Capacities-Centered Integral Software Process Formalization

S. T. Acuña[1], R. Giandini[2], C. M. Lasserre[3], V. Quincoces[3]
[1]*Universidad Nacional de Santiago del Estero (UNSE),*
Avenida Belgrano (S) 1912, (4200) Santiago del Estero, Argentina.
E-mail: ***silvac@unse.edu.ar***
[2]*Universidad Nacional de La Plata (UNLP), LIFIA, Calle 50 esq. 115, (1900) La Plata, Argentina*
E-mail: giandini@info.unlp.edu.ar
[3]*Universidad Nacional de Jujuy (UNJu), Gorriti 237, (4600) San Salvador de Jujuy, Argentina.*
E-mail: classerre@fi.unju.edu.ar; vquincoc@unju.edu.ar

Abstract

In this paper, we formalize an integral software process model which is applied to the construction of conventional systems (CS) and knowledge-based systems (KBS), centered in the capacities. The Capacities Centered Integral Software Process Model (CCISPM) is formalized through an object oriented approach. Aiming to automate the CCISPM, the formalization of the process dynamic aspects is begun. In this context, it is presented the dynamic modeling of the activities of the Project Initiation, Planning and Estimation Process. The formal model obtained, which represents the three Ps: processes, products and people, favours the direct understanding and communication of the process users (engineers, managers or developers) in relation to the model considered aspects.

1. Introduction

The prescription of the software construction process is a subject which has been studied in software engineering (SE) since the 80's. In 1991, the IEEE published a qualitative, informal and prescriptive model [16]. Since then many other proposals have emerged [20], [10], [19], [6], [11], [18], [17], [23], [9] trying to formalize and automate the construction process. Following the international trend which tries to integrate and bring closer the SE and Knowledge Engineering (KE), in 1996 the research group of UNSE-UNJU, Argentina, has developed an Integral Software Process Model (ISPM) which is applied in the SE and KE [2], validated and refined in 1999 [4] [5]. The originality of this model is especially for the KE, where there isn't a defined software process. Although this proposal might be known to SE, there are radical differences with the existing models: more importance is given to the first stages of development adding new activities to these stages.

These models have centered their representations in three characteristics of the process: the activity, the product and the agent (human and computarized) [22], but it is empirically shown [7], [24], [15], [26], [14], [8], [25], the great influence of other characteristics in the production process: human roles and organization. In general, the first characteristic is partly dealt in the existing software process models [11]. While the second is considered as an independent characteristic of those applied for the software process modelling [3], [23] or is ignored [11], since the organization belongs to the software process environment and that is why it is not explicitly modelled.

Yu and Mylopoulos [27] developed a descriptive model, The Actor Dependence Model, which is centered in the intentional relationships of the process actors in order to determine how they affect the decisions of an actor in the work of the others. [1] designed a capacities-centered integral software process model. This model considers and formalizes the capacities or behavioral competences of the software process actors. That is to say "who" are carrying out the activities of each of the ISPM processes.

This work has a double goal. On one hand, it describes the *Capacities Centered Integral Software Process Model* and its formalization. On the other hand, it presents *the dynamic aspects of the modeling of the activities of the Project Initiation, Planning and Estimation Process of that model.*

The integral software process based in the capacities is described in section 2. In section 3 the CCISPM is

0-7695-0810-3/00 $10.00 © 2000 IEEE

formalized by means of object orientation following the UML methodology. In section 4 using activity diagrams, the activities of Project Initiation, Planning and Estimation Process of the CCISPM are dynamically modeled. Lastly, in point 5 we state the conclusions.

2. Capacities Centered Integral Software Process Model

We name Capacities Centered Integral Software Process Model to the software process model that prescribes: a) the building of CS and KBS and of software integrating CS and KBS, and b) the capacities of those executing the defined role for each process. That is to say, the software process that represents "what is done" and "who is doing it", useful to the SE and KE.

The CCISPM represented in figure 1 [5] is made up by a subprocess that allows the selection of a software life cycle model that is set up as its axis and of other three subprocesses: one that manages it, other that models it and a third one assisting modelling. These subprocesses have been respectively named: Software Life Cycle Model Process, Project Management Processes, Software Modelling Processes and Integral Support Processes.

The proposed model involves four characteristics:

- *Activity:* It defines the activities made by the actors in each subprocess to develop one product.
- *Role:* It describes a group of capacities and responsibilities necessary to achieve the activities of each subprocess.
- *Product:* It defines the products generated by the subprocess activities.
- *Capacity:* It defines the skill or personal attribute of a subject's behaviour that can be defined as characteristic of his behaviour, and according to it, the activity oriented behaviour can be classified on a reliable and logical way.

