**3.3 INCREMENTAL PROCESS MODELS**

**from Pressman 5h ed.**

There are many situations in which initial software requirements are reasonably well-defined, but the overall scope of the development effort precludes a purely linear process. In addition, there may be a compelling need to provide a limited set of software functionality to users quickly and then refine and expand on that functionality in later software releases. In such cases, a process model that is designed to produce the software in increments is chosen.

*“Too often, software work follows the first law of bicycling: No matter where you’re going, it’s uphill and against the wind.” Author unknown*

**3.3.1 The Incremental Model**

***Key Point:*** *The incremental model delivers a series of releases, called increments, that provide progressively more functionality for the customer as each increment is delivered.*

The incremental model combines elements of the waterfall model applied in an iterative fashion. Referring to Figure 3.2, the incremental model applies linear sequences in a staggered fashion as calendar time progresses. Each linear sequence produces deliverable "increments" of the software [MCD93]. For example, word-processing software developed using the incremental paradigm might deliver basic file management, editing, and document production functions in the first increment; more sophisticated editing, and document production capabilities in the second increment; spelling and grammar checking in the third increment; and advanced page layout capability in the fourth increment. It should be noted that the process flow for any increment may incorporate the prototyping paradigm discussed in Section 3.4.1.

When an incremental model is used, the first increment is often a core product That is, basic requirements are addressed, but many supplementary features

81

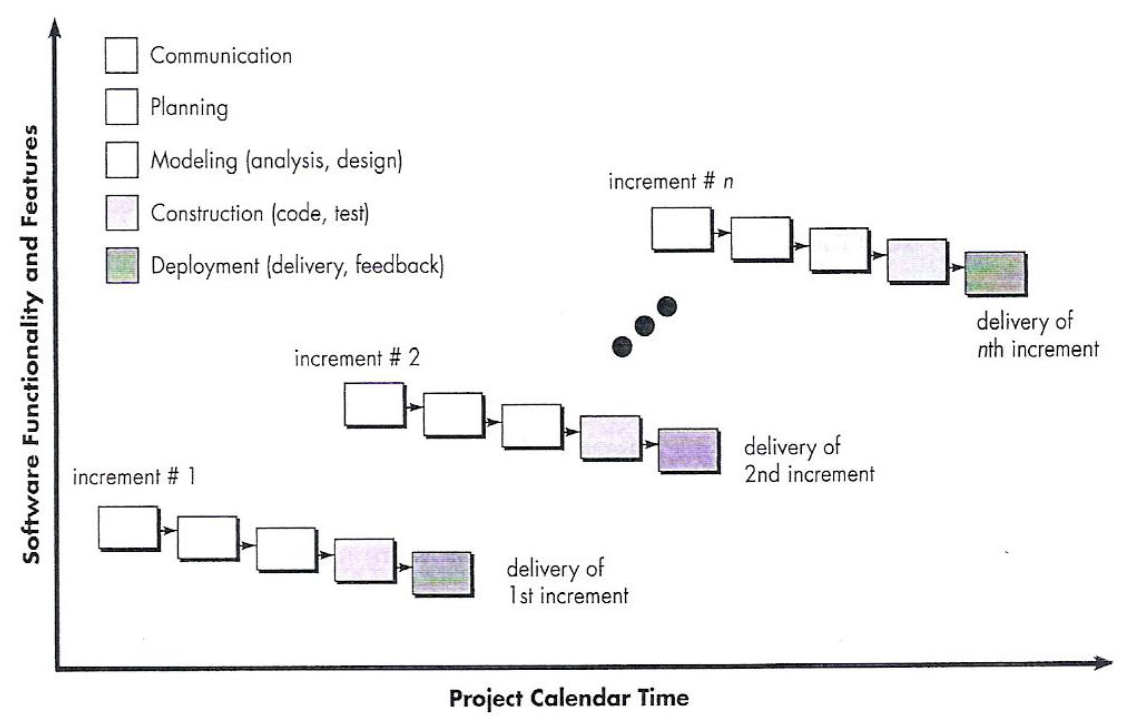


FIGURE 3.2 The incremental model

(some known, others unknown) remain undelivered. The core product is used by the customer (or undergoes detailed evaluation). As a result of use and/or evaluation, a plan is developed for the next increment. The plan addresses the modification of the core product to better meet the needs of the customer and the delivery of additional features and functionality. This process is repeated following the delivery of each increment, until the complete product is produced.

The incremental process model, like prototyping and other evolutionary approaches, is iterative in nature. But unlike prototyping, the incremental model focuses on the delivery of an operational product with each increment. Early increments are "stripped down" versions of the final product, but they do provide capability that serves the user and also provides a platform for evaluation by the user.

