METHODOLOGICAL AND ARCHITECTURAL ISSUES IN THE EXPERIENCE FACTORY

Victor R. Basili Gianluigi Caldiera

Institute for Advanced Computer Studies
Department of Computer Science
University of Maryland
College Park, Maryland

Extended Abstract

The Experience Factory

The concept of experience factory as been introduced in order to institutionalize the collective learning of the organization that is at the root of continuous improvement and competitive advantage.

In the short run, the success of a software organization can be measured by its cost/performance attributes: it delivers (or updates) the needed systems generally on time and without budget overruns. In a longer run, though, we have to consider today's market characterized today by shrinking budgets and increased global competition. In the second half of the '90, the most successful organizations will probably be the ones that have been able to converge to better levels of cost and quality. Therefore, the real advantage will come from the ability of the software organization to deliver solutions that anticipate the needs of the system users and provide a real enhancement to their business or mission [Hamel and Prahalad 1991].

Research for this study was supported by NASA (Grant NSG-5123)

Within this framework, reuse of experience and collective learning cannot be left to the imagination of single, very talented, managers: they become a corporate concern like the portfolio of businesses or the company assets. The *experience factory* is the organization that supports reuse of experience and collective learning developing, updating and providing upon request to the *project organizations* clusters of competencies, that we call *experience packages*. The project organizations offer to the experience factory their products, the plans used in their development, and the data gathered during development and operation; the experience factory transforms these objects into reusable units and supplies them to the project organizations, together with specific support made of monitoring and consulting (Figure 1).

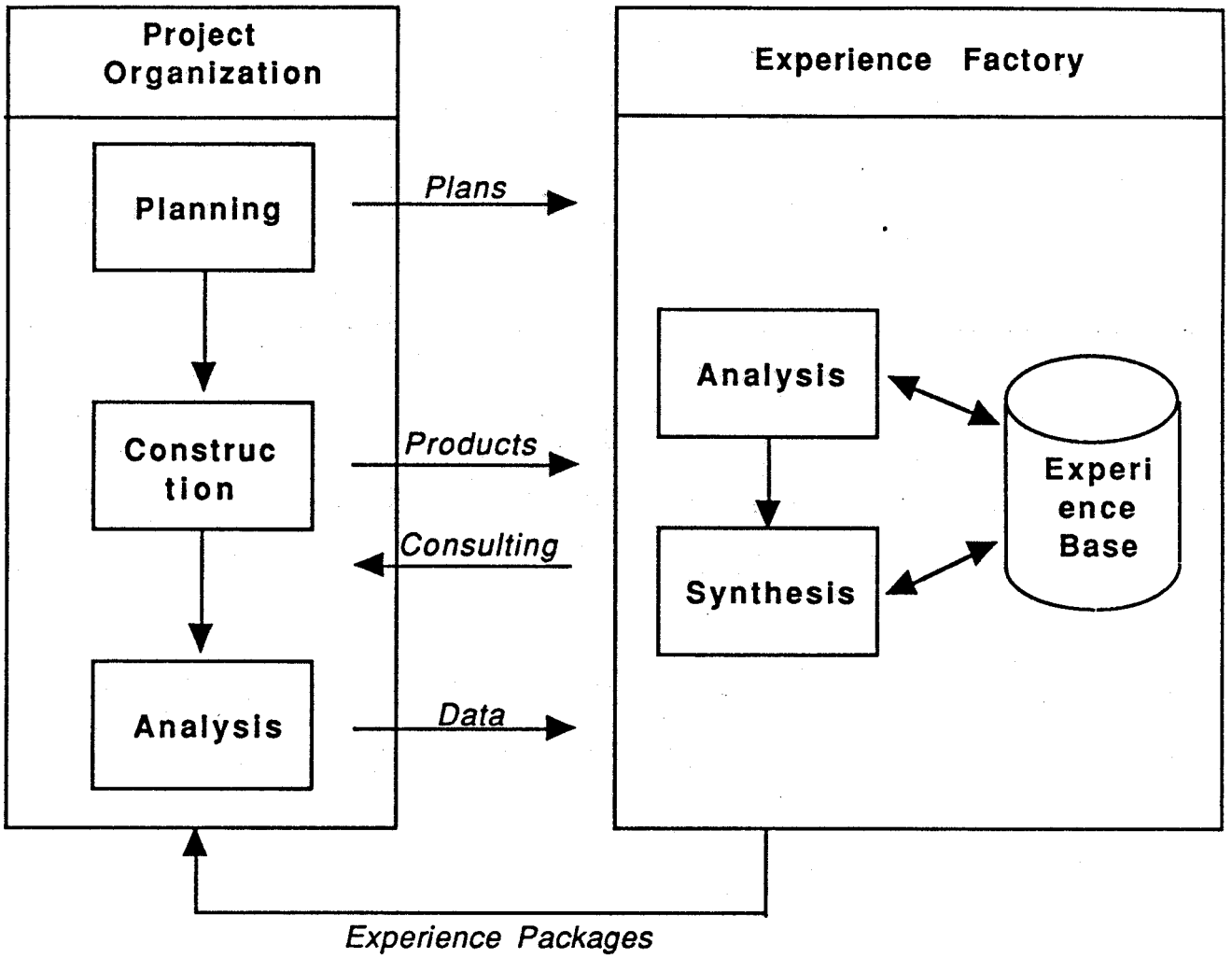
The experience factory can be a logical and/or physical organization, but it is important that its activities are separated and made independent from the ones of the project organization [Basili 1989].