The CCISPM is decomposed, so as to have a clearer description, on what is done? and who's who, the person doing it? The CCISPM subprocesses are described in the following sections considering these aspects.

2.1. What is done?

This aspect represents which products are being created, designed, managed, produced, delivered or assessed. In the rest of the work, due to extension, we shall center only in Project Initiation, Planning and Estimation Process.

In table 1 the functions of this process of the CCISPM are described and we present their activities to be made in order to become the input documents in output documents.

2.2. Who's who, the person doing it?

This aspect represents "who is the one" doing the "what". It is characterized by abstract people achieving the different model activities with their associated capacities. These capacities are exclusively referred to the *characteristics or skills of a person's general behaviour in his/her role in the process*, independently of other aspects such as the control of technological elements or specific knowledge.

In table 2 are shown the capacities that people performing the specified process activities must have. The detailed capacities are compulsory for each process. This does not imply that the person might possess other desirable capacities. Those marked H show one level of high requirement of capacity and with an L a low requirement in relation with the M that shows a medium requirement.

3. CCISPM Static Formalization

In this section the CCISPM static structure is formalized using an object oriented approach following the UML methodology [21]. The techniques and guides used by this *object oriented methodology* allow to support the elements *software process direct modelling and characteristics* described in the previous section in a definite way. This allows obtaining an integral software process representation in a direct way and close to reality, through totally detailed methodological steps.

Starting from the CCISPM textual specification, the relevant classes in the application domain are identified. The CCISPM class diagram is shown in figure 2. Due to simplicity only the name classes are allocated in figure 2. In figure 3 we specify the operations and capacities for Computing Engineer and Planning Engineer roles because of the importance it has in the dynamic formalization, which is presented in the following section.

The classes Project Team, Project Documents, System Supplier and User are a part of another class, Project. The Project Teams are made up of a variable number of Computing Engineers. Each Computing Engineer can fulfill one or more roles. In figure 2 this situation is shown using The Role Object Pattern presented by Fowler [12] where we find a main object having the roles it can play, as a set of role instancies, involving the State Pattern [13]. This representation allows a dynamic assignment because the instancies representing it can change in execution time.

