
- _pl_ represents the information needed by cost models,
- _condDa_ expresses the range of activities on which the activity is dependent,
- _condVs_ represents the range of activities depending on the activity,
- _ct_ represents the costs associated to the activity.

October 15, 1987
activity.

\( q_t \) qualifies the importance of the activity in the general economy of the plan.

\( s_t \) represents the current progress of the activity.

4.1.1.2. CONSISTENCY CHECKS

NUCLEUS does not only perform the job to store and retrieve records in the data base, but it moreover executes a full set of mandatory consistency checks whenever a plan data base modification is requested.

A plan modification data base is involved whenever creation of new records is requested, deletion of records is requested, or whenever a modification to records is requested.

This means that the consistency checks are active whenever a new project is planned, whenever a project has to be cancelled, or whenever modifications to the planned data of a project is requested.

The consistency checks fall into the following classes:

a) identification checks.

Whenever an activity is addressed, NUCLEUS checks that it actually does exist in the plan data base. This is a pre-requisite to address the record on which modifications are to be applied.

In the case of activities to be added, the check applies to the father of the activity addressed.

In the case of activities to be deleted, the check applies to any dependency expressed in the activity, or in its possible sons.

This protects dependent sub-projects from a sudden cancellation of the external project on which they are dependent. This is feasible only after appropriate replanning of the dependent sub-projects.

These checks maintain the data base consistent from the point of view of the organisation, in that every planned activity belongs to a corporate schema, with no room for activities out of it;

b) schedule checks.

Whenever an activity is addressed, NUCLEUS checks that the schedules associated to the activity are consistent with those of the father activity, in that the start and end date of the activity must be contained in the interval defined by the start and end date of the father activity.

Moreover, the check is also active on the scheduled dates of all the possible sons of the activity itself.

The check is also active on the scheduled dates of

October 15, 1987
dependent activities.

These checks maintain the data base consistent from the point of view of schedules, in that any slippage of a sub-project is subject to be consistent with the planned dates of the project itself. On the other side, any schedule constraint imposed by the management (i.e. shortening the period for the project) is refused unless the subproject are accordingly replanned.

Slippage of activities are also subject to be consistent with the schedules of dependent sub-projects;

resources checks.

Whenever an activity is addressed, NUCLEUS checks that the costs associated to the activity are consistent with those associated to its father.

The budgetary value expressed in any activity is distributed between the son activities on the basis of their planned values, and its value is decreased accordingly. This is done provided that the budgetary value associated to the father is sufficient.

Cancellation of an activity automatically adds its budgetary costs value to the father values.

This check maintains the data base consistent from the point of view of accounting, in that the budgetary cost limits can not be exceeded, unless explicitly replanned at the superior level. This allows to control that any project that is started is also provided with all the budget that is needed.

4.1.2. UDE

The Universal Data Entry utility (from now on called UDE), is the unified interface for the end-user willing to do planning proposals.

It refers to a common and shared data base containing the names of the activities that are already planned and therefore present in the plan data base.

This utility uses this data base to force to plan in a top-down fashion. The user is driven with a series of menu allowing to walk through the structure of the names of the already planned activities, then localize the correct name of the activity to be modified or removed.

The user is also driven to walk through the structure of the names of the already planned activities even in the case that a new activity has to be planned. In this case, the user is forced to choose the father of its activity, then correctly positioning its plan as a refinement level of an already existing plan.

This philosophy is also applied when the user wishes to express dependencies from an activity.
In this case, he is driven to choose among the names of already planned activities, thus avoiding the phenomenon of plans missing the correct level of integration between separate sub-projects.

UDE allows the user to enter all the data contained in the format of the plan database records, which is described below. In entering the values for this data, the menu-driven approach is followed. The user is always provided with a series of choices of possible values, apart in the cases where the information is intrinsically left to the ens-user itself.

UDE is made available on a variety of systems, allowing the users community to use their ordinary computer resources also to make plan proposals.
4.2. Data enquiry subsystem
end-user system

user

exec PLANNER
exec SCRIBA
exec OLISLIM
exec OLICOMO
exec ACCOUNTER
exec OLIPERT

(LAN/WAN)

user

+--------+
+--------+
+--------+
+--------+
+--------+
+--------+
+--------+
+--------+
+--------+
+--------+
+--------+

planning staff system

+--------+
+--------+
+--------+
+--------+
+--------+
+--------+
+--------+
+--------+
+--------+
+--------+
+--------+
+--------+

NUCLEUS

+--------+

plan

data
base

DATA-ENQUIRY

October 15, 1987

October 15, 1987
Figure 3 represents the data enquiry subsystem, which is composed of the following parts:

a) NUCLEUS
b) PLANNER
c) SCRIBA
d) OLISLIM
e) OLICOMO
f) ACCOUNTER
g) OLIPERT

Note that all the tools of the data enquiry subsystem may be accessed both from the system belonging to the planning staff and from user systems, provided that a suitable LAN/WAN connection and the appropriate authorization level are available.