Incremental development is particularly useful when staffing is unavailable for a complete implementation by the business deadline that has been established for the project. Early increments can be implemented with fewer people. If the core product is well received, additional staff (if required) can be added to implement the next increment. In addition, increments can be planned to manage technical risks. For example, a major system might require the availability of new hardware that is under development and whose delivery date is uncertain. It might be possible to plan early increments in a way that avoids the use of this hardware, thereby enabling partial functionality to be delivered to end-users without inordinate delay.

*Avice: If your customer demands delivery by a dote that is impossible to meet suggest delivering one or more increments by fat dote ond the rest of the software (additional increments) later.*

**3.3.2 The RAD Model**

*Rapid Application Development* (RAD) is an incremental software process model that emphasizes a short development cycle. The RAD model is a "high-speed" adaptation

*3 It is important to note that an incremental philosophy is also used for all “agile” process models discussed in Chapter 4.*

82

of the waterfall model, in which rapid development is achieved by using a component-based construction approach. If requirements are well understood and project scope is constrained,**4** the RAD process enables a development team to create a "fully functional system" within a very short time period (e.g., 60 to 90 days) [MAR91].

Like other process models, the RAD approach maps into the generic framework activities presented earlier. Communication works to understand the business problem and the information characteristics that the software must accommodate. Planning is essential because multiple software teams work in parallel on different system functions. Modeling encompasses three major phases—business modeling, data modeling and process modeling—and establishes design representations that serve as the basis for RAD's construction activity. Construction emphasizes the use of preexisting software components and the application of automatic code generation. Finally, deployment establishes a basis for subsequent iterations, if required [KER94j.

The RAD process model is illustrated in Figure 3.3. Obviously, the time constraints imposed on a RAD project demand "scalable scope" [KER94j. If a business application can be modularized in a way that enables each major function to be completed

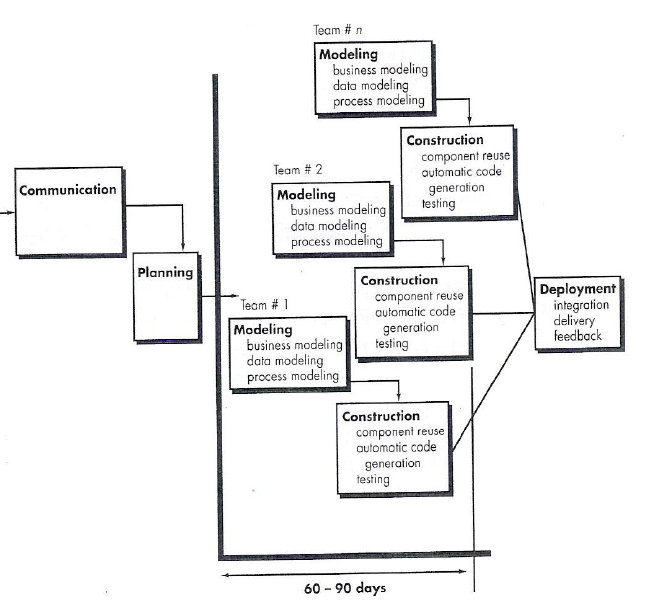


Figure 3.3 The RAD Model

*4 These conditions are by no means guaranteed. In fact, many software projects have poorly defined requirements at the start. In such cases prototyping or evolutionary approaches (Section 3.4) are much better process options. See [REI95].*

83

in less than three months (using the approach described above), it is a candidate for RAD. Each major function can be addressed by a separate RAD team and then integrated to form a whole.

Like all process models, the RAD approach has drawbacks [BUT94]:

(1) for large, but scalable projects, RAD requires sufficient human resources to create the right number of RAD teams; Büyük fakat ölçeklenebilir projelerde RAD doğru sayıda takım oluşturmak için tatmin edici insan kaynağı bulmayı gerektirir.

(2) if developers and customers are not committed to the rapid-fire activities necessary to complete the system in a much abbreviated time frame, RAD projects will fail; Eğer kullnıcılar ve geliştiriciler sistemi oldukça sıkıştırılmış sürede tamamlayacak hızlı faaliyete razı değillerse (adanmış değillerse) RAD projeleri çöker.

(3) if a system cannot be properly modularized, building the components necessary for RAD will be problematic; Eğer sistem doğru şekilde modülerize edilmemişse RAD için gerekli bileşenleri inşa etmek problemli olabilir.

(4) if high performance is an issue, and performance is to be achieved through tuning the interfaces to system components, the RAD approach may not work; Eğer performnas bir mesele ise sistem bileşenlerinin tüne dilmesi ile ulaşıabilir ise RAD yaklaşımı çalışmayabilir.

and

(5) RAD may not be appropriate when technical risks are high (e.g., when a new application makes heavy use of new technology). RAD teknik risklerin yüksek olması durumunda uygun olmayabilir.