Thus, the formal model in figure 2 represents in an

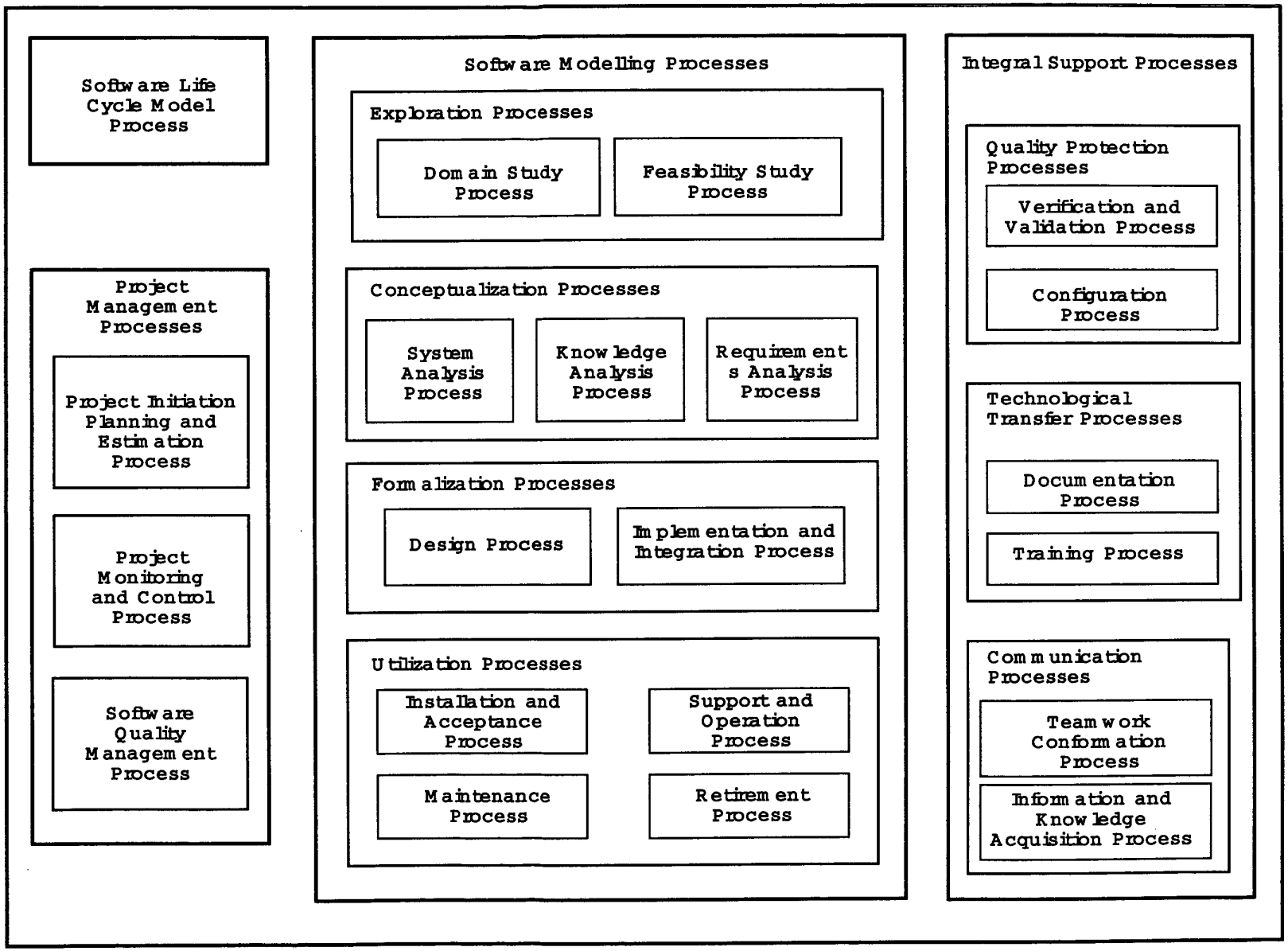


Figure 1. Integral Software Process Model

integral form both the what and the who's who in the proposed software process. This model assures the modelling of all the influencing elements of the software process in just one formalism, which allows to analyze in an efficient way the relationships both on the technical and human part of the process.

Description	Input Documents	Activities	Output Documents
During the *Project Initiation, Planning and Estimation Process* the software life cycle is defined (appointing a responsible person for each activity) for the project and management plans established. Necessary resources are estimated and allocated so as to execute the different tasks of the project. Standards, methodologies and tools are identified and selected for the management and execution of it and, lastly, a plan is prepared and established for an adequate and correct implemen-tation including milestones and revisions. During this process information collection and requi-red knowledge are planned during the whole software life cycle star-ting from any necessary source to construct CS and KBS.	- *Software Quality Improvement Recommendations* - *Software Quality assuran-ce Plan* - *Contractual Requirement* - *Selected Software Life Cycle Model* - *Need Report* - *Historical Project Records* - *Methodologies* - *Standards* - *Tools* - *Reusable Components* - *Written Work* - *Contingencies Plan* - *Transition Plan* - *Recommended Solutions* - *Teamwork List* - *Knowledge Sources*	*1) Establish the activi-ties map for the se-lected software life cycle model* *2) Project resources allocation.* *3) Project environment definition* *4) Knowledge acquisition planning* *5) Project management planning*	- *Software Project Management Plan* - *Retirement Plan* - *Information and Knowledge Acquisition Plan*

Table 1. Description of the elements of the Project Initiation, Planning and Estimation Process

4. Dynamic Formalization of the Initiation, Planning and Estimation Process

The proposed model captures the system static side. A static model implicitally includes the behaviour management that constitutes the dynamic aspect of it. Role allocation in team making is one of the behaviour aspects which is implicit in the static model. Next we are going to focus in the formalization of the dynamic aspects of the CCISPM. This modelling is represented using a graphical notation called the activity diagram, where there are internal operations (rounded rectangles), external operations (rectangles), events and triggers. Inputs and outputs for an activity or task are represented with a dotted rectangle.

Operations are processes that can be requested as a unit and can carry out the state change. Events define state changes that result from operations and invoke other operations via triggers. Control conditions ensure that a certain state exists before a certain operation has been triggered. A method is a process specification for an operation, which can be shown with an activity diagram thus making layer hierarchy.

In figure 4 we represent the modeling of the Project Initiation, Planning and Estimation Process. In another breaking down level, more specific, figure 5 shows Project Resources Allocation, Project Environment Definition, and Project Management Planning processes. Lastly, in figure 6 is shown the breaking down of Role Allocation Process in Employee Incorporation, Employee Dismissal and Role Replacement Processes.