4.2.1. NUCLEUS

The functionalities of NUCLEUS in this case are restricted to the location of the information requested by the various enquiry tools. NUCLEUS offers read-only functionalities to the enquiry tools, those involving no consistency check.

4.2.2. PLANNER

PLANNER is a tool conceived to deal with dependencies. It is able to list and keep track of all the dependencies involved in an activity. The operations could be recursive, i.e. also indirectly dependent activities may be listed. Detection of deadlocks is part of the job of PLANNER.

What-if functionalities are offered.

4.2.3. SCRIBA

SCRIBA is a Gantt reporter. It allows a Gantt or bar chart representation of the plan, made on the information gathered in the plan data base. Activities may be shown in various fashions, for example selection on the basis of responsible names is allowed. Also selection on the basis of schedule is allowed. SCRIBA grants both local (i.e. restricted to the activity identified) and global (i.e. recursive) operativity. SCRIBA offers also the possibility to draw histograms of the percentage of global involvement of individuals.

October 15, 1987
What-if functionalities are offered.

4.2.4. OLISLIM

OLISLIM is an implementation of the SLIM cost model. It is supported by NUCLEUS in that the needed data on the total amount of source line code necessary for cost modeling is obtained walking through the tree corresponding to the activity to inspect, collecting the values of source code statements present at the lowest levels, and cumulating them. In this case, the finest the decomposition of the project in sub-projects (eventually elementary), the best the evaluation of the source statement number, and the best the modeling.

The technology constant involved by SLIM is contained in a special file associated to the subtree in the plan database.

OLISLIM offers the possibility to calculate the technology constant from historical data (i.e. terminated projects), thus allowing to heuristically deduce it from a terminated plan, and to put this value in the plan related to the activity to be modeled.

OLISLIM offers trade-off region handling, man-power and schedules curve handling.

What-if functionalities are offered.

4.2.5. OLICOMO

OLICOMO is an implementation of the detailed COCOMO cost model.

It is supported by NUCLEUS in that the needed data on the total amount of source line code necessary for cost modeling is obtained walking through the tree corresponding to the activity to inspect, collecting the values of source code statements present at the lowest levels, and cumulating them.

In this case, the finest the decomposition of the project in sub-projects (eventually elementary), the best the evaluation of the source statement number, and the best the modeling.

The cost drivers involved by COCOMO are also got from the lowest level of decomposition of the activity. Also in this case, the finest the decomposition, the best the accuracy.

OLICOMO offers the full capabilities of COCOMO model, with resources and schedules handling.

What-if functionalities are offered.

October 15, 1987
4.2.3.1. Cost models integration

The advantages of having integrated the data needed by cost models in the plan data base are apparent.

First, the accuracy of data is ensured, since estimates of the cost model parameters are done even at the lowest level of abstraction, thus producing reliable values.

Second, any modification involved by project redefinition can be immediately reflected in the values collected for the models, so the management may be provided with new estimates in real-time with re-planning.

Third, the plan data base constitutes an enormous data base of projects that can be used to tune and calibrate the models.

Indeed, the past experience showed that no cost model can be used as it is in the company. Modifications, calibrations, or even a new corporate model, would be needed to reflect the real life cycle.

Any of these activities would be supported at the best working on the data contained in a corporate data base.

4.2.6. ACCOUNTER

ACCOUNTER is a tool specifically designed for use of the accounting staff.

Its purpose is to perform a quantitative analysis of the project.

It allows periodical reporting of the costs encountered in a project, and organisation of this data on the basis of cost classification.

These operations are performed with the support of NUCLEUS which provides an extract feature able to provide ACCOUNTER with cumulative values of the cost present at the bottom level of the data base.

Between its features there is also the possibility to produce a projected trend of the costs to be sustained in a project, based on the amount spent (versus the budgetary amount) and on a distribution law that may be chosen between various alternatives.

4.2.7. OLIPERT

OLIPERT is a PERT analyser able to produce schedule, cost and load analysis.

For this, it is supported by NUCLEUS, which is able to provide the dependencies information needed to carry on PERT analysis.

An interesting feature of OLIPERT is that the graphs that are analysed are quite different from the strictly hierarchical graphs taken into account by most PERT packages.

October 15, 1987
The graphs which come from the plan database are indeed hierarchical, but the possibility to set dependencies between any sub-project in the structure adds a template of diagonal links over the hierarchy. This has to be taken into account with a specific implementation of the PERT algorithms.

A graphic representation feature of the plan is also provided.

