مهندسی نرم افزار

تدریس استادار جمند دکتر حمید خورسند رحیم زاده

استاد دانشکده فنی شهید شمسی پور تهران

گردآوری شده توسط محمد حیدری

دانشجوی رشته مهندسی کامپیوتر دانشکده فنی شهید شمسی پور تهران

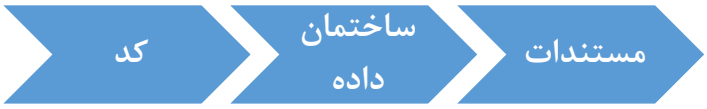
برای شادی روح برادر عزیزم احمد حیدری صلواتی هدیه بفرمایید . سپاس

تقدیم به روح آسمانی دانشجوی شهید حسینعلی شمسی پور

وهمه شهدای دانشجوی این مرز و بوم

منبع اصلی درس : **Software Engineering , A Practitioners Approach 8th Edition**

یک نرم افزار از بخش های ذیل تشکیل شده است :



ویژگی های نرم افزار :

- **Software will be Developed Not Manufactured**

نرم افزار توسعه داده می شود و ساخته نمی شود.

Develop: به معنای پیشرفت بهتر شدن و کامل شدن است. مراحل شروع فکر کردن ساخت نرم افزار تا تحویل آن به مشتری نام دارد.

- **Software doesn't wear out**

Wear out: به معنای خستگی یا کهنگی می باشد. مثال : یک کفش وصله دار.

Custom Base: نرم افزار بصورت سفارشی ساخته میشود.

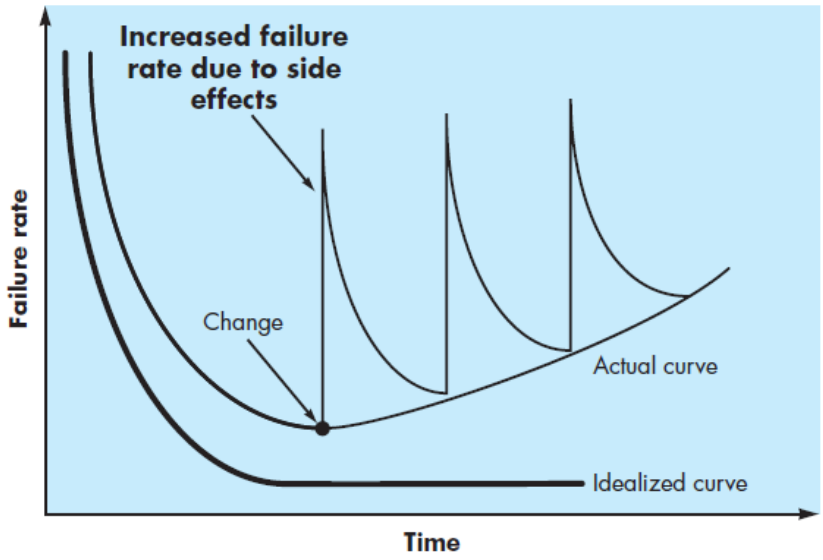
- **Software is Custom based**

نکته : مستندات داخل هر نرم افزار گنجانده شده است.

سوالاتی در رابطه با نرم افزار :

- ✓ **چرا زمان زیادی نیاز داریم تا نرم افزار تمام شود ؟**
- ✓ **چرا هزینه های توسعه نرم افزار زیاد است ؟**
- ✓ **چرا نمی توانیم تمامی ایرادهای سیستم را پیش از تحویل شناسایی کنیم ؟**
- ✓ **چرا نمی توانیم میزان پیشرفت را اندازه گیری کنیم ؟**

نمودار نرخ خرابی

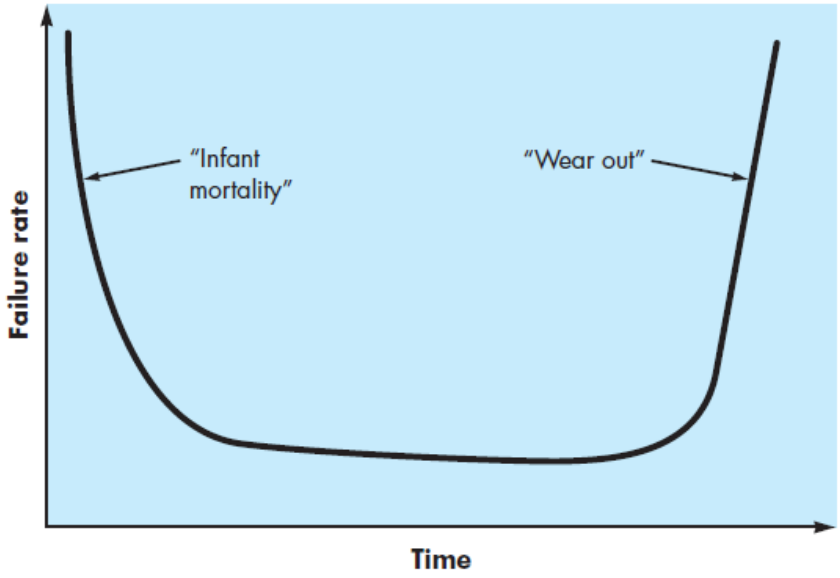


در ابتدای کار تمامی نرم افزارها خرابی دارند و این مرحله را تست نامگذاری میکنیم.

Legacy Software: **User Interface** را به شکل ثابت حفظ میکنیم ولی **Back End** را تغییر میدهیم.

KEY POINT

Software is both a product and a vehicle that delivers a product.



نمودار 1-1 نرخ شکست سخت افزار صفحه 5 کتاب

Quote:

"Software is a place where dreams are planted and nightmares harvested, an abstract, mystical swamp where terrible demons compete with magical panaceas, a world of werewolves and silver bullets."

Brad J. Cox

انواع نرم افزارها

- **System Software**
- **App Software**
- **Engineering Software**
- **Embedded Software**
- **Web App Software**
- **AI Software**

نرم افزارهای سیستمی : سرویس دهی به سایر نرم افزارها را بر عهده دارند.

مثال : سیستم عامل

نرم افزارهای کاربردی : جهت پشتیبانی فعالیت های یک حرفه مخصوص طراحی شده اند.

مثال : سیستم آموزش دانشگاه

نرم افزارهای مهندسی : در حل مشکلات کاربردی روزمره استفاده میشوند.

مثال : برنامه ای که محل نصب ایستگاه های تقویت سیگنال موبایل در تهران را مشخص کند.

روش : تشکیل دستگاه نا معادله – جواب : محل نصب ایستگاه ها – زبان : Fortran

نرم افزارهای تعبیه شده : کمک میکنند تا از کارآیی سخت افزار به شکل بهینه استفاده کنیم.

مثال : نرم افزار موجود در ماشین لباسشویی

نکته : Embedded به معنای تعبیه شده است.

نرم افزارهای هوش مصنوعی : هوش مصنوعی یعنی هر نوع قابلیت هوشمندی که به ابزار بدهیم.

یکی از شاخصه های مهم بحث هوشمندی بحث تصمیم گیری است.

مثال : نرم افزارهایی که ایمیل ها را با توجه به Key که به سیستم داده ایم چک می کنند.

مثال 2 : بازی شطرنج . برای برنامه نوسی این نوع مباحث از زبان هایی مثل Lisp بهره می بریم.

دو دسته بندی دیگر هم می توان قائل شد :

- **Open Source**
- **Open Word Computing**

ویژگی های خاص نرم افزارهای تحت وب

- **Network Intensiveness**
- **Concurrency**
- **Performance**
- **Continuous Evolution**
- **Immediacy**
- **Unpredictable Load**
- **Content Sensitive**

- درجه نفوذ شبکه: قابلیت استفاده از نرم افزار تحت وب در همه جا
- هم زمانی : استفاده چندین کاربر از سیستم به صورت همزمان با توجه به پهنای باند
- کارآیی : در نظر گرفتن کارآیی نرم فزار با توجه به تایم مصرفی آن. این یعنی برنامه باید دارای کارآیی بالایی باشد و در نتیجه زمان انتظار استفاده از دستگاه کاهش می یابد و به کاربر در کمترین زمان ممکن پاسخ بدهد. مثال : دستگاه ATM
- تکامل مداوم : مثلا در زمان کنونی در دستگاه ATM صرفه جویی و پاسخگویی در زمان 3 ثانیه برای ما مطلوب است اما ممکن است در آینده این زمان تغییر کند.
- تحمل بارکاری غیر قابل پیش بینی : تحمل حجم بالای ثبت نام در سیستم را داشته باشد.
- حساس به محتوی بودن : UI / UX چشم انداز و زیبا باشد. مثلا درکی مطلوب از هارمونی رنگ ها را داشته باشد.

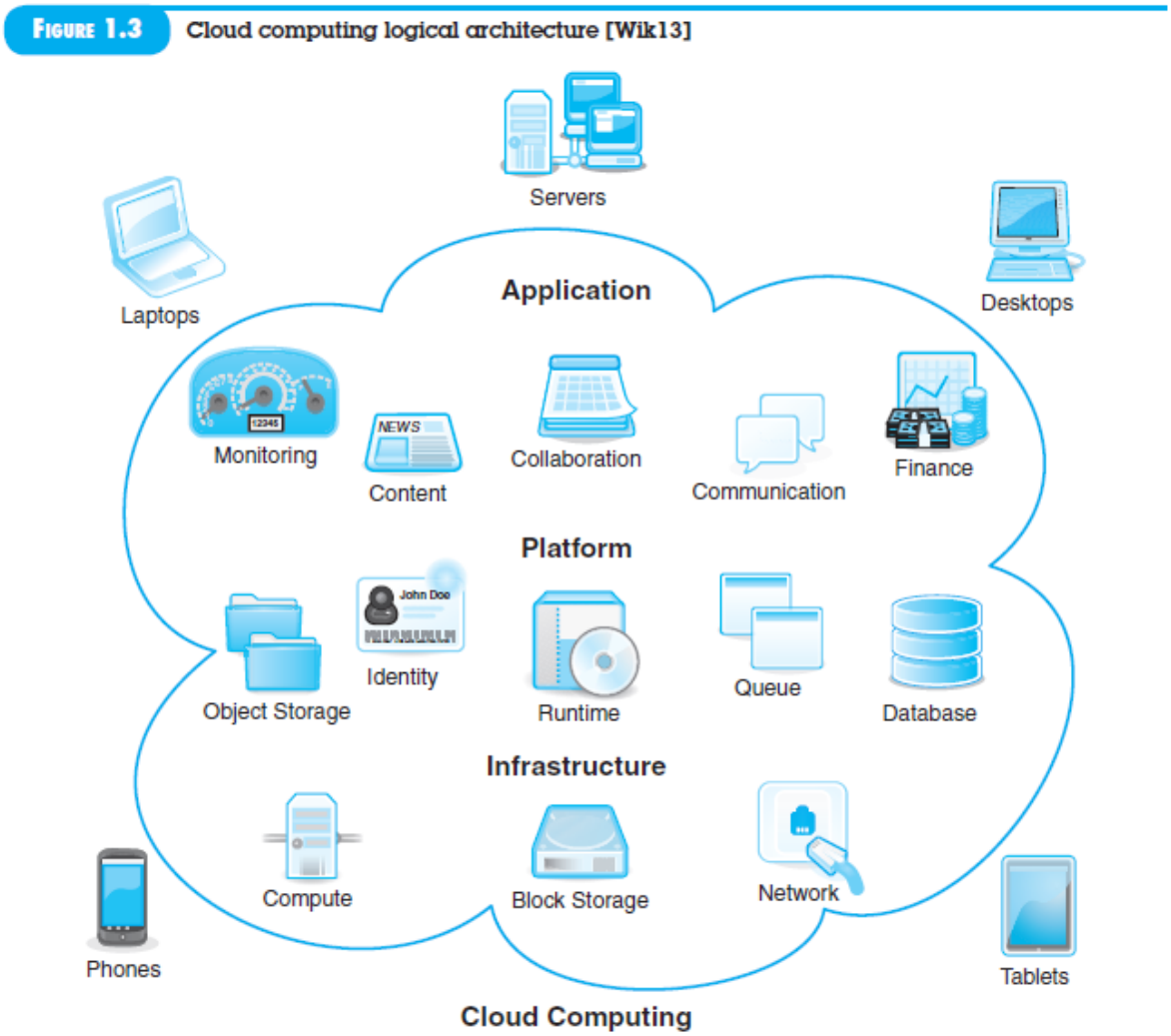


FIGURE 1.3 Cloud computing logical architecture [Wik13]

شکل 3-1 صفحه 10 کتاب

مهندسی نرم افزار Software Engineering (صفحه 14 کتاب)

How do we define software engineering?