5. Conclusions

The obtained formal model favours the process users (engineer, manager, developer) direct communication and comprehension of the process in relation to the considered model aspects. Human resources constitute the least formalized factor in current software process models. Yet, it is obvious its importance: they present a non determining and subjective behaviour that influences in the production results of software which is basically an intellectual activity. Also the lack of specification in human resources makes that the process does not reflect

PROCESSES	ROLES	CAPACITIES																				
		Metaskills					Betaskills				Operative Skills			Interpersonal Skills					Managing Skills		
		Problem analysis	Decision	Independence	Judgement	Stress tolerance	Environment knowledge	Innovation/creativity	Perseverance	Risk aversion	Self-organization	Discipline	Environmental approach	Customer Service	Negotiation capacity	Customer orientation	Sociability	Team Work/Cooperation	Collaborators evaluation	Group leadership	Planning and organization
SLCM PROCESS	Software Engineer	M	H	M		M	M	M	M			M	M			M		M			
PROJECT MANAGEMENT PROCESSES																					
Initiation, Planning and Estimation Process	Planning Engineer		H	M	M	M	M			M			M		H	M		H	M	H	H
Project Monitoring and Control Process	Monitoring and Control Engineer	H	H	M	M	M		M		M			M			M		H	M	H	
Software Quality Management Process	Quality Assurance Engineer	H	M	M		M		M					M			M		H	M	H	H
SOFTWARE MODELLING PROCESSES																					
EXPLORATION PROCESSES																					
Domain Study Process	Domain Analyst	M	L	M	M	M		M					M		M	M	M	H			
Feasibility Study Process	Feasibility Analyst	M	H	M	M	M		M					M		M	M	M	H			
CONCEPTUALIZATION PROCESSES																					
System Analysis Process	System Analyst	M	L	M		M							M	M	L	M	M	H			
Knowledge Analysis Process	Knowledge Analyst	M	L	M		M							M	M	L	M	M	H			
Requeriments Analysis Process	Requirements Analyst	M	L	M		M							M	M	L	M	M	H			
FORMALIZATION PROCESSES																					
Design Process	Designer		M	M		M			M		M	M	M			M		M			
Implementation and Integration Process	Implementation Manager		L	M		M			M		M	M	M			M		M			
UTILIZATION PROCESSES																					
Installation and Acceptance Process	Installation Manager		L	M		M			M		M	M	M	M		M		M			
Support and Operation Process	System Operator		L	M		M			M		M	M	M	M		M		M			
Maintenance Process	Maintenance Manager		L	M		M					M	M	M	M		M		M			
Retirement Process	Retirement Manager		L	M		M					M		M			M		M			
INTEGRAL SUPPORT PROCESSES																					
QUALITY PROTECTION PROCESSES																					
Verification and Validation Process	V & V Manager	M	L	M		M			M		M	M	M			M		M			M
Configuration Process	Configuration Manager		L	M		M					M	M	M			M		M			
TECHNOLOGICAL TRANSFER PROCESSES																					
Documentation Process	Documentalist		L	M		M					M	M	M			M		M			
Training Process	Trainer		L	M		M						M	M	M	M	M		M			
COMUNICATION PROCESSES																					
Teamwork Conformation Process	Team Manager	H	M	M		M		M				M	M	M	M	M		H	H	H	
Information and Knowledge Acquisition Process	Knowledge Engineer	M	L	M		M		M	M			M	M	M	M	M	M	H			