What-if functionalities are offered.

5. Hardware and software assumptions

PACOS does not make any special assumption on hardware and software support. Being conceived as a corporate system, there is the need of suitable amount of mass disc space. The distribution of both the data-entry and the data-enquiry subsystems makes it necessary to have a LAN/WAN support.

From the point of view of software, an underlying operating system with a hierarchical path name structure is needed. A suitable complex files access method, giving access to indexed files is also needed.

6. Conclusions

The company needs which oriented the design of PACOS were described and the distinctive features of the system have been addressed. Moreover, a thorough explanation of the architecture of the system was given.

It is apparent that PACOS is not only a proposal for an integrated planning and project control system, but also it involves some strategic decision of the company on its organisational structure. This is analogous to what many different companies did in the past, electing an already existent system as their corporate control system, or deciding to build their own.

At this extent it may be said that PACOS represents a proposal for an organisational structure able to be supported by a corporate decision support system. In this schema, the company organisation and the decision support system act as complementary, integrated parts.

7. Acknowledgements

The original idea of PACOS was conceived with the contribution of Renato Obermto. Many thanks also to all people that contributed to the development and the refinement of the idea, in particular.
Massimo Ancona, Alessandro Chiesi, Laura Dell'Agnello, Gabriella Dodero, Anna Rosso, and Natale Tos.

A prototype of PACOS was implemented.
Many thanks to all people that were actively involved in the work, in particular Mariangela Aime, Laura Cappello, Alessandra Maldini, Luca Reboulaz, Massimo Teruggi and Roberto Vaghi.

A special thanks to Flavio Serughetti who sponsored and supported the project.

October 15, 1987
8. References

[1] Frederick P. Brooks, Jr
The mythical man-month - Addison-Wesley - 1978

[2] Lawrence H. Putnam
A general empirical solution to the Macro Software Siz-
ing and Estimating Problem - IEEE Transactions on
Software Engineering - 1978

Software Engineering - Prentice-Hall 1979

[4] F.N. Farr
An alternative to the Rayleigh Curve Model for Software
Development Effort - IEEE Transactions on Software
Engineering - 1980

[5] Alan J. Perlis, Frederick G. Sayward, Mary Shaw
Software metrics - The MIT Press 1981

[6] Victor Basili

[7] Laura Dell’Agnello
Un modello per il ciclo di vita di grandi sistemi
software - University of Genova - Graduation Thesis in
Applied Mathematics - 1981

Software Cost Estimation: Present and Future - Software
Practice and Experience - 1981

[9] Barry W. Boehm

[10] Roger S. Pressman
Software Engineering: A Practitioner’s Approach -
McGraw-Hill 1982

[11] Laura Cappello
Modelli per il ciclo di vita del software: applicazione
alla realta’ - University of Genova - Graduation Thesis in
Applied Mathematics - 1985

[12] T. Lydon
WICOMO at Raytheon - Proceedings of the 1st
COCOMO/WICOMO Users’ Group Meeting - 1985

[13] Luca Reboulaz
Strumenti per la pianificazione e il controllo dei
costi in ambiente di produzione industriale hw/sw: parte prima - University of Torino - Graduation Thesis in
Computer Sciences - 1986

October 15, 1987
[14] Massimo Teruggi
Strumenti per la pianificazione e il controllo dei costi in ambiente di produzione industriale hw/sw:
parte seconda - University of Torino - Graduation Thesis in Computer Sciences - 1986

A comparison of SLIM and COCOMO estimates versus historical man-power and effort allocation - Proceedings of the 2nd COCOMO/WICOMO Users' Group Meeting - 1986

[16] Mariangela Aime
Determinazione del cammino critico e rappresentazione grafica PERT in un sistema di pianificazione e controllo - University of Genova - Graduation Thesis in Applied Mathematics - not yet issued

October 15, 1987
**TABLE OF CONTENTS**

(No Reference Information)

1. **Introduction**

2. **Generalities and motivation**

3. **Features**
   3.1. Integrated solution
   3.2. Cooperative solution
   3.3. Systemistic solution

4. **Structure**
   4.1. Data entry subsystem
       4.1.1. NUCLEUS
       4.1.1.1. PLAN DATA BASE ARCHITECTURE
       4.1.1.2. CONSISTENCY CHECKS
       4.1.2. UDE
   4.2. Data enquiry subsystem
       4.2.1. NUCLEUS
       4.2.2. PLANNER
       4.2.3. SCRIBA
       4.2.4. OLISLIM
       4.2.5. OLICOMO
       4.2.5.1. Cost models integration
       4.2.6. ACCOUNTER
       4.2.7. OLIFERT

5. **Hardware and software assumptions**

6. **Conclusions**

7. **Acknowledgements**

8. **References**