دو تعریف مشهور در مهندسی نرم افزار :

تعریف اول : Firsts Bauer : ایجاد و استفاده از اصول مهندسی(Systematic, Disciplined, Quantifiable) جهت بدست آوردن اقتصادی نرم افزارهایی که قابل اعتماد هستند و در محیط های واقعی به صورت کارآمد مورد استفاده قرار میگیرند.

Systematic: ساخت یافته | Disciplined: نظم و انظباط کاری | Quantifiable: قابل اندازه گیری

تعریف دوم : IEEE : Institute of Electrical and Electronic Engineering

1. کاربرد اصول مهندسی در مراحل Development, Operation, Maintenance
2. مطالعه تمامی زمینه های مرحله یک

KEY POINT

Software engineering encompasses a process, methods for managing and engineering software, and tools.

Development: توسعه: زمانیکه نرم افزار را در دانشکده نصب می کنیم.

Operation: اجرایی: سیستم را اجرا کنیم و بصورت واقعی در محیط عملیاتی اجرا شود.

Maintenance: نگهداری: تعمیر و نگهداری نرم افزار هنگام بروز مشکلات احتمالی در زمان اجرا.

مثال : فردی دردرس نرم افزار ثبت نام کرده اما نامش در لیست موجود نیست. ممکن است ما حدنصاب ثبت نامی را 40 نفر مد نظر گرفته باشیم اما در زمان ثبت نام نفر 40 + 1 از لیست حذف شده باشد.

همچنین به معنای پشتیبانی در شرایط خاص نیز می باشد. بدین شکل که ثبت نام همزمان افراد تا 12 نیمه شب مجاز باشد.

QUICK LOOK

What is it? Software engineering encompasses a process, a collection of methods (practice) and an array of tools that allow professionals to build high-quality computer software.

Who does it? Software engineers apply the software engineering process.

Why is it important? Software engineering is important because it enables us to build complex systems in a timely manner and with high quality. It imposes discipline to work that can become quite chaotic, but it also allows the people who build computer software to adapt their approach in a manner that best suits their needs.

What are the steps? You build computer software like you build any successful product, by applying an agile, adaptable process that leads to a high-quality result that meets the needs of the people who will use the product. You apply a software engineering approach.

What is the work product? From the point of view of a software engineer, the work product is the set of programs, content (data), and other work products that are computer software. But from the user's viewpoint, the work product is the resultant information that somehow makes the user's world better.

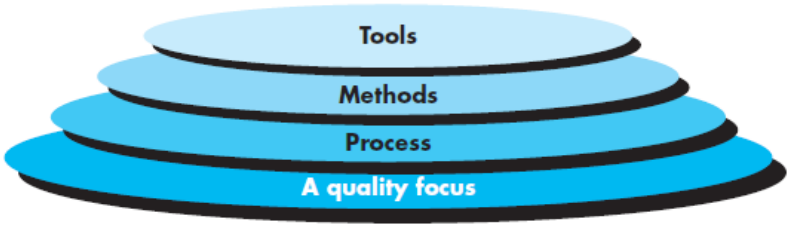
How do I ensure that I've done it right? Read the remainder of this book, select those ideas that are applicable to the software that you build, and apply them to your work.

System Life Cycle

دوره زندگانی نرم افزار (سیستم)

لایه های مهندسی نرم افزار :

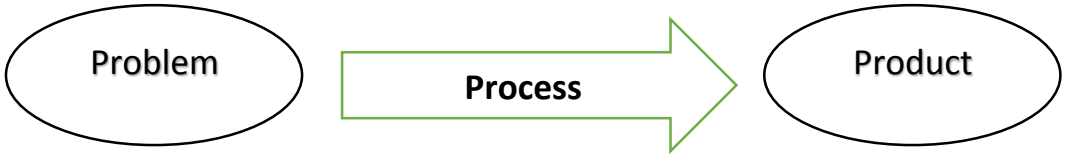
- **A Quality Focus** : توجه ویژه به کیفیت
- **Process** : فرآیندهای تولید نرم افزار
- **Method** : چگونگی انجام کار
- **Tools**: ابزار: بعنوان مثال از چه زبان و دیتابیسی استفاده کنیم.



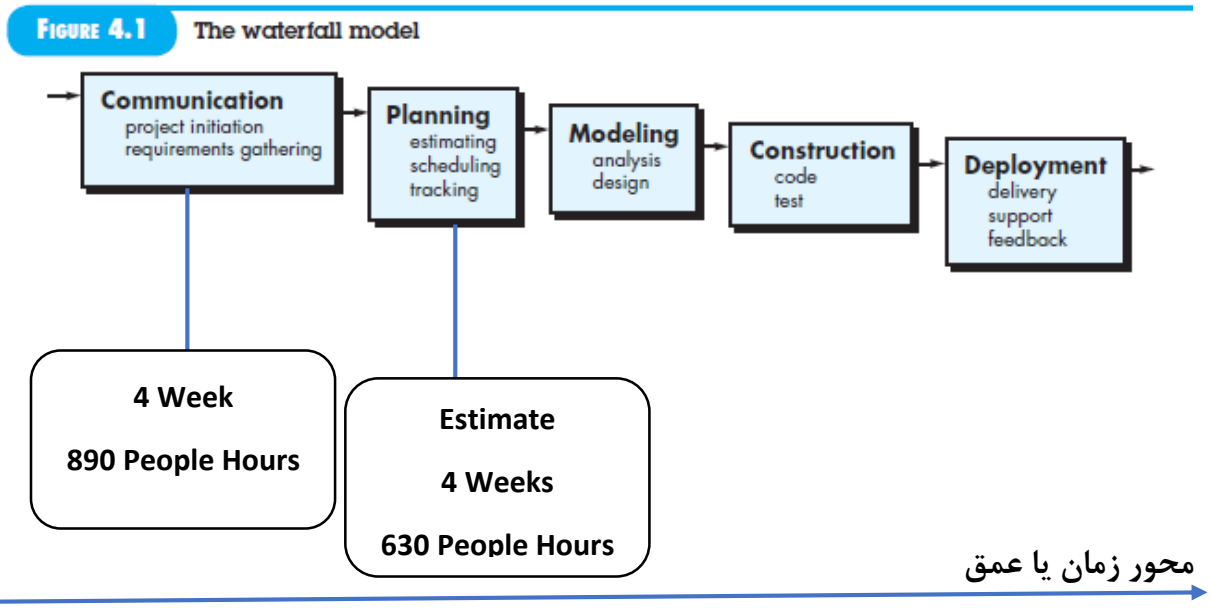
Process Model

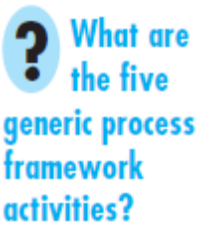
مدلهای فرآیند

منظور از **Process** نرم افزارها مجموعه کارهایی است که باید انجام بدهیم تا نرم افزار موردنظر به وجود بیاید.



نمودار روش آبشاری یا Water Fall





در بحث Communication دو کار زیر انجام میشود :

- راه اندازی پروژه یا Project Initiation
- جمع آوری نیازها یا Requirements Gathering

برنامه ریزی Planning :

- برآورد پروژه Estimate : برآورد پروژه از نظر زمانی مالی نیروی انسانی و ....
- زمانبدی Schedule : زمانبندی بر اساس زمان و دانش مدیریتی.
- پیگیری Track : ارزشیابی سیستم را انجام می دهیم. مسئولیت این کار را بر عهده فرد در نظر میگیریم.

تحلیل Modeling :

- آنالیز Analysis : تحلیل و بررسی سیستم برای نیل به نموداری مبتنی بر حقیقت
- طراحی Design : طراحی الگوریتم ها و نمودارها بر مبنی تحلیل سیستم

ساختن Construction

- کدنویسی Coding : بررسی و انتخاب زبان برنامه نویسی مناسب و نصب سخت افزار مناسب با توجه به شناخت کافی و وافی از سیستم.
- رفع اشکالات Test : بررسی و تست هنگام کار در محیط (نه در مرحله تحویل). سیستم در مرحله ای شبیه سازی شده به واقعیت آزمایش میشود تا اشکالات و معایبش نشان داده شود. این مرحله الزامی و غیر قابل موکول کردن است.

نکته : ارایه نتایج عیب یابی در قالب گزارشات به مدیر پروژه تحویل داده میشود که اصطلاحا Technical Review نام دارد.

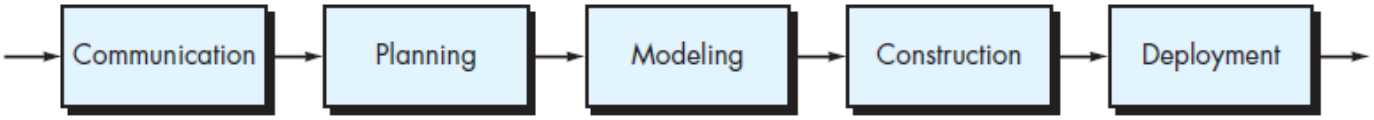
گسترش Deployment

- حمل Delivery : نصب نرم افزار در محیط عملیاتی واقعی.
- بازخورد . پشتیبانی و نگهداری Feedback : فرستادن اعلانات خرابی + نیازهای نو
- حمایت Support : این مرحله بعد از نصب نرم افزار در محیط عملیاتی واقعی می باشد. تعدادی پرسنل برای دریافت آموزشهای لازم از سمت کارفرما تعیین می شود تا با سیستم در تعامل باشند و نیازهای جدید در قالب فرمهایی که به آنها اعطا می کنیم برای ما ارسال کنند. هنگام دریافت گزارش خرابی ها برآورد زمان و نیروی انسانی را برای رفع اشکالات مد نظر قرار می دهیم.

وجه تسمیه نمودار آبشاری یا Waterfall : زیرا در این نمودار مراحل مختلف پشت سر هم هستند و نمی توانیم به مرحله ی قبل برگردیم.

