Software Processes Modeling: 
What, Who, and When.

K. Benali and J.C. Derniame

Université de Metz/C.R.I.N. (Nancy) 
FRANCE

e-mail : benali@loria.fr

Introduction

Classical design methods (e.g. Data Base (DB) design methods) help the designer to tackle (and to answer) three essential questions: what, who, and when? At the conceptual level, the designer must decide:

- what is to be designed,
- who is concerned by a specific task,
- and when such a task occurs.

These three basic questions are the guidelines for designers (e.g. DB designers). As software process modeling is a key issue in users' assistance and guidance[2], we are going, in this short paper, to consider ourselves as Software Processes Model (SPM) designer, and from this point of view, we will highlight essential properties that a SPM formalism must fulfill to make SPM designer able to describe SPM.

1 What is to be modeled in SP?

The SPM designer needs to describe:

- activities: software development involves many activities of different nature as editing textual or graphical documents, handwriting documents, designing software gross architecture, designing in detail, coding detailed design, documenting source files, sending mail, compiling source files, linking object files, testing, making periodical saves of the source files... These activities range from menial tasks like sending a mail to complex tasks like design, and they can be classified in three types in regards to their immediate relation to the computer:
  - menial and possibly entirely automatable tasks (e.g. periodical saves of the source files),
  - menial but not performable without a human actor (e.g. sending mail),
  - complex and then not performable without a human actor (e.g. design).

SPM designers need specific formalism to be able to describe precisely this wide range of activities.

- artifacts: software development produces many artifacts of different nature as textual or graphical documents, handwritten documents, source files, object files,... These artifacts may be classified in two types in regards to their immediate relation to the computer:
- artifacts possibly within the storage system of the computer (e.g. files),
- artifacts not within the storage system of the computer (e.g. handwritten documents).

- actors: many actors are involved in software development. An actor, person or system, can play several roles, which are coherent sets of activities. A role can be played by several cooperating actors. The relevant roles will be shown in the next section entitled "Who are the actors of SP" These actors may be classified in two types in regards to their immediate relation to the computer:
  - human actors who are people developing software,
  - system actors which are logical or physical components of the computer.

- Interactions activities-artifacts: an activity is obviously "using" and producing artifacts. This relationship must be clearly described and even nuances in the way of using artifacts. For example, activities may consume artifacts or use an artifact and forbid or restrict its use to any other activity in the meantime: editing a source $s$ should forbid any updating action on $s$.

- Interactions activities-actors: an activity is performed by an actor playing a role while performing this activity. Describing this relationship should be more than just describing access rights.

- Interactions activities-activities: activities are not performed in a random order. A software product is built through precise sequences of activities. SPM driven IPSEs (Integrated Project Support Environments) are certainly not tool boxes offering tools for achieving activities at user’s convenience. These activities must be performed in respect to method, a policy. It is not useful to compile a source if this source has not been modified since its previous compilation. The more numerous are the ordering constraints expressing capabilities, the more powerfull is the SPM formalism. Ordering constraints may be the simple sequence, the concurrency, the repetition, the optionality,…

- Interactions artifacts-actors: the first interaction of this kind is through the interactions activities-artifacts and activities-actors. An actor performing an activity producing an artifact is indirectly producing this artifact. Nevertheless, other relationships may be of use: an actor may be responsible of artifacts (even if indirectly produced by someone else), or manage artifacts, or store artifacts,…

- Interactions artifacts-artifacts: an artifacts may be a “composite object” and then be a compound of many artifacts. This composition may be of different kinds. A component may be compulsory or not (e.g. a technical document may contain an index or not), a component may appear an undefined number of times (e.g. a technical document may have several authors),… More generally, artifacts are linked through semantic relationships.

- Interactions actors-actors: these interactions can be considered of two kinds:
  - informal interactions (e.g. exchanging ideas or informations in the coffee room) should not be explicitely modeled to avoid uncontroled growing of the SPM (even if such interactions are far from being the most unproductive),
  - "formal interactions" are already partialy described through the interactions activities-activities, activities-artifacts, artifacts-artifacts, activities-actors, and artifacts-actors. Nevertheless, direct formal interactions between actors may be of use: formal meetings between actors, administrative hierarchy between actors,…