cess Models of SDLCs cOMPArIsON AND EVOLUtION OF PrOcEss MODELs OF sDLcs Well-Defined Software Process Oktaba and Ibarguengoita (1998) have developed a general meta-model of software process (Figure 1), which provides a 'parsimonious' and 'engineer- ing-based' mode to conceptualize a well-defined software process.In this model, a process is com- posed of the following elements: phases, activities, artifacts, roles, and agents; where a software process is the main concept that is being modeled; a phase is the highest-activity level of a process that is be- ing modeled; and an activity is the execution of an useful work to deliver a main artifact or artifacts (e.g., pieces of the full software artifact, documents, components, data files, or codes). The concepts of role and agent complete this semi-formal definition. A role is a functional responsibility in the process that is assigned to an agent, which can be a human- being, a tool, or a combination of both. The class software process is made up of the several instances of the phase class. Figure 1 shows the meta-model using a class diagram notation. An instance of a process class is composed of several instances of phase class. Phase class is re- lated with several instances of activity class. Under an operation of specialization, the authors report that the phase class can be specialized in analysis, design, code and test, and installation phases. Similarly, the activity class can be specialized in production, control, technology, and communica- tion activities, and each one of these, in other spe- cializations. The activity class is also related to at least an input artifact and an output one, represented by the instances of the artifact class. Specialization of artifacts is also feasible in the model. Finally, a many-to-many association between the role and activity classes and a one-to-many from agent and role classes are defined in the model. We use this meta-model as base for a conceptual framework to compare different process models. Under the consideration of each life cycle is an instance of the model; the comparative framework provides a theoretical base to develop instances for the generic classes of phase, artifact, and role. Activity and agent classes are not considered in this chapter. Well-defined software process model Software Process Phase Artifact Activity Rol Agentare difficult to develop and test. Very often, software exhibits unexpected and undesired behaviors that may even cause severe problems and damages.For these reasons, researchers and practitioners have been paying increasing attention to understanding and improving the quality of the software being developed. The underlying assumption is that there is a direct correlation between the quality of the process and the quality of the developed software. The research area that deals with these issues is referred to using the term software process. The large diversity of PM-SDLCs suggests, then, that apparently none of the PM-SDLCs is sufficient for covering all needs to guarantee a successful development of software-intensive systems. This study, then develops a comparison of the main PM-SDLCs based on their historical evolution, and in terms of their component structure (e.g., based in the Oktaba & Ibarguengoitia meta-model, 1998) to help organize the available knowledge on these models.The 13 PM-SDLCs analyzed are: waterfall (Royce, 1970), SADT (Dickover, McGowa, & Ross, 1977), prototyping (Naumann & Jenkins, 1982), structured cycle (Yourdon, 1993), spiral (Boehm, 1988), win-win spiral (Boehm & Rose, 1994; Egyed & Boehm, 1998), unified process (Rational Software Corporation, 1998), MBASE (Center for Software Engineering, 1997), component-based cycles (Aoyama, 1998; Brown & Wallnau, 1996), XP (Beck, 1999), PSP (Humphrey, 2000), TSP (McAndrews, 2000), and RAD (Cross, 2006).The most relevant findings from Table 1 can be summarized as follows:

a. The set of common phases includes the analysis, design, codification, testing, and implementation phases (Note: the emergent agile-based systems approaches such as XP also support an engineering view of these phases); b. The initial business and high-level systems phases (as part of the macro-phase definition of the system) are only part of some PM- SDLCs; c. The iterative approach was disseminated by prototyping SDLC, but this was originally suggested in the Royce's4 (1970) variant of the waterfall model.Later, was reinforced and extended by the spiral model; and d. The postmortem phase, which appeared previ- ous to year 2000, was indirectly suggested by MBASE and XP, and it is attributed mainly to PSP and TSP models.For this, the following specialization of phases was identified in the same study: user conditions, business context pre-systematization, component identification, requirements, analysis, design, coding, test, implementation, postmortem analysis, and iteration decisions.Table 1 contributes to organize comprehensively the phases reported of practically all public PM-SDLCs, and suggests from its analysis a set of generic phases.Table 2 shows the comparative framework for the "artifact" component versus the several PM-SDLCs studied.We propose a comparative specialization of artifacts.Phases and Artifacts in the PM-sDLcs It has been also reported (Fuggetta, 2000) that: software applications are complex products that Figure 1.For the cases of "2," "3," "4," and "5," they indicate that more of one artifact of the PM-SDLC (indicated in the column) corresponds to a unique artifact of the comparative specialization.These activities constitute the generic life-cycle (proposed in this chapter) that includes all activities of the PM-SDLC under study.A scheme of three macro-phases (definition, development, and deployment) well-known in systems engineering is used to group the phases (Sage & Armstrong, 2000).used in Table 1 indicates that the phase reported in the related row is part of the PM-SDLC reported in the corresponding column.Phases are grouped by the general macro-phases: definition of the system, development of the system, and deployment of the system, a well-know systems engineering model.It must be noted that the unique features are considered in the period of their formulation, then, some elements that were considered unique at once, later were incorporated to other models.The phases are proposed by consider- ing all activities that are part of each phase of each PM-SDLC under study.Table 1 shows the comparative framework, for the phase class of the 13 PM-SDLCs analyzed.In this table, each number means the number of artifacts that are generated as equivalent to the artifact specialization proposed.No similar comparison was found in the literature.The symbol ?