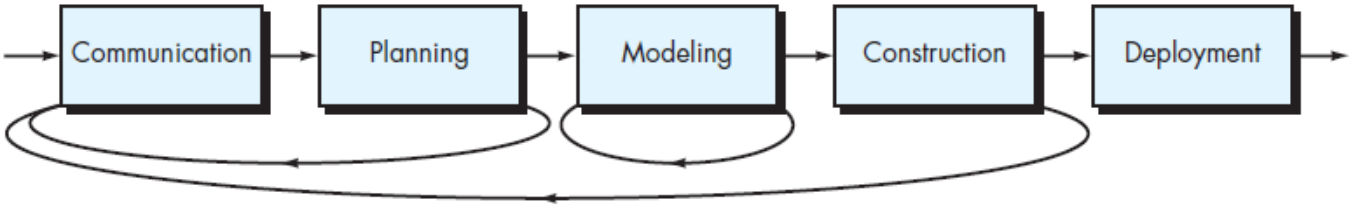
الگوهای تکاملی

روش آبشاری



(a) Linear process flow

روش تکراری : در بسیاری از پروژه ما پروژه را به مشتری تحویل می دهیم مشتری ایرادات و اشکالات موجود در سیستم را پیدا میکند و اطلاع میدهد پس باید به مراحل قبلی بازگشته و کار را اصلاح کنیم.

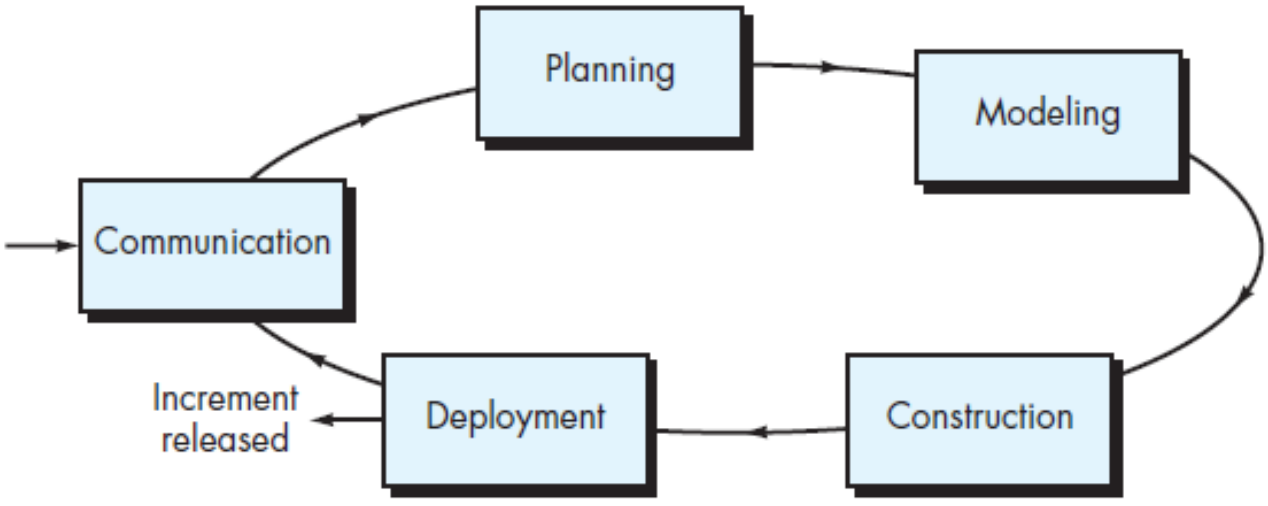


(b) Iterative process flow

KEY POINT

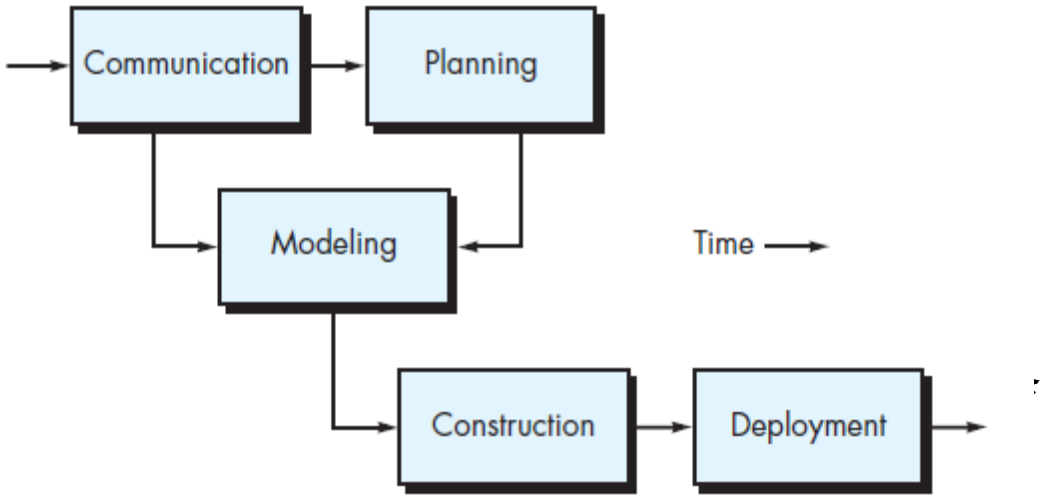
The hierarchy of technical work within the software process is activities, encompassing actions, populated by tasks.

روش تکاملی : تکرار و تکامل در کنار هم می باشند. یعنی هنگامی که مشتری مشکلات خود را بیان میکند و ما به مراحل قبلی بر میگردیم باعث میشود تا منطق برنامه را صحیح تر پیش ببریم در نتیجه نرم افزاری جامع تری ارایه می دهیم.

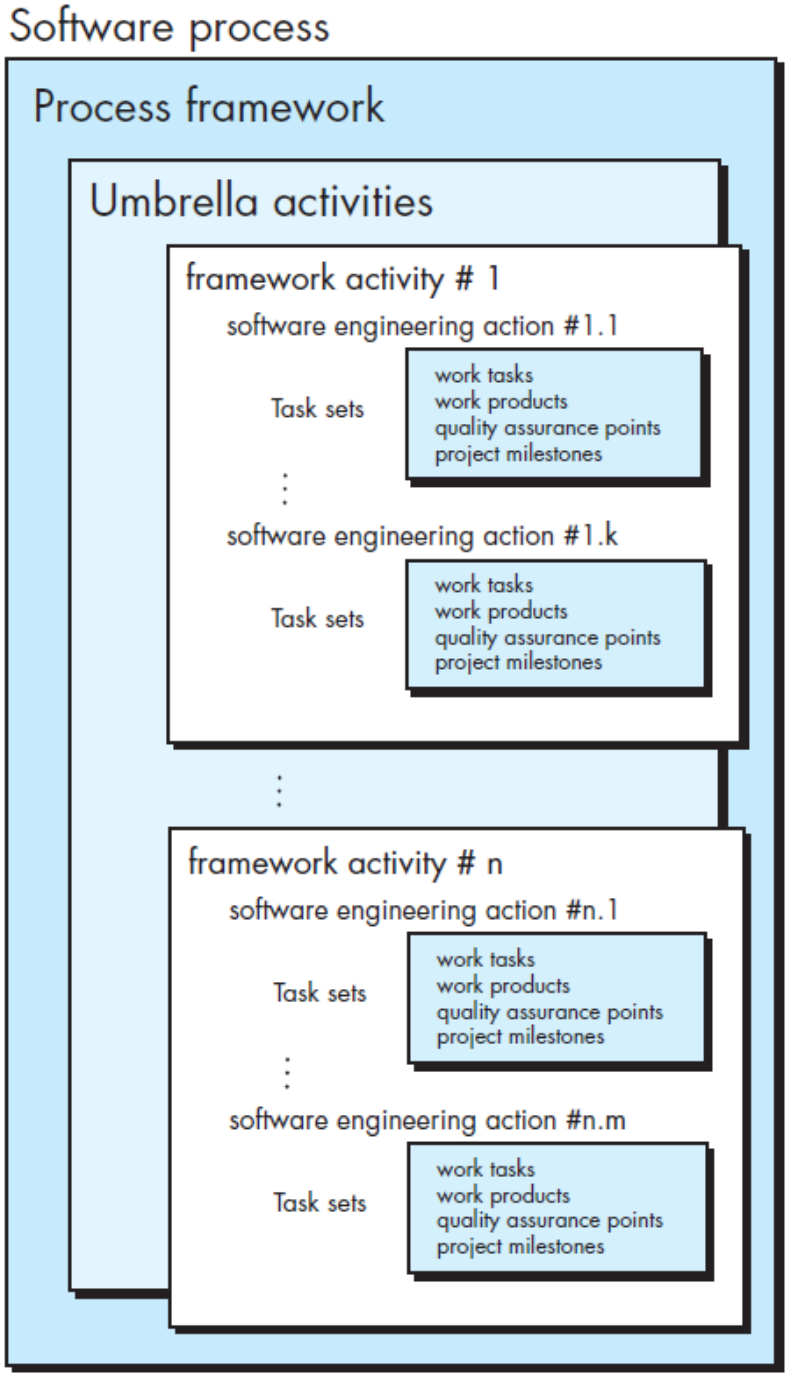


(c) Evolutionary process flow

روش موازی



What is process flow?



شرح جریان پردازش :

مرحله نخست **Process Framework** مراحل آبشاری را نشان میدهد.

مرحله بعدی فعالیت های پشیتبان را نشان می دهد.

در قسمت **Framework Activity** مجموعه وظایف مشخص میشود

Task Set یا جمع آوری نیازها به شرح ذیل است:

فهرستی از افراد موثر در پروژه را تهیه می کنیم

آنها را به جلسات غیر رسمی دعوت می کنیم.

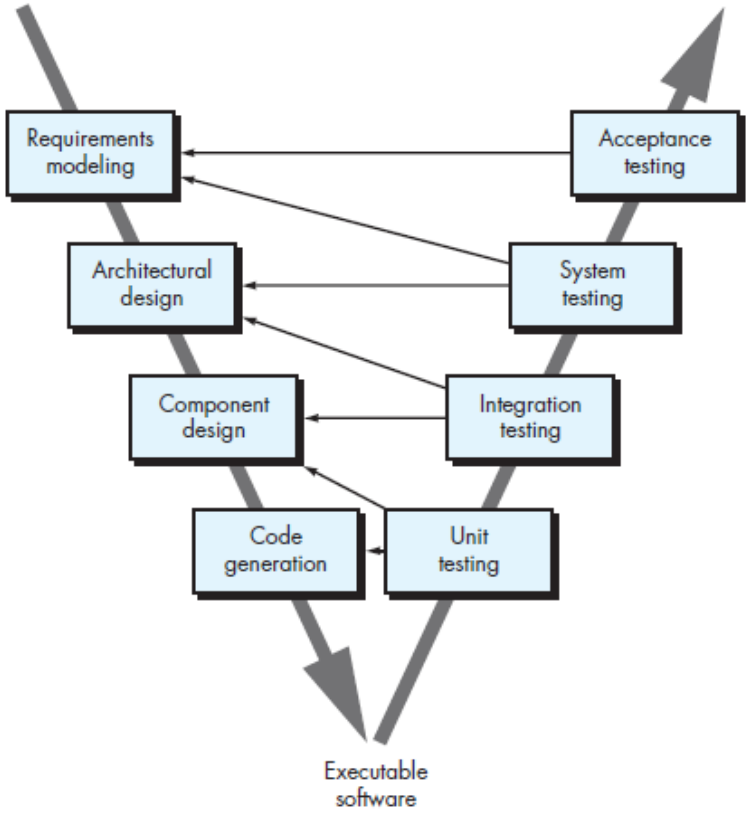
از آنها در رابطه با توقعاتشان از سیستم سوال می کنیم.

لیست ها را به ترتیب اولویت مرتب می کنیم. موارد نا مشخص را شناسایی می کنیم.