The packaging of experience is based on tenets and techniques that are different from the problem solving activity used in project development:

Problem Solving	Experience Packaging
Decomposition of a complex problem into simpler ones	Unification of different solutions and redefinition of the problem
Instantiation	Generalization
Design/Implementation process	Analysis/Synthesis process
Validation and verification	Experimentation

In a correct implementation of the experience factory paradigm projects and factory will have different process models: each project will choose its process model based upon the characteristics of the software product that will be delivered, while the experience factory will define (and change) its process model

FIGURE 1
Experience Factory and Project Organization



based upon organizational and performance issues.

Outline of a Methodology for the Experience Factory

The main product of the experience factory is the experience package. There are a variety of software engineering experiences that can be packaged: resource baselines and models, change and defect baselines and models, product baselines and models, process definitions and models, method and technique models and evaluations, products, lessons learned, quality models. The content and the structure of an experience package vary based upon the kind of experience clustered in the package. There is, generally, a central element that determines what the package is: a software life cycle product or process, a mathematical relationship, an empirical or theoretical model, a data base, etc. We can use this central element as identifier of the experience package and produce a taxonomy of experience packages based upon the characteristics of this central element.

We can delineate a taxonomy consisting of the following classes of packages:

1.- Product Packages have as their central element a life-cycle product, clustered with the information needed to reuse it and the lessons learned in reusing it.

2.- Process Packages have as their central element a life-cycle process, clustered with the information needed to execute it and the lessons learned in executing it.

3.- Relationship Packages have as their central element a relationship or a system of relationships among observable characteristics of a software project. There are time based relationships and time independent relationships. In any case, these packages are used for analysis and/or forecast of relevant phenomena.

4.- Data Base Packages have as their central element a collection of data relevant for a software project or for activities within it. Project databases and quality records are common examples of this class of experience

packages.

5.- Management Packages have as their central element any container of reference information for project management: handbook, lessons learned document, reports, etc.

Experience packages are linked to each other by semantic links in which one package uses or incorporates another one. This network of semantic links provides the schema of the experience base, the repository of all products of the experience factory.

The core of the methodology used by the experience factory in order to transform existing software project experience into experience packages is the Improvement Paradigm. This paradigm is derived from the scientific method in order to supply environments such as software projects, in which mathematical formalization and organizational institutionalization play a prominent role, with an incremental learning methodology. It plays in software environments a role similar to the Shewart-Deming Cycle, known as Plan/Do/Check/Act (PDCA) Cycle, in manufacturing environments [Deming 1986].