Table 2. Capacities and roles by CCISPM subprocess

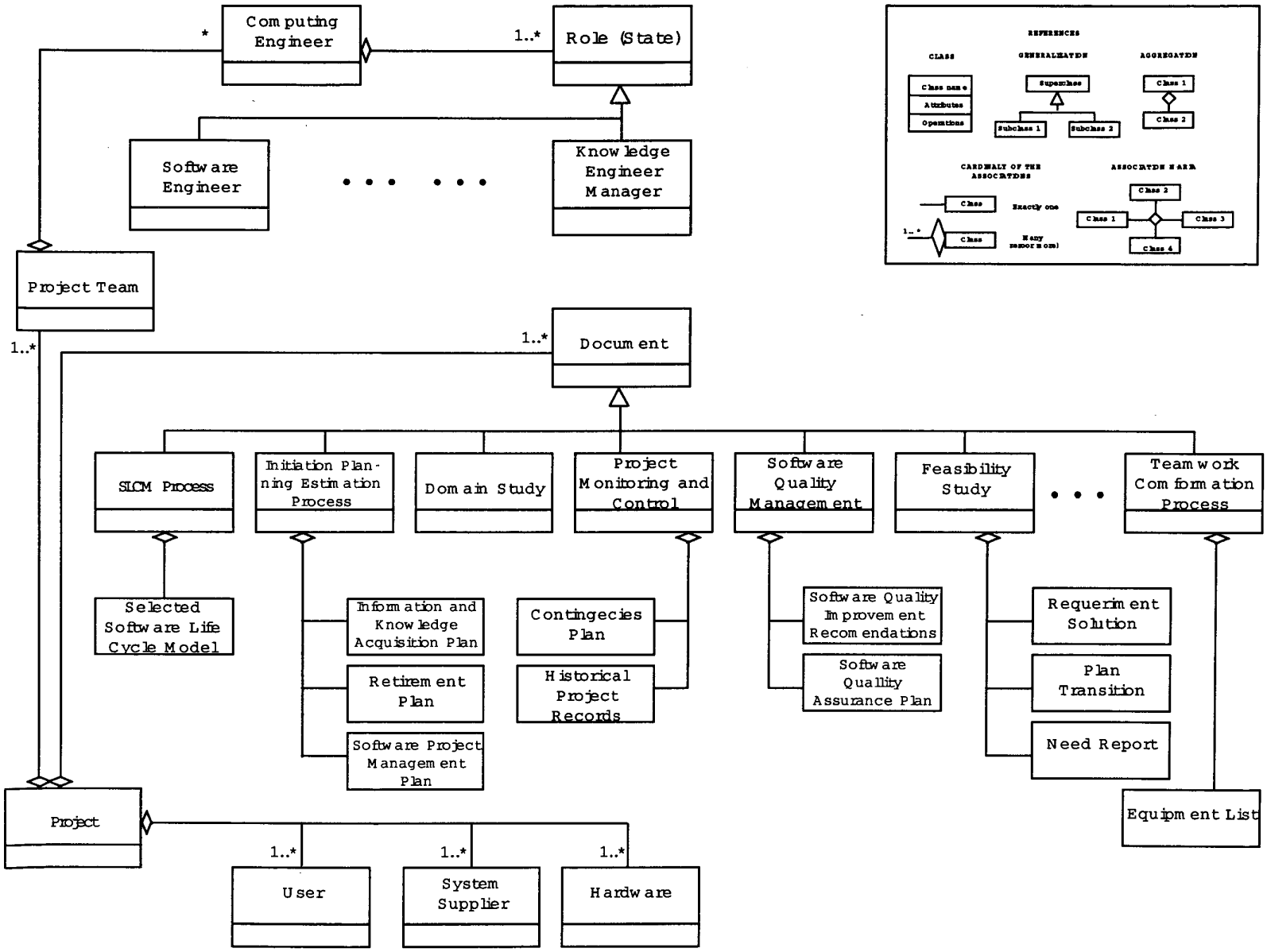


Figure 2. CCISPM Classes Diagram

the real situation of software process of the modelized organization, with the following risk that non-adequated processes to the capacity of these resources of the organization are executed. The Capacities-Centered Integral Software Process Model is a solution to this situation. This work presents a first approach to automatization of the CCISPM. A future work will go deeper analyzing how to incorporate intelligence to the system.

Computing Engineer	Planning Engineer
NAME EXPERIENCE AVAILABILITY LENGTH OF TASKS STARTING DATE DECISION INDEPENDENCE STRESS TOLERANCE ENVIRONMENTAL APPROACH CUSTOMER ORIENTATION TEAM WORK-COOPERATION	JUDGEMENT ENVIRONMENT KNOWLEDGE RISK AVERSION NEGOTIATION CAPACITY (H) COLLABORATOR EVALUATION GROUP LEADERSHIP (H) PLANNING AND ORGANIZATION (H)
Task execution	*Establish activities matrix (need report, contractual requirements, selected software life cycle model)* *Allocate project resources (activities of chosen life cycle, teamwork list)* *Define project environment (methodologies, historical project records, need report, tools, standards, written work)* *Plan project management (information and knowledge acquisition plan, software quality improvement recommendation, software quality assurance plan, need report, contractual requirement, reusable components, transition plan, recommended solutions, contingencies plan)* *Plan knowledge acquisition (written work, knowledge sources)*

Figure 3. Attributes and operations of Computing and Planning Engineers Roles Classes

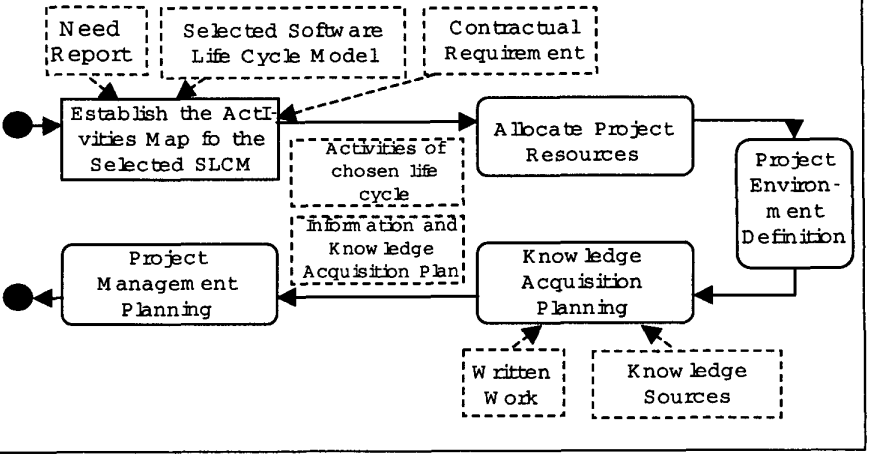
Project Initiation, Planning and Estimation Process	
Operation I : Establishes a correspondence between software process activities and those of the chosen life cycle. Operation II : Allocates persons to life cycle. Operation III: Defines pro-ject environment. Operation IV: Plans information and knowledge acquisition. Operation V: Plans project management.	