11 شکل 1-3 صفحه 32 کتاب

The V-Model

FIGURE 4.2
The V-model



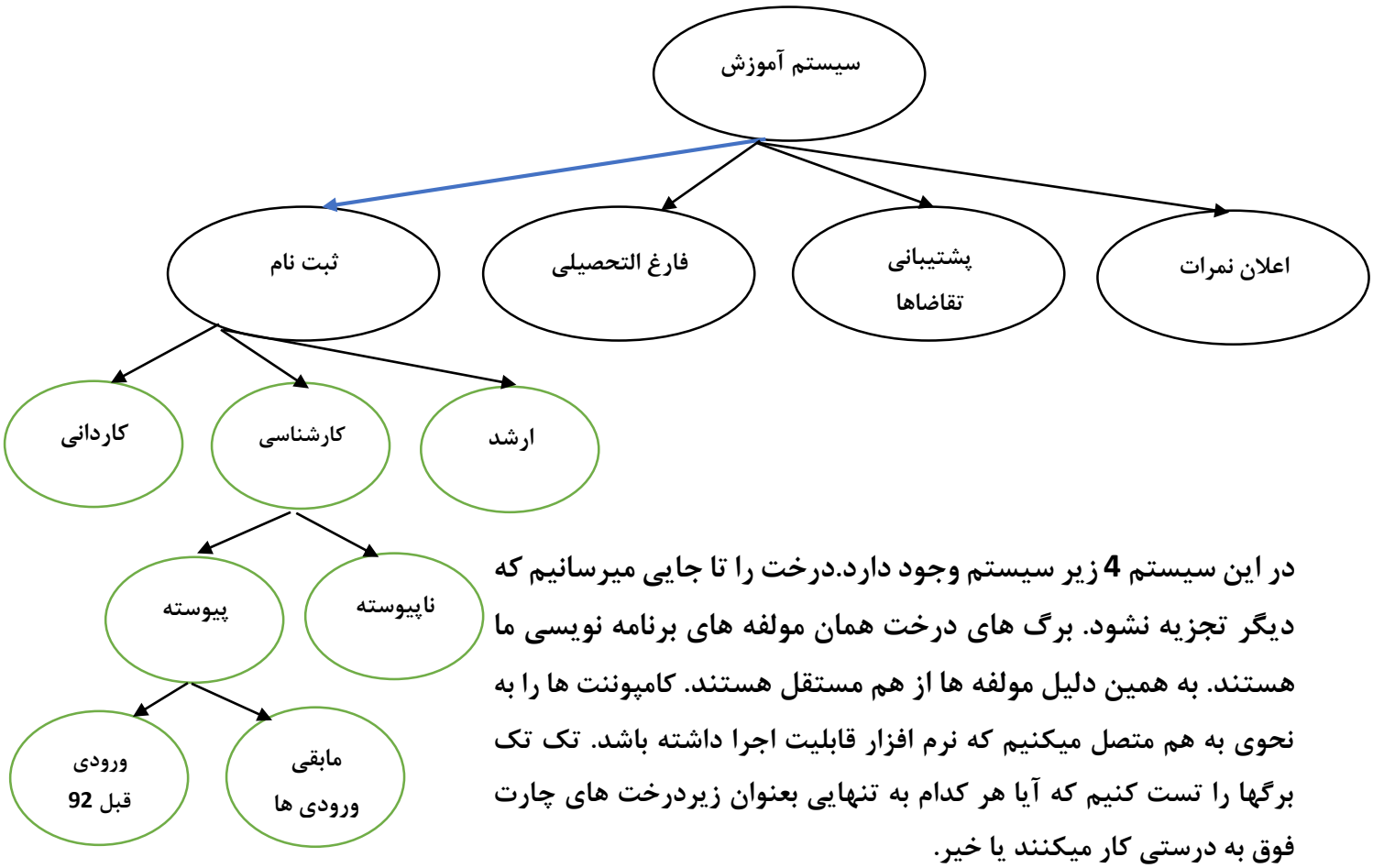
Requirements Modeling	الگوی نیازمندی ها را مشخص میکند. مثال : به **SQL 2014** نیازمندیم
Architectural Design	شرح وظایف بخش های مختلف و ارتباط آنها با هم ابزار : استفاده از **Tree** برای سیستم آموزش
Component Design	برگ های درخت را می نامند : **Leaf** نقطه شروع در برنامه نویسی فرعی نام دارد هر برگ یک **ER** نیاز دارد.
Code Generation	تولید یا نوشتن کد
Execution Code	کامپوننت ها را به نحوی به هم متصل میکنیم که نرم افزار قابلیت اجرا داشته باشد.
Unit Testing	تک تک برگ ها را تست کنیم که آیا هر کدام به تنهایی بعنوان زیردرخت های چارت فوق به درستی کار میکنند یا خیر !
Integration Testing	هر زیر درخت با بالایی اش سنجیده می شود
System Testing	کل سیستم به یکباره تست میشود
Accept Testing	قابل قبول بودن عملکرد نرم افزار در مرحله تست

نکته : مدل V حالت خاصی از مدل آبشاری است.

قصد طراحی یک سیستم را داریم. اولین مساله خروجی ما خواهد بود.(Out Put) و یا در اصل ورودی های سیستم باید برای ما محرز شوند(Input) . در نتیجه Process سیستم را استخراج میکنیم. نکته ی مهم بعدی : سیستم موجود چه ایرادهایی داردتا ایرادات مورد نظر کاربران را استخراج کنیم.

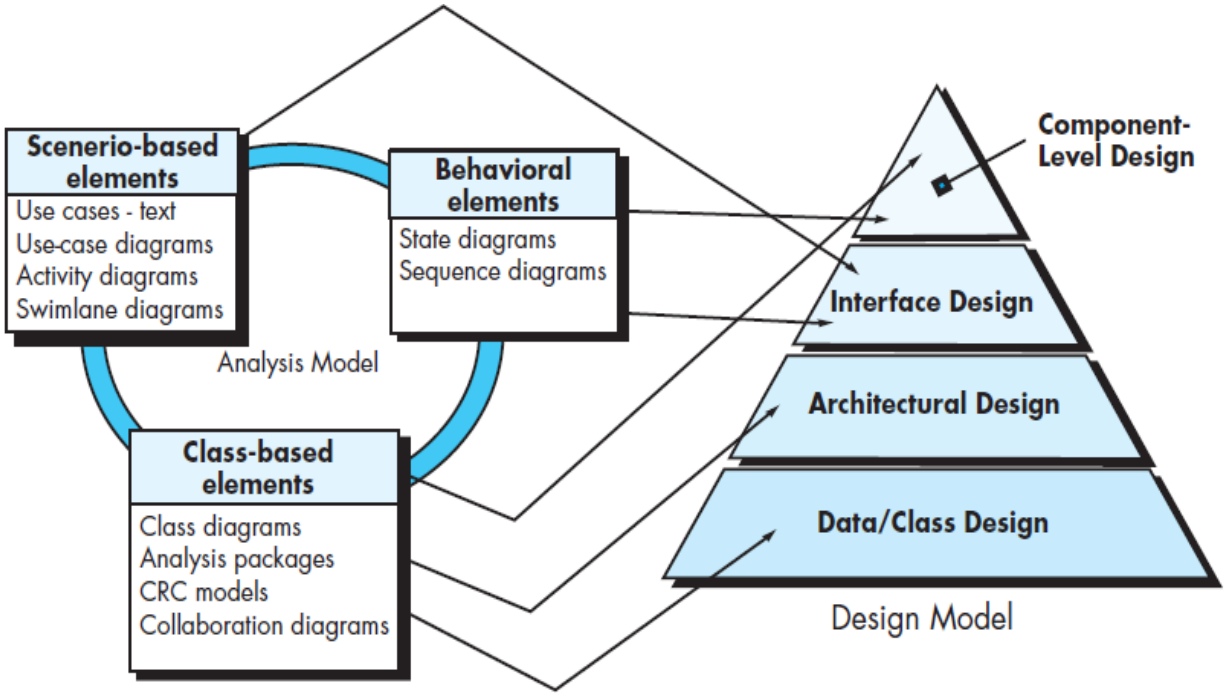
مبحث Elicit اطلاعات : به معنای استخراج و استنباط کردن است. این یعنی از مشتری که نیازهای سیستم را بخوبی نمی داند و نمی تواند به خوبی توضیح بدهد نیازمندی های سیستم را مشخص کنیم.

تجزیه سیستم آموزش : منطق به شرح ذیل است : (این دیدگاه کل به جزء است).



در این سیستم 4 زیر سیستم وجود دارد.درخت را تا جایی میرسانیم که دیگر تجزیه نشود. برگ های درخت همان مولفه های برنامه نویسی ما هستند. به همین دلیل مولفه ها از هم مستقل هستند. کامپوننت ها را به نحوی به هم متصل میکنیم که نرم افزار قابلیت اجرا داشته باشد. تک تک برگها را تست کنیم که آیا هر کدام به تنهایی بعنوان زیردرخت های چارت فوق به درستی کار میکنند یا خیر.

ساختار چهار لایه ای در طراحی



نمودار 1-12 صفحه 226

لایه نخست : برای طراحی نرم افزار ابتدا می بایست داده ها را شناسایی کنیم و پایگاه داده را طراحی کنیم.

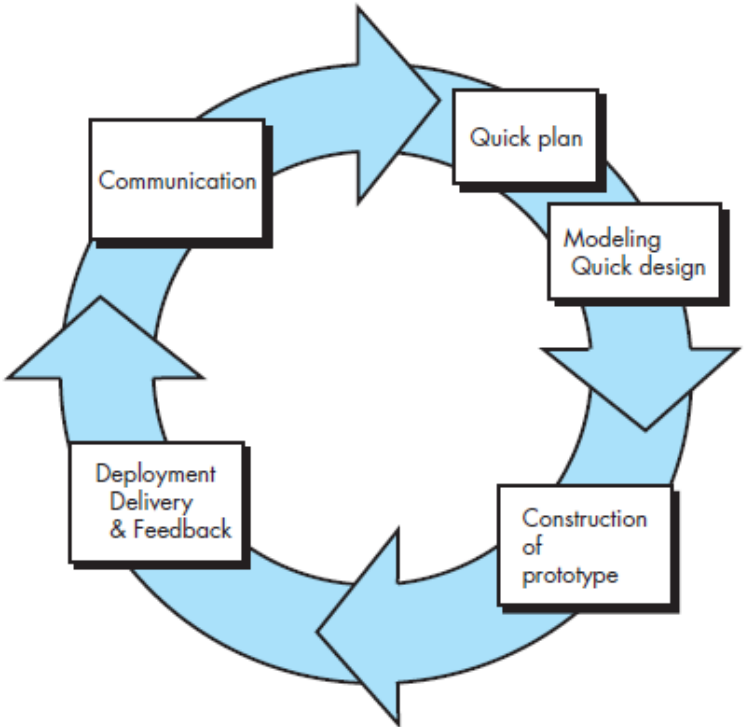
لایه دوم : ارتباطات بین جداول در قالب Entity Relation Diagram (ERD)

لایه سوم : لایه رابط کاربران است. اگر در این مرحله داده ی جدیدی ایجاد شود در درون جداول قرار میگیرد.

لایه چهارم : آخرین مرحله در امر طراحی نرم افزار است.