As shown in Figure 2, the experience factory process can be outlined in three phases:

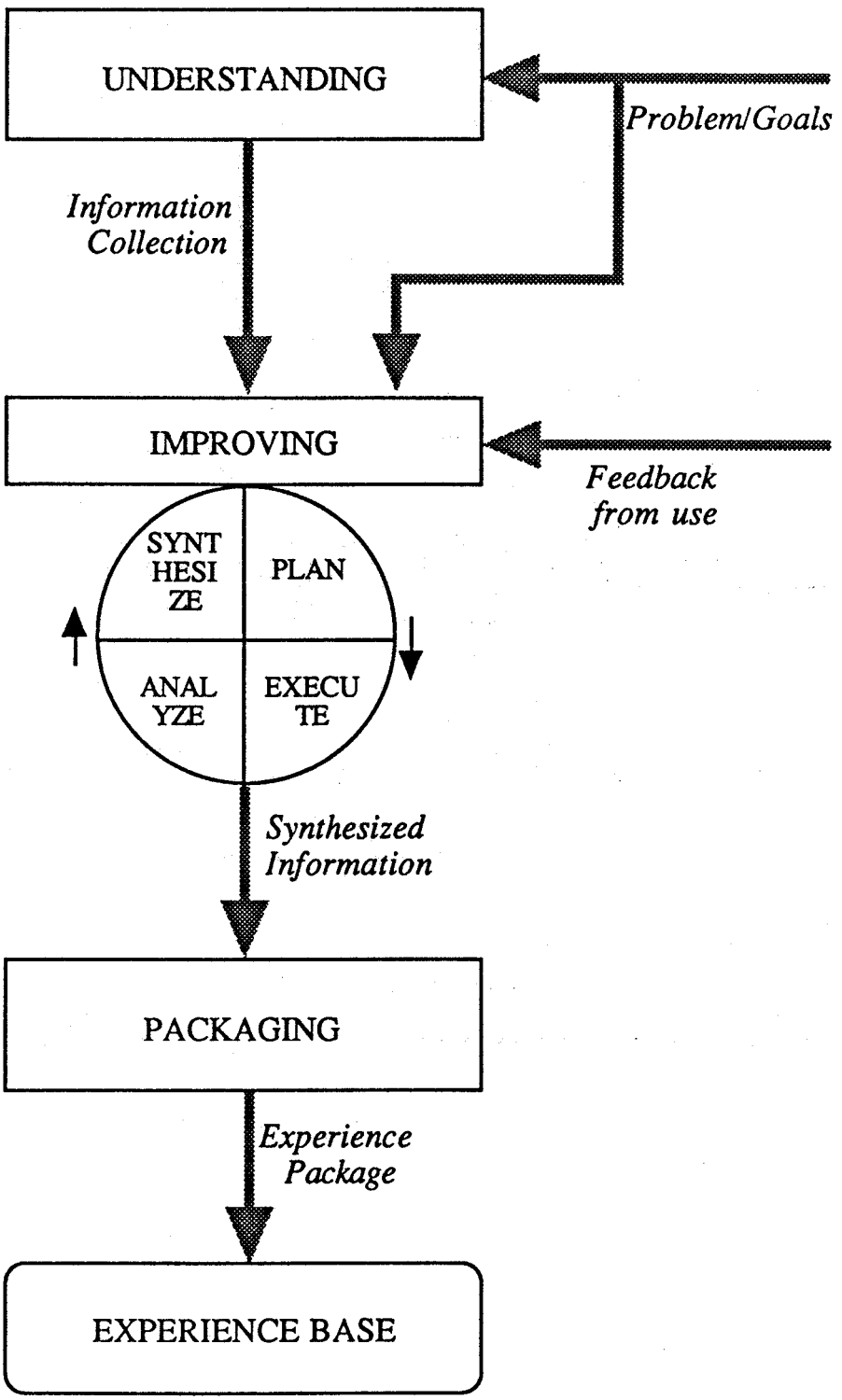
I. Understanding

- Define problem and set overall goals
- Collect information and data (internal and external, existing experience packages)
- Build a preliminary (i.e. not yet organized as an experience package) collection

II. Improving

- Single-project experimentation (Improvement Paradigm)

FIGURE 2
Experience Package Development



·· Plan

- Characterize the project environment

- Set experimentation goals and refine them into measurable form

- Choose execution process model, methods and tools

·· Execute the process and control it

·· Analyze the results and compare them with the goals

·· Synthesize the results into new or updated models

· Multiple-project experimentation
(Improvement Paradigm)

·· Plan

- Characterize project environments

- Set experimentation goals for each project and refine them into measurable form

- Adapt the execution process model, methods and tools to the projects

·· Execute the processes and control them

·· Analyze the results and compare them with the goals

·· Synthesize the results and update the models

III. Packaging

· Compare and consolidate synthesized information

· Choose outcome

· Compose/Update package from synthesized experience

· Store in the experience base

When a problem becomes relevant or a request for support has been formulated, the experience factory either selects and delivers an already existing experience package, or starts a development process. The outcome of this process may be the development of a new experience package or the integration of new information in an already existing package. Once an experience package has been developed, it is supplied to the project organizations that need it and used with the direct support of the experience factory. Feedback from this use is incorporated into the production process in the Improving Phase.