Figure 4. Activity Diagram for Project Initiation, Planning and Estimation Process

Project Resources Allocation Process

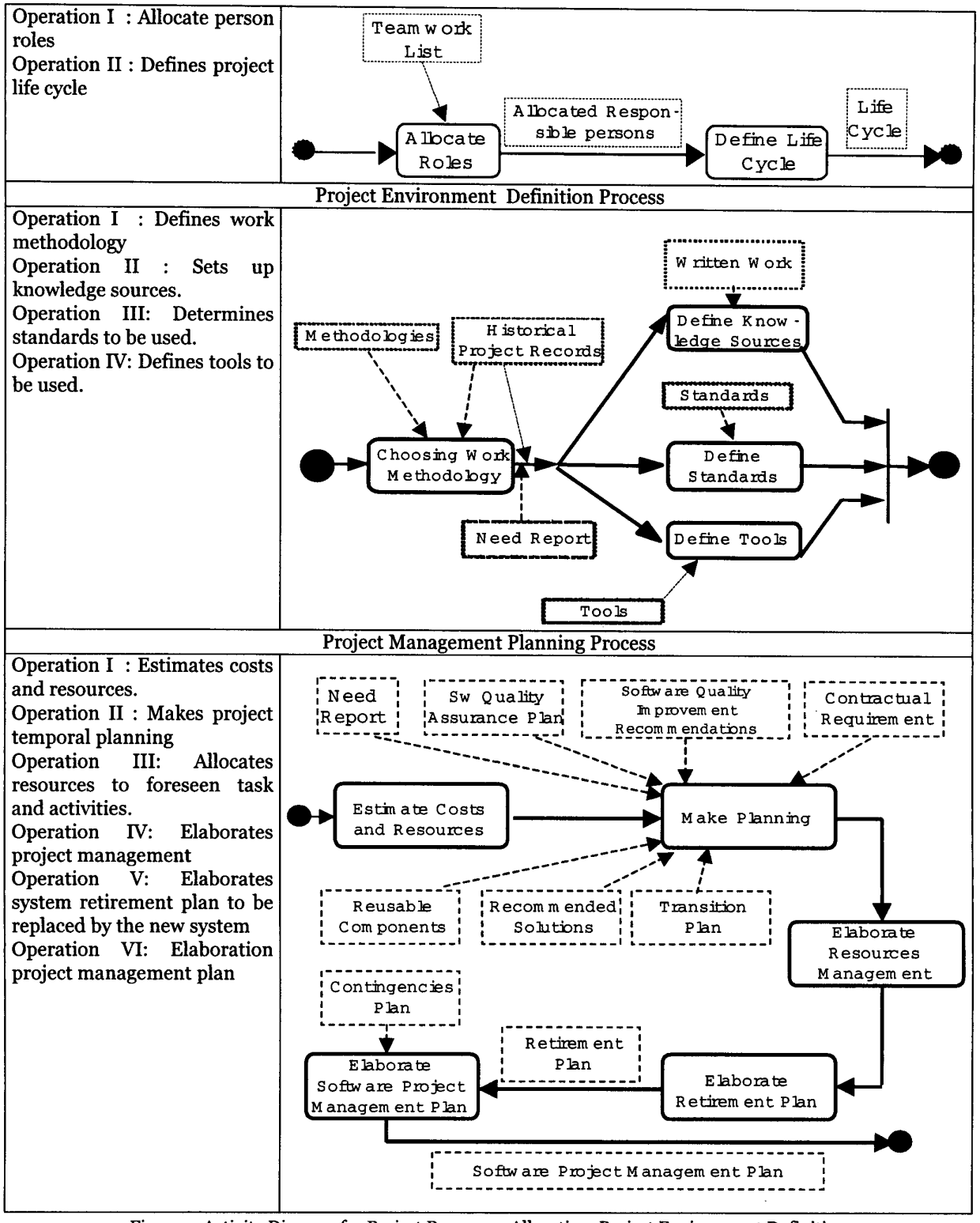


Figure 5. Activity Diagram for Project Resources Allocation, Project Environment Definition and Project Management Planning Processes