الگوهای تکاملی

روش نمونه سازی Prototyping Model



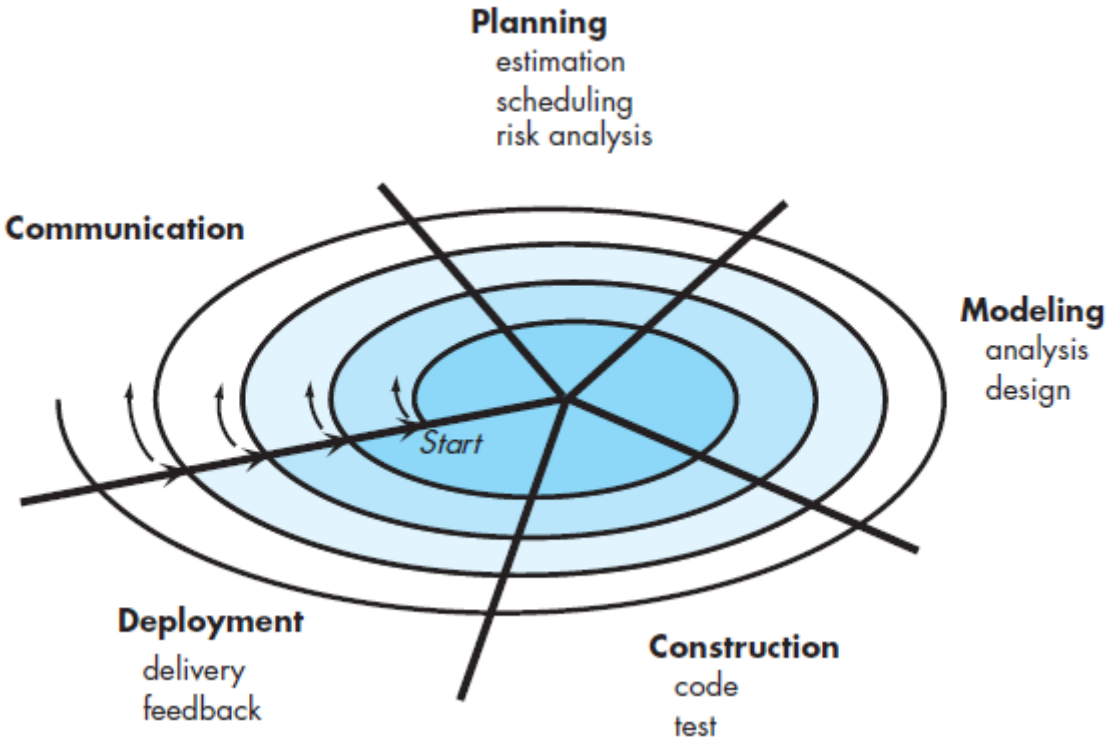
نسبت به نام اولیه تاکید داریم. مراحل این روش را در شکل مشاهده فرمایید.

ویژگی های این روش را در ادامه می بینیم :

- مشتری نحوه کار نرم افزارش را مشاهده میکند.
- این روش طبیعتا کند است بدین جهت که تکاملی است.

دیدگاه ناقص : هر چه مشتری می گوید تیم طراحان متوجه می شوند اما با جدول آنها تطابق ندارد و در صورت مد نظر گرفتن درخواست های او جداول تغییر میکنند. بدین جهت سعی بر تغییر نظرش می نهیم اما در مراحل بعدی مشتری بر درخواست خود استوار است و انتظار برآورده شدن نیازهای خودش را دارد و این دلیل معیوب بودن این روش خاصه است.

روش حلزونی Spiral



شکل 5-4 صفحه 78 کتاب

دور نخست : "داخلی ترین" دور پروژه است و به توسعه فهم مساله می پردازد.

دور دوم : به طراحی جدید می پردازد.

دور سوم : به بهبود مساله می پردازد.

دور چهارم : به تعمیر نگهداری و پشتیبانی پروژه می پردازد.

ویژگی های روش تکاملی (نمونه اولیه و حلزونی)

- طبیعت کندی دارند.
- خود تکامل از سرعت تکامل مهمتر است.
- انعطاف و قابلیت گسترش از نظر اهمیت بر کیفیت خوب ارجحیت دارد.

نکته : اصل در مهندسی نرم افزار کیفیت است اما در این روش ها انعطاف و قابلیت گسترش ارجحیت دارد. از آن جهت که این مدل طولانی است برای پروژه های بزرگ مورد استفاده قرار میگیرد بر خلاف مدل آبشاری که برای پروژه های کوچک مورد استفاده قرار میگیرد.

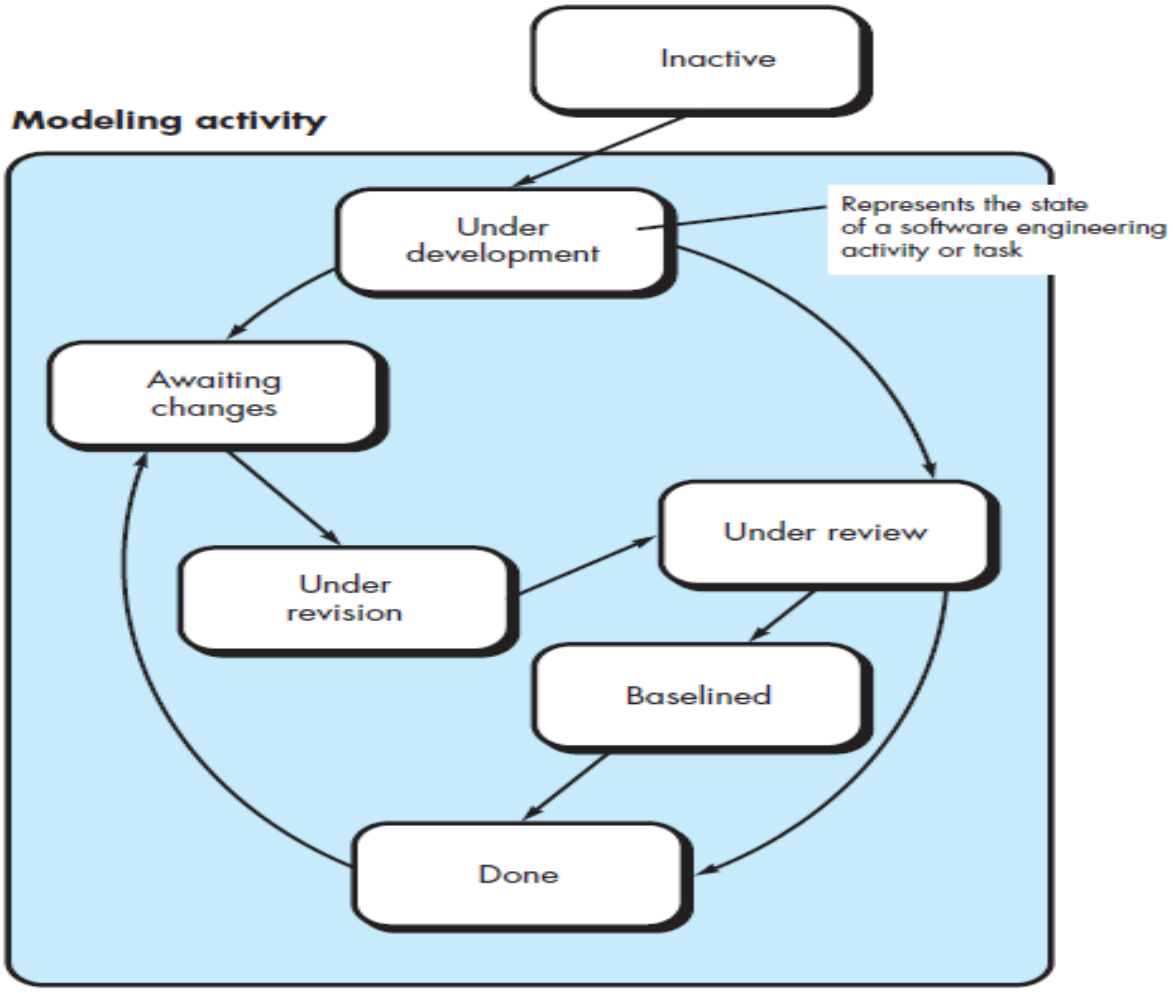
مدل همزمانی Concurrency

امکان همزمانی انجام چندین کار در پروژه را همزمانی می نامند.

مزایا :

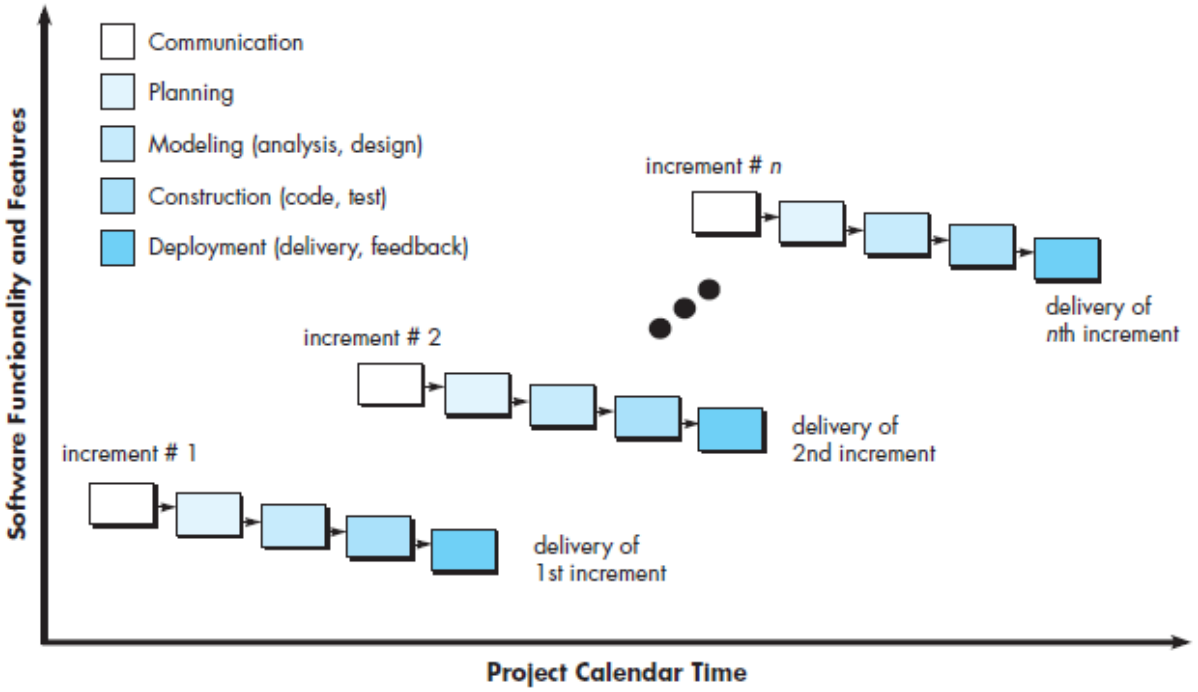
- مدت زمان پروژه کاهش می یابد.
- تاثیر مثبتی بر زمان مصرفی و هزینه های پروژه دارد.

پیش نیاز : مدیریت قوی . تجربه . معلومات بالا



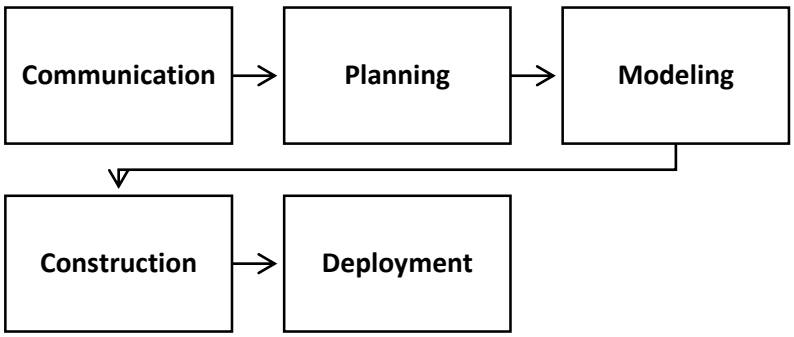
تفاوت بین Under Review & Under Revision : هنگامی که از گزارشات اشکالات مطلع شدیم اصطلاحات را در قسمت Under Revision اعمال می کنیم . یعنی جزییات و زیر سیستم ها را اصطلاح می کنیم. در کنار جزئیات ما یک دید سراسری نسبت به سیستم داریم و اگر نیاز به اصطلاحی در کل سیستم باشد اعمال می کنیم. این همان Under Review می باشد.