The Software Engineering Laboratory (SEL) at NASA Goddard Space Flight Center provides us with a concrete example of experience factory and of the application of the methodology we have just outlined [McGarry and Pajerski 1990]. The main interest of SEL has been software measurement and management by measurement. In the Understanding Phase at SEL the information and data gathering consisted in an extensive measurement program aimed at building descriptive baselines and models describing the specific environment. In the Improving Phase, while continuing the development of baselines and models, the SEL evaluated and fed back information to the projects. Most of the models used in this phase were informal but suited for experimentation. In the Packaging Phase, potentially reusable experiences were organized in experience packages such as

- Process packages: SEL Ada Process, SEL Cleanroom Process
- Database packages: SME (Software Management Environment)
- Model packages: SEL Schedule Model, SEL Basic Measure Model, SEL Environment Model, etc.
- Management packages: Manager's Handbook for Software Development

The diagram that follows shows the phases of the development of several experience packages as they were in the Fall of 1990 [Basili 1990]

SEL Creation	**SEL Experimentation**	**SEL Maturity**
		Packaging
		SEL Ada Process
		SEL Cleanroom Process
		SME
		Managers Handbook
	Improving	
	Methodology evaluation	Ada
	Cost Model Analysis	OOD
	Test Technique Analysis	Cleanroom
	QIP	CASE
Understanding		
Modeling environment	Design Measures	Test Method
Data Collection	Cost vs Size and Complexity	Reuse
Resource Baselines		
Defect Baselines		

Architecture of an Experience Factory

The structure and the functions of an efficient implementation of the experience factory concept are modeled on the characteristics and the goals of the organization supported by it. Therefore we need different levels of abstraction in the description of the architecture of an experience factory in order to introduce at the right level the specificity of each environment without loosing the

representation of the global picture and the ability to compare different solutions with each other [Basili, Caldiera and Cantone 1991].

We will discuss now briefly the levels of abstraction that we propose to use in order to represent the architecture of an experience factory (Figure 3):

- *Reference level*

  At this first and more abstract level, we represent the activities in the experience factory by active objects, called *architectural agents*: they are specified by their ability to perform specific tasks and to interact with each other.