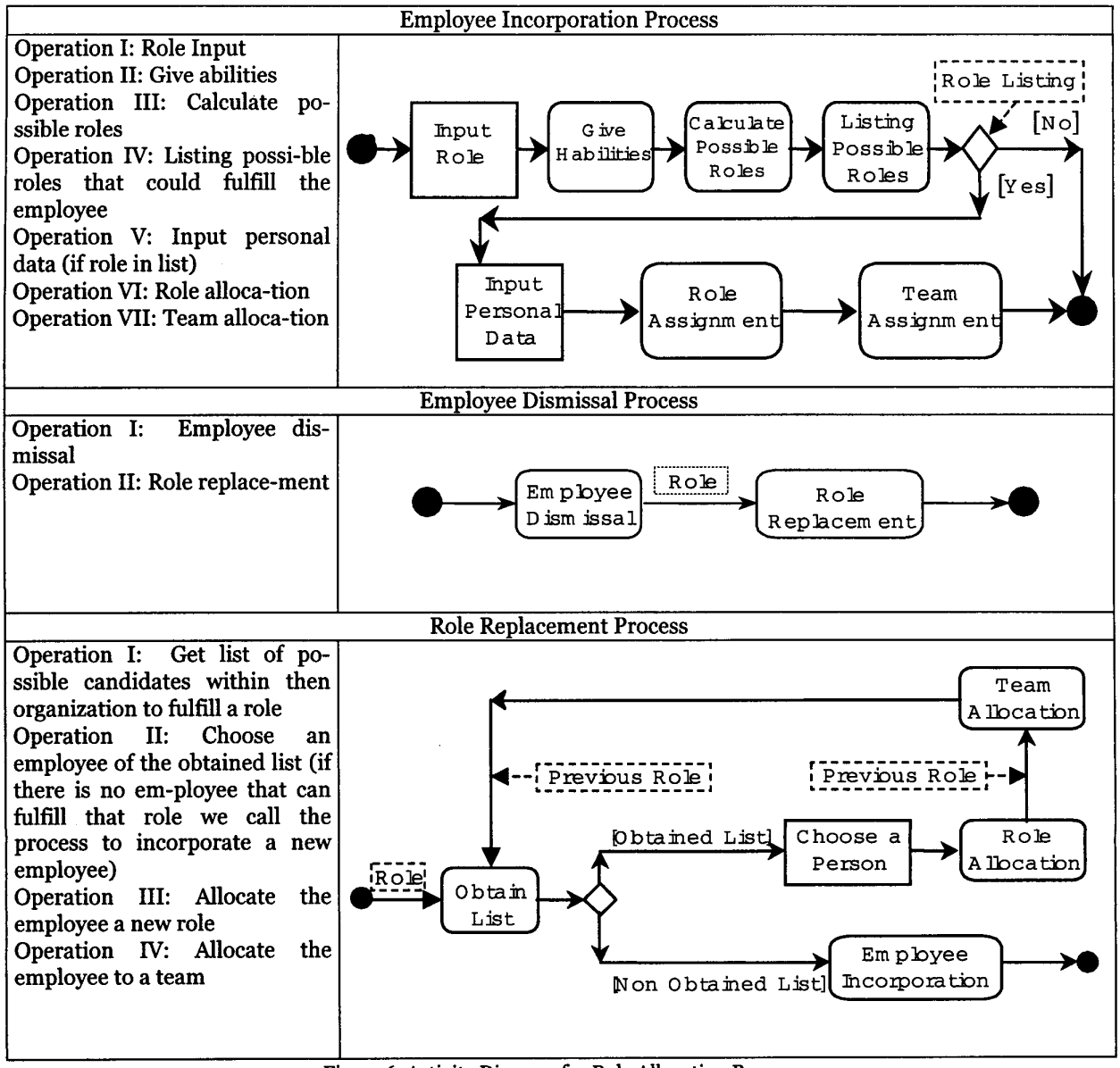


Figure 6. Activity Diagram for Role Allocation Process

References

1. S. T. Acuña, G. E. Barchini, C. Lasserre, V. Quincoces, M. Sosa and C. Tabera. "Software Engineering and Knowledge Engineering Software Process: Formalizing the Who's Who" . *Proceedings of the 13th International Conference on Software Engineering and Knowledge Engineering* (July 2000) In Press.

2. S. T. Acuña and N. Juristo, "Software process model for KBS development". The Journal for the Integrated Study of Artificial Intelligence, Cognitive Science and Applied Epistemology. *CC-AI* 13, 4 (1996) 1-39.