روش افزایشی



شکل 3-4 صفحه 44

یادآوری : منظور از فرآیند مجموعه کارهایی است که باید انجام دهیم تا نرم افزار مورد نظر حاصل شود. الگوی اصلی شامل موارد ذیل است :



یادآوری : الگوی نخست آبشاری بود و الگوی بعدی V . با توسعه روش آبشاری جواب بسیاری از مسائل برای ما روشن شد. نام این روش جدید مدل افزایشی است.

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.

مثال : طراحی سیستم دبیرخانه

این سیستم در طول روز می تواند 5000 کاغذ را ساپورت کند. به تجهیز تیم می پردازیم و مراحل کار را شروع می کنیم. با گشت و گذار در سایر کمپانی ها به این نتیجه می رسیم که چرا سقف ساپورت روی 15000 ارتقاء ندهیم! می توانیم گروه دومی را تجهیز کنیم تا با سقف 15000 برسد ولی کار را قطع نمی کنیم و تغییرات جدید را اعمال می کنیم. گروه سومی را برای سقف 50000 گردش کار در طول روز مجهز می کنیم.

سوال : این پروژه برای چه افرادی کاراست ؟

پاسخ : برای افرادی که فاز Investment را سپری کرده و تجربه کافی در انجام پروژه های موازی را درکارنامه خود دارند.

مزیت : می توان از زیر برنامه های یک گروه برای سایر گروه های دیگر بهره مند شد.

سوال : گروه دوم و سوم به چه منظور ایجاد شد ؟

پاسخ : به منظو پاسخگویی به تقاضاها و نیازهای جدید

سوال : آیا امکان بازگشت به مراحل قبل در این نمودار وجود دارد ؟

پاسخ : این ممکن با استفاده از ابزارهای پیاده سازی نظیر C# & SQL مقدور است.

یک چالش : شاید در ابتدای کار سه گزارش داشته باشیم اما در انتها 10 گزارش. پس ابتدای و انتهای پروژه با هم متفاوت است. اولویت ابزاری انتخاب ابزار گران نیست بلکه ابزار می بایست قابلیت هماهنگی با پروژه را دارا باشد.

SOFTWARE TOOLS

Process Management

Objective: To assist in the definition, execution, and management of prescriptive process models.

Mechanics: Process management tools allow a software organization or team to define a complete software process model (framework activities, actions, tasks, QA checkpoints, milestones, and work products). In addition, the tools provide a road map as software engineers do technical work and a template for managers who must track and control the software process.

Representative tools:[12]

GDPA, a research process definition tool suite, developed at Bremen University in Germany (**www.informatik.uni-bremen.de/uniform/gdpa/home.htm**), provides a wide array of process modeling and management functions.

ALM Studio, developed by Kovair Corporation (**http://www.kovair.com/**) encompasses a suite of tools for process definition, requirements management, issue resolution, project planning, and tracking.

ProVision BPMx, developed by OpenText (**http://bps.opentext.com/**), is representative of many tools that assist in process definition and workflow automation.

A worthwhile listing of many different tools associated with the software process can be found at **www.computer.org/portal/web/swebok/html/ch10**.

Unified Process Model

Unified به معنای مشترک و یکسان است.

مثال : دیدگاه کل به جز نسبت به سیستم آموزش

در این تفکر کار تقسیم بندی میکنیم اما به **Data** بی توجه هستیم. پس داده در اینجا خنثی است. **Object** به شکل روبرو نمایش می یابد.

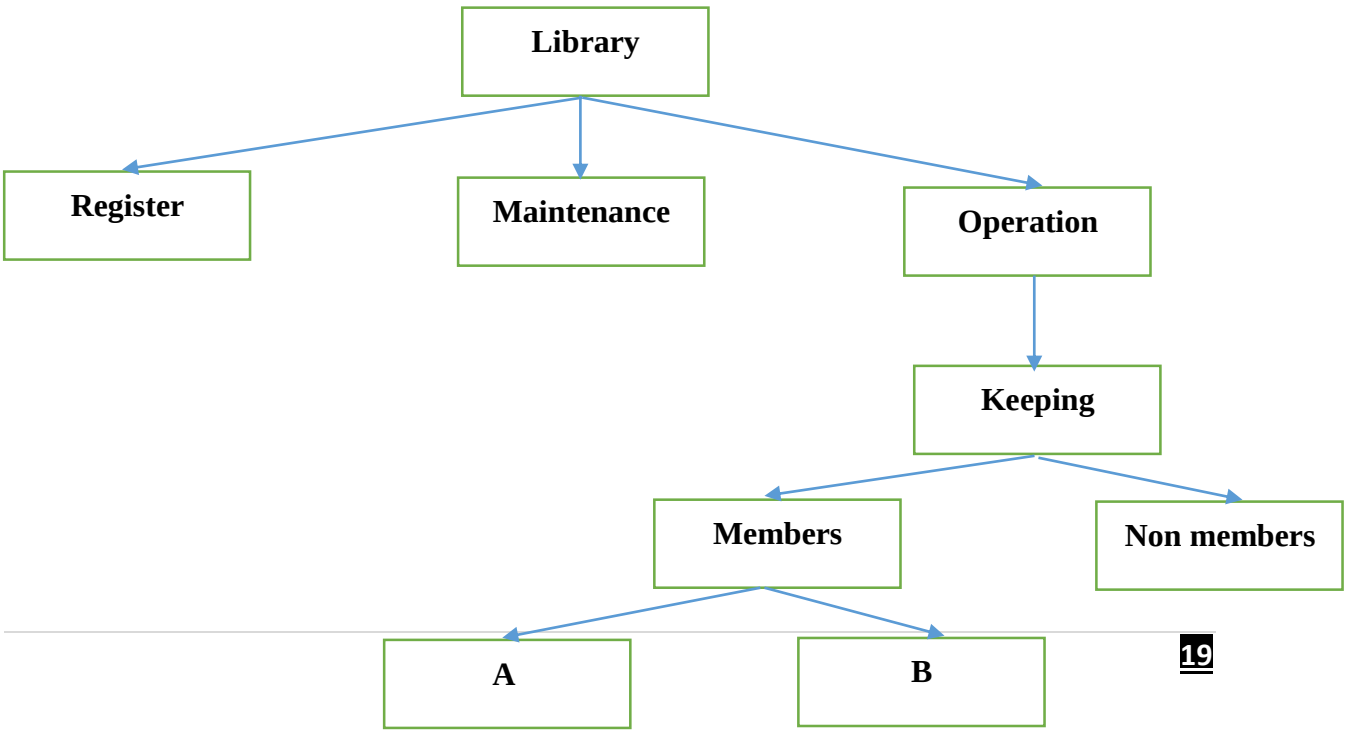
- **Object**
  - **Data**
  - **Operation**
  - **Work**

Stack
Attribute
Method

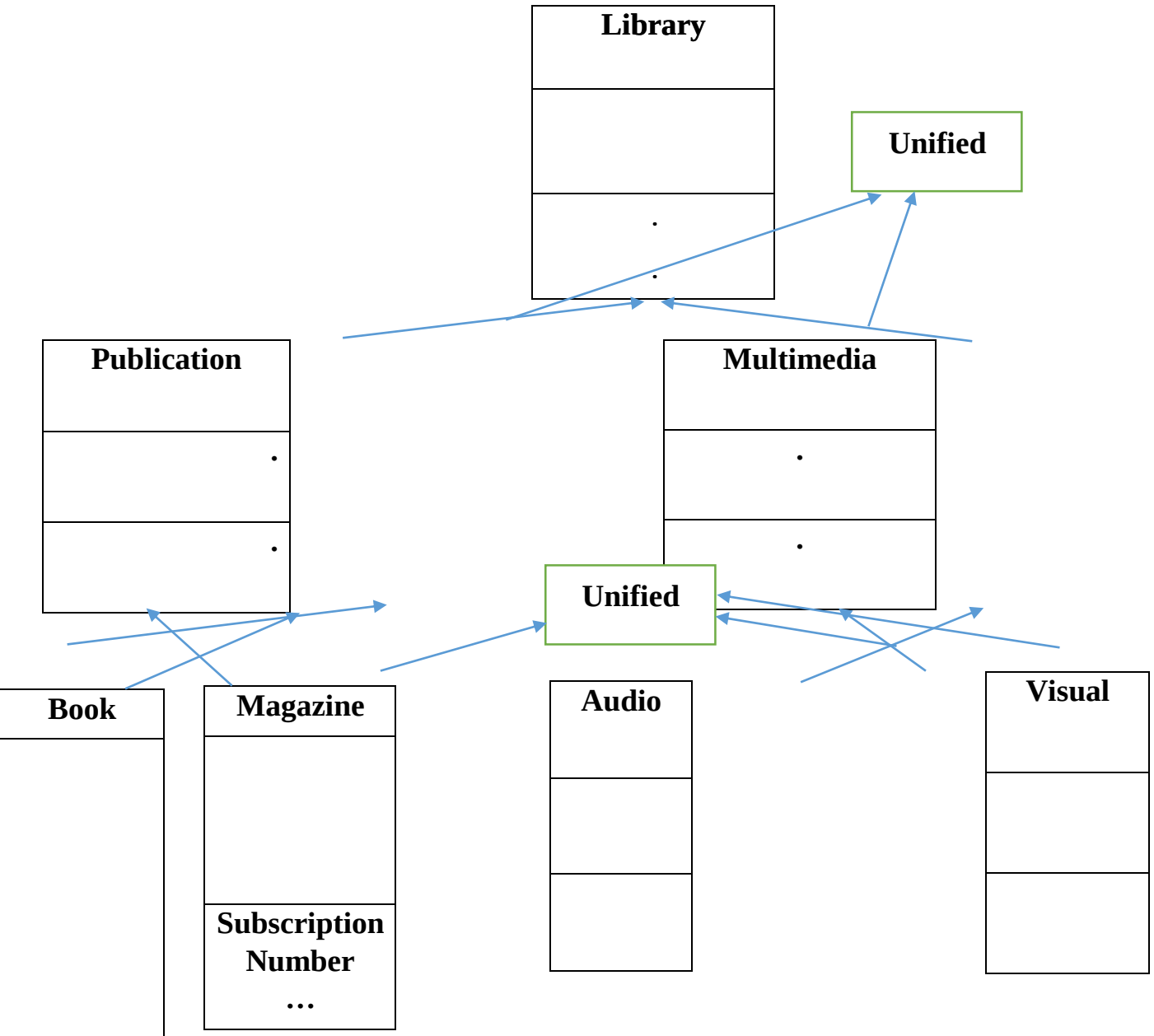
مثال : **Stack** نمونه یک کلاس.

Attribute های **Stack** فیلدهایی است که نیاز داریم.

پیاده سازی در یک سیستم نرم افزاری مثل کتابخانه



تفکر دوم : سیستم کتابخانه از دیدگاه جزء به کل



از Object شروع می کنیم و OO (شی گرایی) را بررسی میکنیم. برای هر Object هم Attribute هم Method را استخراج می کنیم. این نگرش پایگاهی است. در قدم اول اینها را با هم ترکیب می کنیم. سپس یک Super Class داریم. به این فرآیند Unified گفته میشود. یعنی Book & Magazine & Newspaper یکی شد و Publication را بوجود آورد و Audio & Visual یکی شد و به همراه Library , Publication را بوجود آورد.