- *Conceptual level*

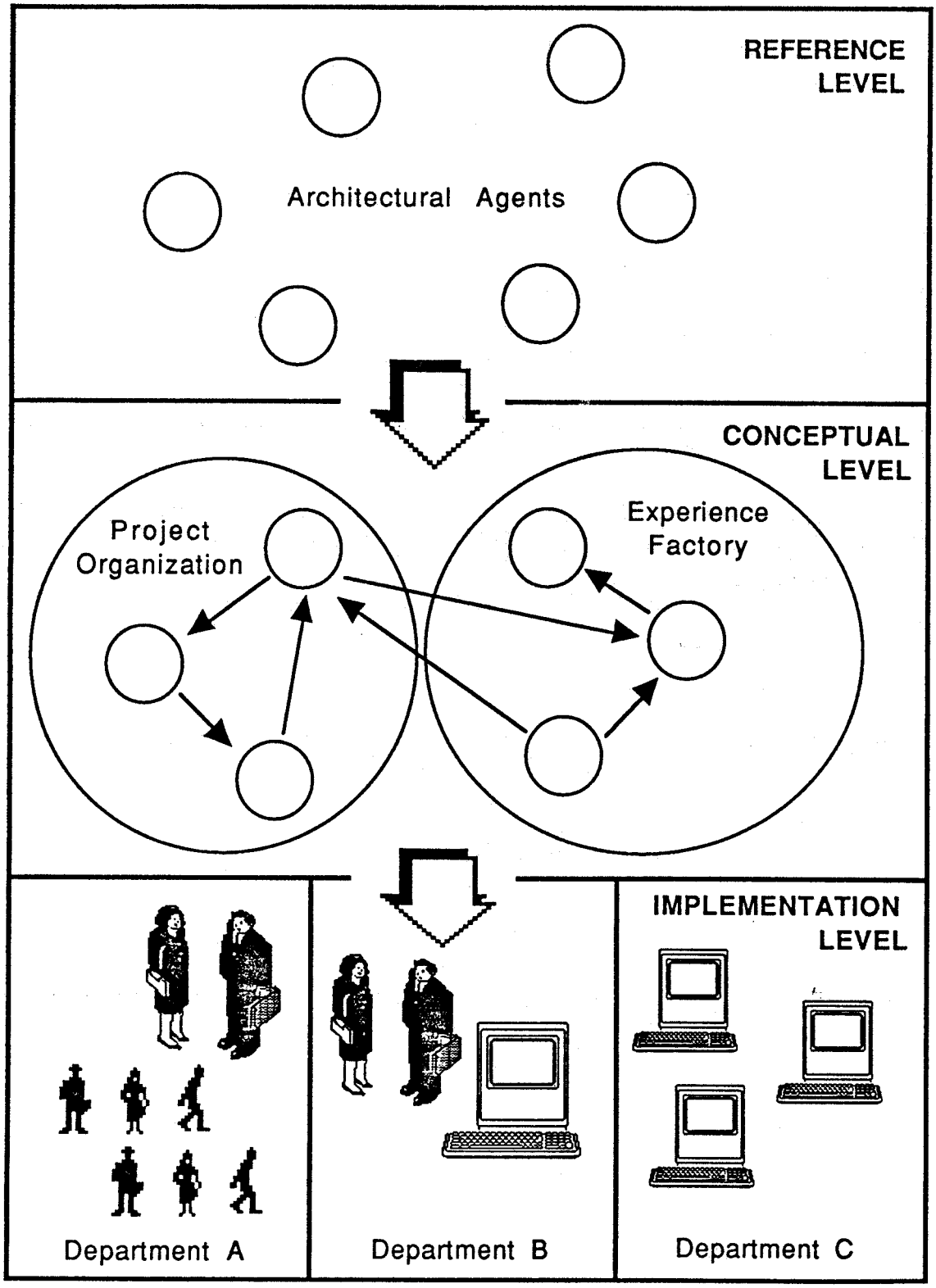
  At this level we represent the interface of the architectural agents and the flows of data and control among them, and specify who communicates with whom, what is done in the experience factory and what in the project organization. The boundary of the experience factory, i.e. the line that separates it from the project organization, is defined at this level based on needs and characteristics of an organization, and can change with them.

- *Implementation level*

  At this level we define the actual implementation, both technical and organizational, of the architectural agents and of their connections specified at conceptual level. We assign to them process and product models, synchronization and communication rules, appropriate performers (people or computers) and specify other implementation details. The mapping of the agents over the departments of the organization is included in the specifications provided at this level.

The architecture of the experience factory can be regarded as a very special instance of an experience package, whose design and evolution are based on the levels of abstraction just introduced and on the methodological framework of the improvement paradigm applied to the specific architecture.

FIGURE 3
Levels of Abstraction



The first step in the design of the experience factory is the identification of the activities aimed at capturing and packaging software project experience. In the last section, when we presented the experience package development process, we gave a summary of these activities

1. Problem definition and goal setting
2. Information collection
3. Preliminary clustering
4. Experimentation planning
5. Experimentation execution
6. Experimentation analysis
7. Experimentation synthesis
8. Information consolidation
9. Outcome selection
10. Experience package composition
11. Experience package update
12. Experience package storing
13. Experience package selection
14. Experience package use
15. User feedback generation

The second step in the design of the experience factory is the specification of the reference architecture (Reference Level Representation). This is done by grouping the activities identified in the previous step into architectural agents and specifying the possible connections between architectural agents. We can, for instance, identify the following architectural agents with the indicated distribution of activities:

A. Experience factory coordinator

Activities:

Problem definition and goal setting
Experimentation planning
Outcome selection

Possible connections with

Researcher

Experimenter
Experience package builder
Experience base manager
Experience package user

B. Researcher

Activities:
Information collection
Preliminary clustering

Possible connections with
Experience factory coordinator
Experience base manager
Experience package user