3. S. T. Acuña, G. E. Barchini and C. Lasserre, "Symbiosis of software and knowledge engineering: A multilevel software process cycle model". *Proceedings of the 9th International Conference on Software Engineering and Knowledge Engineering* (June 1997) 87-96.

4. S. T. Acuña and C. Lasserre, *Proceso Software: Un Modelo para la Ingeniería del Software y del Conocimiento.* (EDUNJU, 1999).
5. S. T. Acuña, J. Ares, N. Juristo, J. L. Morant and A. Pazos, "A process model applicable to software engineering and knowledge engineering". *International Journal of Software Engineering and Knowledge Engineering* (1999) 1-30. In press.
6. S. C. Bandinelli, A. Fuggetta and C. Ghezzi, "Software process model evolution in the SPADE environment". *IEEE Trans. Software Engineering* 19, 12 (December 1993) 1128-1144.
7. B. W. Boehm, *Software Engineering Economics.* (Prentice-Hall, Englewood Cliffs, 1981).
8. B. Curtis, H. Krasner and N. Iscoe, "A field study of the software design process for large systems". *Communications of the ACM* 31, 11 (November 1988) 1268-1287.
9. F. M. de Vasconcelos Jr. and C. M. L. Werner, "Software development process reuse based on patterns". *Proceedings of the 9th International Conference on Software Engineering and Knowledge Engineering* (June 1997) 97-104.
10. W. Deiters and V. Gruhn, "Software process analysis based on FUNSOFT nets". *Systems Analysis Modelling Simulation* 8, 4-5 (1991) 315-325.
11. J. Finkelstein, Kramer and B. Nuseibeh, *Software Process Modelling and Technology.* (Research Studies Press, 1994).
12. M. Fowler,."Dealing with roles". *Proccedings of the 4th Annual Conference on the Patterns Languages of Programs*, Monticello, Illinois, USA,. (September 1997) 2-5.
13. E. Gamma, R. Helm and J.Vlissides, *Design Patterns: Elements of reusable object-oriented software.* Reading, MA, (Addison-Wesley, 1995).
14. R. Guindon and B. Curtis, "Control of cognitive processes during design: What tools would support software designers?", *Proceedings of CHI'88*, ACM (1988) 263-268.
15. R. Hastie, *"Experimental evidence on group accuracy".* In Information Processing and Group Decision-Making, Ed. G. Owen and B. Grofman, (JAI Press, 1987) 129-157.
16. *IEEE Standard for Developing Software Life Cycle Processes*, IEEE Standard 1074-1991.
17. ISO/IEC *International Standard: Information Technology. Software Life Cycle Processes*, ISO/IEC Standard 12207-1995.
18. G. Junkermann, B. Peuschel, W. Schäfer and S. Wolf, *"MERLIN: Supporting cooperation in software development through a knowledge-based environment".* In Software Process Modelling and Technology cap. 5. (Research Studies Press, 1994) 103-129.
19. M. I. Kellner, "Software process modelling support for management planning and process modelling". *Proceedings of the First International Conference Software Process* (October 1991) 8-28.
20. J. Lonchamp, K. Benali, J. C. Derniame and C. Godart, "Towards assisted software engineering environments". *Information and Software Technology* 33, 8 (October 1991) 581-593.
21. J. Martin and J. Odell,. *Object Oriented Methods. A Foundation.* 2º Ed. (Prentice Hall, 1998).
22. I. R. McChesney, "Toward a classification scheme for software process modelling approaches". *Information and Software Technology* 37, 7 (1995) 363-374.
23. Sang-Yoon Min and Doo-Hwan Bae, "MAM nets: A Petri-net based approach to software process modeling, analysis and management". *Proceedings of the 9th International Conference on Software Engineering and Knowledge Engineering* (June 1997) 78-86.
24. W. Scacchi, "Managing software engineering projects: A social analysis". *IEEE Trans. Software Engineering* 10, 1 (January 1984) 49-59.
25. K. Sherdil and N. H. Madhavji, "Human-oriented improvement in the software process". In *Lecture Notes in Computer Science, Software Process Technology: Proceedings of the 5th European Workshop* 1149 (Springer-Verlag, 1996) 145-166.
26. H. J. Thamhain and D. L. Wilemon, "Building high performance engineering project teams". *IEEE Trans. Engineering Management* 34, 3 (March 1987) 130-137.
27. E. S. K. Yu and J. Mylopoulos, "Understanding "why" in software process modelling, analysis, and design". *Proceedings of the 16th International Conference on Software Engineering*, IEEE Computer Society (May 1994) 1-10.