نکته : در Unified خصوصیات مشترک یکبار نوشته میشود.

Unified Modeling Language

بیشتر بدانیم : UML تصاویر بسیار زیادی از سیستم به ما می دهد و در نتیجه تصورات ما از سیستم کامل میشود.

متدولوژی : چگونگی انجام کار مشخص میشود. تفکر سیستم هم می تواند کل به جزء باشد و هم می تواند جزء به کل باشد. پس از انتخاب متد باید ابزار مرتبط با آن را نیز انتخاب کنیم.

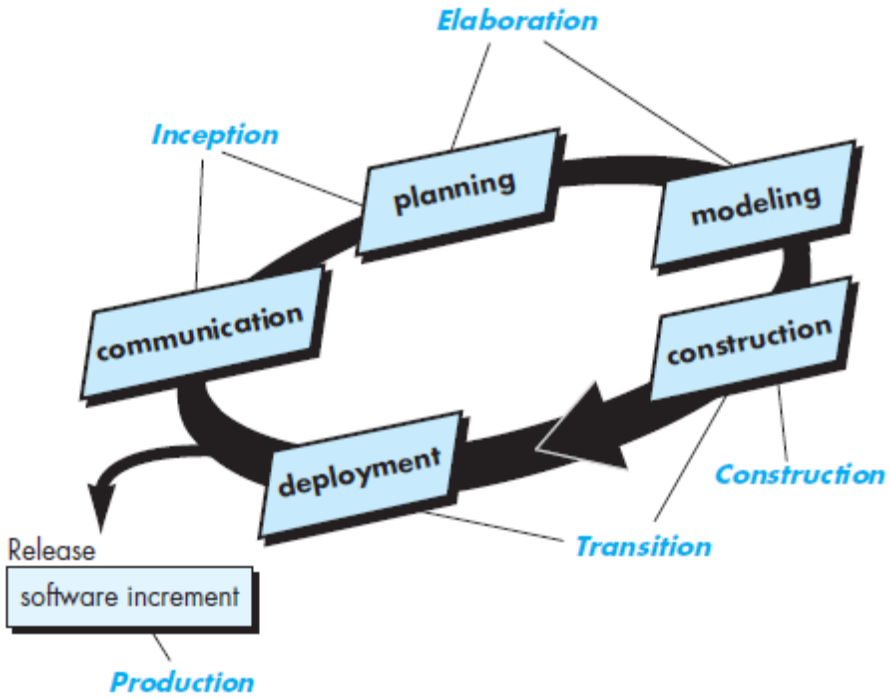
نتیجه گیری : Tools باید بعد از Method انتخاب شود.

KEY POINT

UP *phases* are similar in intent to the generic framework activities defined in this book.

زبان میزبان : Host Language

باید طوری انتخاب شود که با DBMS تطابق داشته باشد.

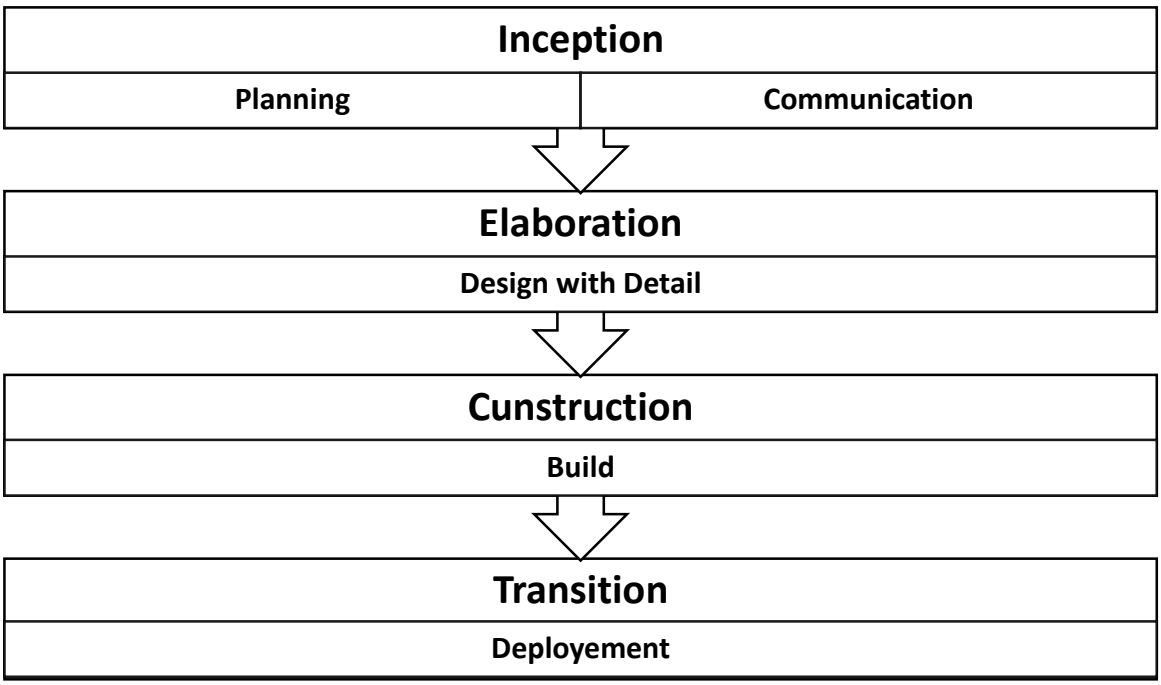


شکل 7-4 صفحه 57

WebRef

An interesting discussion of the UP in the context of agile development can be found at **www.ambysoft.com/unifiedprocess/agileUP.html**.

مراحل روش UP



- ✓ نکته : **Inception** معادل **Planning & Communication** می باشد.
- ✓ نکته : **Elaboration** یعنی طراحی را می بایست به همراه جزئیات سیستم نمایش دهیم.
- ✓ نکته : **Construction** به معنای ساختن است.
- ✓ نکته : **Transition** به معنای انتقال یک مفهوم از تفکر به یک چارچوب دیگر است.

SOFTWARE TOOLS

Process Modeling Tools

Objective: If an organization works to improve a business (or software) process, it must first understand it. Process modeling tools (also called *process technology* or *process management* tools) are used to represent the key elements of a process so that it can be better understood. Such tools can also provide links to process descriptions that help those involved in the process to understand the actions and work tasks that are required to perform it. Process modeling tools provide links to other tools that provide support to defined process activities.

Mechanics: Tools in this category allow a team to define the elements of a unique process model (actions, tasks, work products, QA points), provide detailed guidance on the content or description of each process element, and then manage the process as it is conducted. In some cases, the process technology tools incorporate standard project management tasks such as estimating, scheduling, tracking, and control.

Representative tools:[20]

Igrafx Process Tools—tools that enable a team to map, measure, and model the software process (**http://www.igrafx.com/**)

Adeptia BPM Server—designed to manage, automate, and optimize business processes (**www.adeptia.com**)

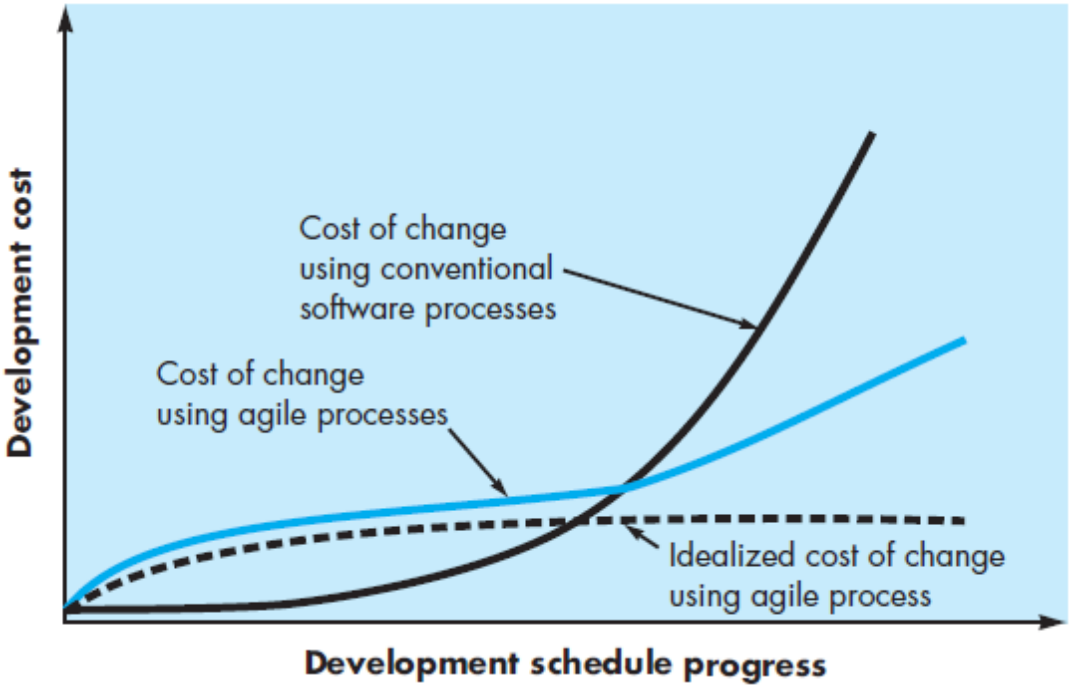
ALM Studio Suite—a collection of tools with a heavy emphasis on the management of communication and modeling activities (**http://www.kovair.com/**)

حجم پردازش ها در حال ازدیاد است و می بایست از این افزایش بکاهیم ولی نمی خواهیم کیفیت را از دست بدهیم.

چاره کار کجاست ؟ می بایست از کارهای اضافی بکاهیم.

اما کار اضافه چیست ؟ در زمان کار کارهایمان را انجام می دهیم بدون کاهش کیفیت.

در واقع حجم Process هایمان را کم می کنیم. پس می بایست بر معلومات خود بیفزاییم و کار اضافی انجام ندهیم چرا که متاسفانه کار اضافی در ذات ماست.



شکل 1-5 صفحه 69 کتاب

- ✓ **هر چه جلوتر می رویم هزینه ها افزایش می یابند.**
- ✓ **اگر تغییرات در مراحل اولیه باشد هزینه ها کاسته میشود.**
- ✓ **اگر در فاز Planning تغییرات صورت گیرد همه چیز به کلی عوض میشود.**
- ✓ **در صورت تغییر Activity Diagram هزینه ها افزایش می یابد.**

حالا می خواهیم روش هایی را مطرح کنیم که هزینع تغییرات را مدیریت کرده و از آن میکاهد.

اصول Agility

در اینجا می خواهیم تغییرات را کنترل کنیم . قبل از آن 2 نکته حائز اهمیت وجود دارد :

1. به سختی می توان پیش بینی کرد چه بخش هایی از نرم افزار ممکن است تغییر کند.
2. سیاست های استفاده از روش های همزمانی می تواند کارمدیریت تغییرات را مشکل کند.