C. Experimenter

Activities:
Experimentation execution
Experimentation analysis

Possible connections with
Experience factory coordinator
Experience package builder
Experience package user

D. Experience package builder

Activities:
Experimentation synthesis
Information consolidation
Experience package composition
Experience package update

Possible connections with
Experience factory coordinator

Experimenter
Experience base manager
Experience package user

E. Experience base manager

Activities:
Experience package storing
Experience package selection

Possible connections with
Experience factory coordinator
Researcher
Experimenter
Experience package builder

F. Experience package user

Activities:
Experience package use
User feedback generation

Possible connections with
Experience factory coordinator
Researcher
Experimenter

The third step in the design of the experience factory is the conceptual representation of the experience factory architecture. In this step the following choices are performed

- Boundary between experience factory and project organization. In making this choice one tries to optimize reuse, on one hand, by incorporating more functions into the experience factory, and to optimize project support, on the other hand, by concentration of the appropriate activities in the project organization.

- Number and connection of agents. This choice is about communication complexity: a large number of agents increases the complexity and the overhead due to communication, a small number of agents produces bottlenecks that would affect the whole organization. Concentrating the control in a small group of agents makes planning easier, but serializes many activities that could be otherwise performed concurrently.

The diagram in Figure 4 is a possible choice of conceptual architecture where all the development of experience packages takes place in the experience factory. Other, less extreme, choices are possible, if some activities of the development process, for instance the experimentation, take place in the project organization. The first choice is probably better suited for environments where the practice of reuse of experience is somewhat formalized and mature. An organization that is just starting should probably instantiate its experience factory using a more traditional architecture and then, when it reaches a sufficient level of maturity and improvement with this architecture, start implementing the other architecture to continue the improvement. The improvement paradigm provides a methodology for a step-by-step approach to this implementation. In this way the organization takes advantage of the flexibility and evolutionary nature of this approach, that are among the primary benefits of reasoning in terms of instantiations of a reference architecture.

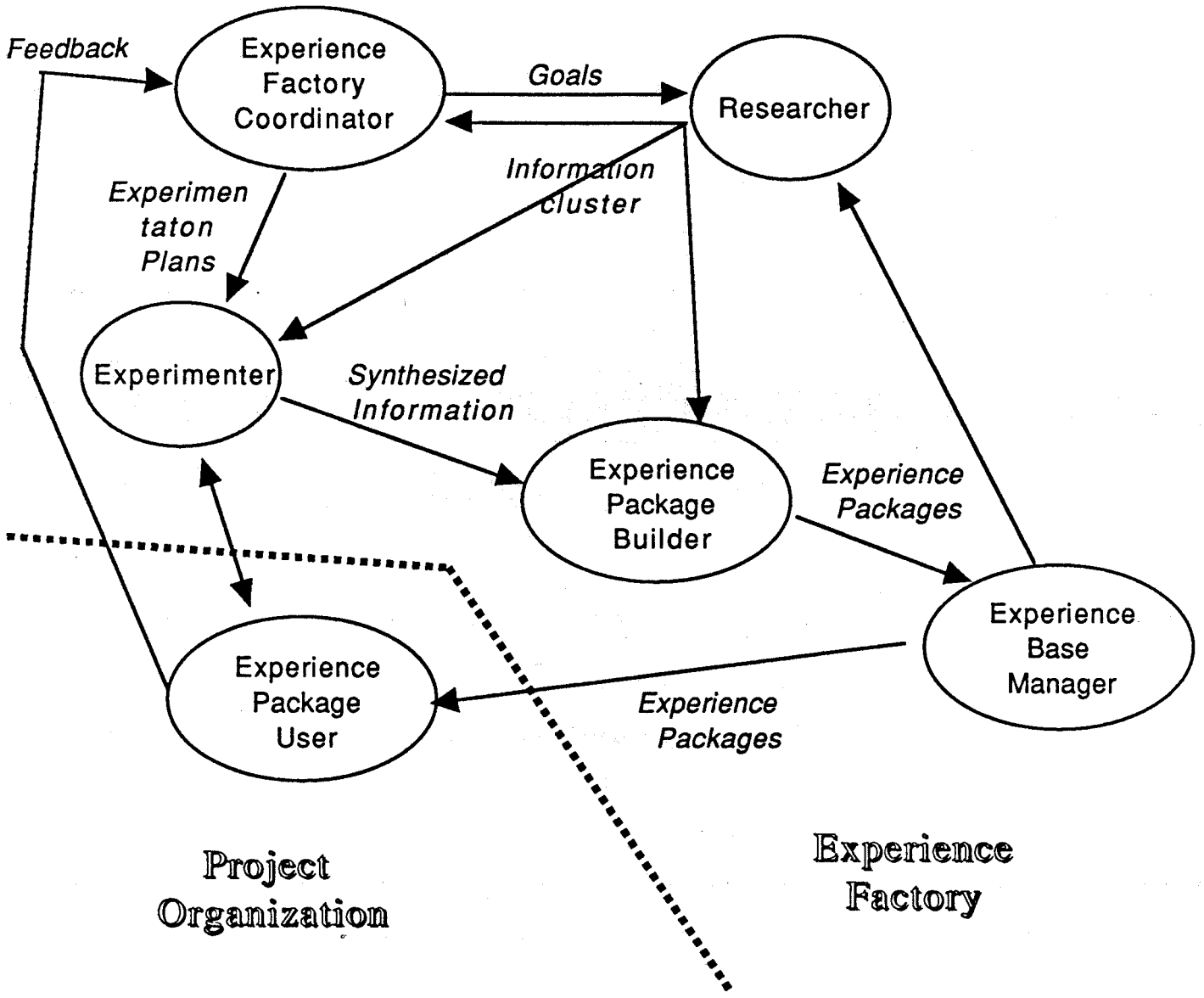
The fourth and final step in the design of the experience factory is the implementation of the specific factory by performing the following choices:

- Distribution of the agents over organizational units. This choice deals with the optimization of the already existing organization units and the smooth evolution to factory concepts. It takes into account the available resources and the historical roles of those units.

- Implementation of the functions of the agents. In choosing procedures, methods and tools one tries to achieve an organizational and technical profile that is correct, efficient and best suited to the overall mission by dealing with the available resources and technology.

The crucial point of the process is the possibility, offered by the reference architecture, of modifying the particular architecture without modifying the

FIGURE 4
Conceptual Architecture of an Experience Factory



interfaces between its building blocks. The modular structure allows configuration and reconfiguration of the processes as required by an efficient and realistic implementation of an optimizing process. The evolution of the conceptual level is more difficult because it has impact on the implementation level, but the explicit definition of the interfaces, which is part of the reference architecture, offers a certain freedom in the evolution, even at the conceptual level. Changes in the automation and organizational choices have definitely a lower impact, if they are applied to the implementation level leaving unchanged the conceptual level.

References

[Basili 1989]
V.R. Basili, "Software Development: A Paradigm for the Future", *Proceedings of the 13th Annual International Computer Software & Applications Conference (COMPSAC '89)*, Orlando, FL, September 20-22, 1989.

[Basili 1990]
V.R. Basili, "Towards a Mature Measurement Environment: Creating a Software Engineering Research Environment", *Proceedings of the 15th Annual Software Engineering Workshop*, NASA Goddard Space Flight Center, Greenbelt, MD, Software Engineering Laboratory Series, SEL-90-006, November 1990.

[McGarry and Pajerski 1990]
F. McGarry and R. Pajerski, "Towards Understanding Software - 15 Years in the SEL", *Proceedings of the 15th Annual Software Engineering Workshop*, NASA Goddard Space Flight Center, Greenbelt, MD, Software Engineering Laboratory Series, SEL-90-006, November 1990.

[Basili, Caldiera and Cantone 1991]
V.R. Basili, G. Caldiera and G. Cantone, "A Reference Architecture for the Component Factory", *Computer Science Technical Report Series*, CS-TR-91-24, University of Maryland, College Park, MD, March 1991.

[Deming 1986]
W. Edwards Deming, *Out of the Crisis*, MIT Center for Advanced Engineering

Study, MIT Press, Cambridge, MA, 1986

[Hamel and Prahalad 1991]
G. Hamel, C.K. Prahalad, "Corporate Imagination and Expeditionary Marketing", *Harvard Business Review*, Vol. 69, No. 4, July-August 1991, pp. 81-92.