12 اصل Agility :

1. بالاترین اولویت برای ما اینست که مشتری از کارهای ما رضایت کافی داشته باشد.
2. از تغییرات استقبال می کنیم حتی اگر با تاخیر بیان شده باشند.
3. کار را طوری پیش می بریم که هر دو سه هفته یکبار بخشی از پروژه را تحویل مشتری دهیم. به این ترتیب از نظرات مشتری مطلع میشویم و مشتری هم از طرز کار پروژه مطلع خواهد شد.
4. سعی می کنیم در طول هفته جلساتی بگذاریم که در آن اعضای تیم پروژه و سایر اعضای موثر در آن حاضر باشند.
   a. مثال : در بسیاری از پروژه ها تیم مالی داریم که بر گردش های مالی نظارت دارد.
   b. اگر گروه خوب کار نکند بچه های نرم افزار فکر میکنند همه حقوق بگیر آنها هستند و همچنین بچه های مالی گمان می برند که اگر نباشند گروه کاری را از پیش نمی برد . فنی ها فکر می کنند همه کارها تحت نظارت آن هاست و گروه بدون آنها هیچ معنایی ندارد.
   c. این جلسات را برگزار می کنیم تا همگی اعضا از سختی کار همدیگر مطلع شوند.
5. Simplicity (The Art of Maximizing the Amount of Work Not Done) Is Essential
   a. این جمله می گوید سادگی اصل ماجراست.
6. بهترین و موثرترین روش روش تبادل اطلاعات ملاقات های Face to Face است.
   a. نکته : عبارت Face to Face نص صریح کتاب Pressman می باشد.
7. بهترین معیار اندازه گیری پیشرفت کار نرم افزاری است که کار میکند.

شاخص های انسانی **Human Factors**

1. **مشخصات اعضای تیم Agile را بررسی میکنیم.**
2. **مشخصات افرادی که در تیم Agile می توانند فعالیت کنند.**

گروه های ما می بایست شاخص های ذیل را داشته باشند :

- ✓ **Competence**
- ✓ **Common Focus**
- ✓ **Collaboration**
- ✓ **Decision Making Ability**
- ✓ **Fuzzy Problem Solving Ability**
- ✓ **Mutual Trust and Respect**
- ✓ **Self-Organization**

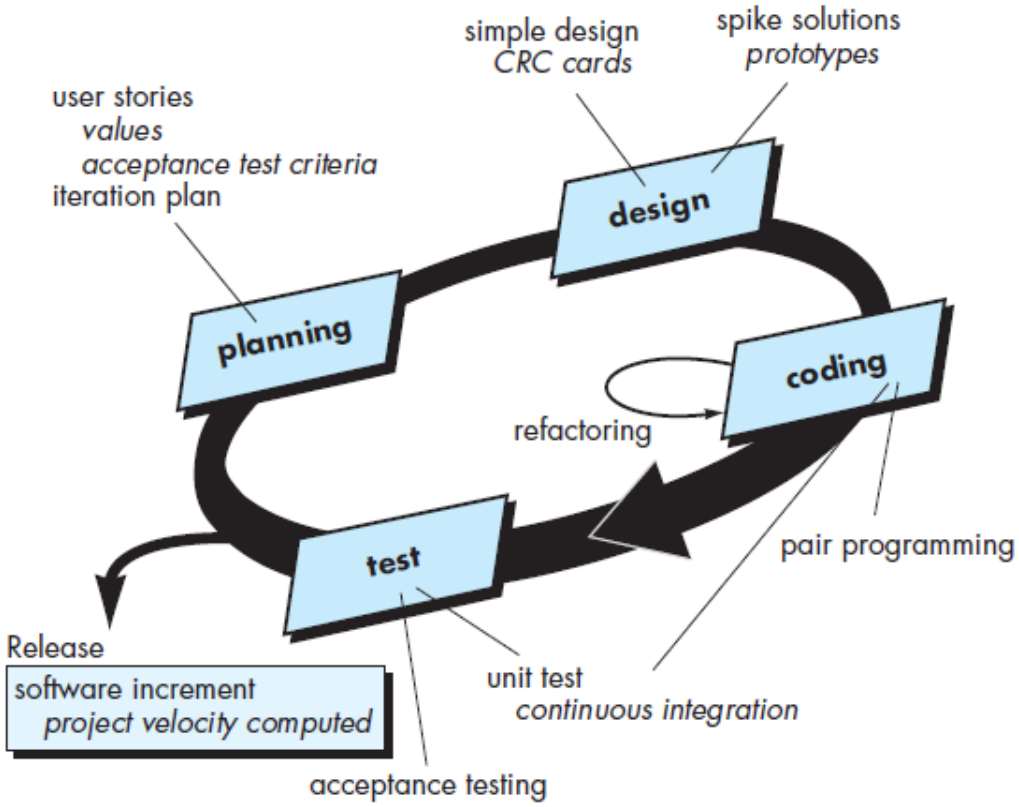
لیاقت
داشتن نقطه نظرات مشترک
داشتن روحیه همکاری
توانایی تصمیم گیری
توانایی حل مسائلی که باشک و تردید همراه است.
احترام و اطمینان متقابل
خود سازماندهی

مثال : در فاز Modeling عده ای مشغول کار هستند و در فاز Construction عده ای دیگر. اعضا گروه Modeling را طوری درست کرده ایم که 4 پروژه بزرگ و مهم در برنامه ی آنان قرار بگیرد و گروه بعدی که در فاز Construction فعال می باشند را از کالج می شناسیم و سیستم آنها عالی بوده است. پس با این سیاست کاری باعث Mutual Trust & Respect (احترام و اطمینان متقابل) میشویم و با همین مدیریت Competence و Common Focus را دربین اعضا تقویت کرده ایم.

در بعضی سازمانها فردی به کمک نفر قبلی میآید که در اینجا 2 حالت ایجاد میشود :

اگر نفر قبلی تصور کند فرد جدید با درست کار کردن و کمک کردن قصد زیرآب زدن را داشته است پس Competence ایجاد نشده است اما اگر فرد قدیمی دانش و عمل بهتری از فرد جدید داشته باشد باعث میشود تا Self-Organization در گروه تقویت شود.

Fuzzy: منطق بیش از 2 مقداری را که هم دانش می خواهد و هم مدیریت را گویند.بعنوان مثال زمانی که ما به فاز planning رسیده ایم اما هنوز DBMS را انتخاب نکرده ایم. پس باید توانایی حل این مساله را داشته باشیم. اینجا مبحث Decision Making Ability کاملا گویاست.



شکل 2-5 روش EXP صفحه 72

- **Simple Design**
- **Pair Programming**
- **Refactoring**

مورد نخست : یک طراحی ساده و روتین هزینه پشتیبانی زیادی ندارد.

مورد دوم : حدالامکان از تکنیک پخشی بهره می بریم و کار ار به چند نفر می سپاریم. این مساله ناقض Dependency خواهد بود و Interactivity را برقرار خواهد کرد.

مورد سوم : حفظ رفتار خارجی یک سیستم و اعمال تغییرات در داخل سیستم

نکات مهم دیگر هم عبارتست از :

- ❖ **Communication**
- ❖ **Simplicity**
- ❖ **Feed Back**
- ❖ **Courage**
- ❖ **Respect**

اگر مراحل کار Communication ضعیف باشد کار را می بایست دوباره پیش ببریم.

اگر چه کنم چه نکنم گویی کنم یعنی کار را بلد نیستم. مثلا در شی گرایی تغییرات ساده میشود.

در مرحله Feed Back بایستی ارزیابی شویم و از اعضای تیم هم در ارزیابی سوال کنیم.

ما به Respect در تمام مراحل کار نیازمندیم : اطمینان به اعضای تیم و انجام کار به نحو احسن.

در روش IXP به نکات ذیل توجه می کنیم :

- Readiness assessment
- Project Community
- Test-Driven Management
- Retrospective
- Continuous Learning

- مورد نخست به ارزشیابی در هر قسمت می پردازد.
- مورد دوم به تمامی افرادی درگیر در پروژه می پردازد.
- مورد سوم تعریف اساسنامه قانونمند کار و هدف گذاری می پردازد.
- مورد چهارم مدیریت بر اساس نتایج آزمونها را مد نظر دارد.
- مورد آخر هم به فراگیری مداوم و پیوسته می پردازد.

ایرادات روش IXP

- Requirement Volatility
- Lack of Formal Design

مورد نخست به نوسانات نیازمندی ها می پردازد که اگر زیاد باشد به مشکل برمی خوریم

مورد دوم هم به فقدان طراحی بر اساس مبنای رسمی می پردازد.

اگر میان نیازمندی های مشتری تناقض باشد این روش دچار مشکل می شود.

سایر روش های Agile - ASD

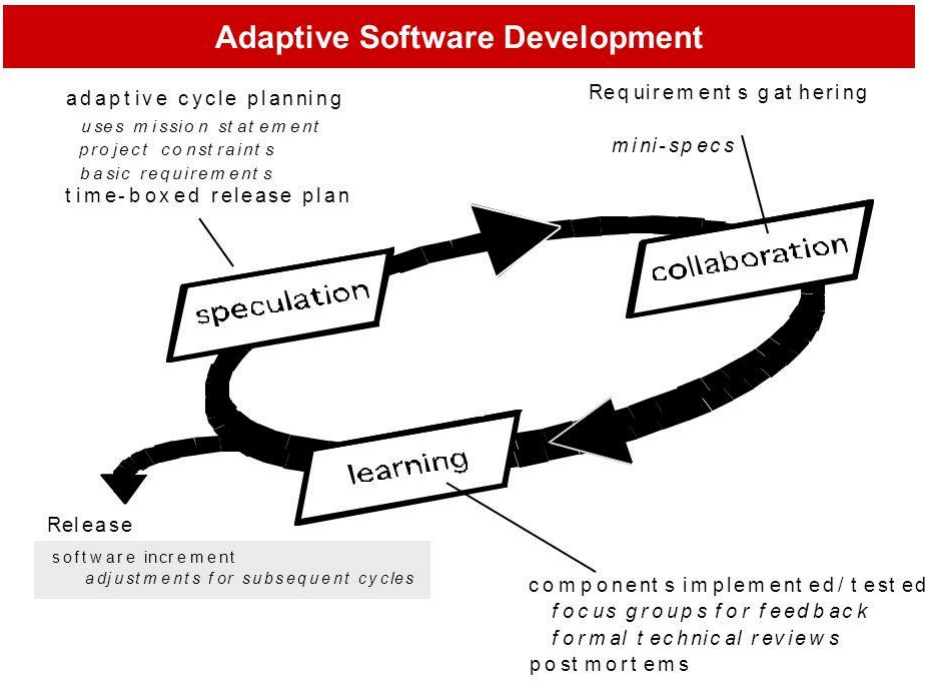
Adaptive Software Development – ASD

ساده ترین نوع مراحل طراحی در نرم افزار فکر کردن در تمامی فازهاست.

Speculation: تفکر

Collaboration: همکاری

Learning: یادگیری – کار را به مشتری تحویل می دهیم و سپس دوباره این 3 مرحله را برای فازهای بعدی انجام می دهیم.



شکل از وب برگرفته شده است.

تیمی که از این روش استفاده می کند را تیم ژله ای نامیده اند.(Galled Team)

مشخصات تیم ژله ای بدین شرح است :

- **اعضای تیم بدون دشمنی با یکدیگر از هم انتقاد می کنند.**
- **بدون محدودیت به یکدیگر یاری رسانی می کنند.**
- **به سختی کار می کنند و بر درجه سختی می افزایند.**
- **بخوبی با مساله ارتباط برقرار می کنند و مهارت انجام کار را دارند.**

روش DSDM : Dynamic Software Development

مراحل اصلی این روش عبارتست از :

- **Feasibility Study**
- **Business Study**
- **Functional Model Iteration**
- **Design and Build Iteration**
- **Implementation**

KEY POINT

DSDM is a process framework that can adopt the tactics of another agile approach such as XP.

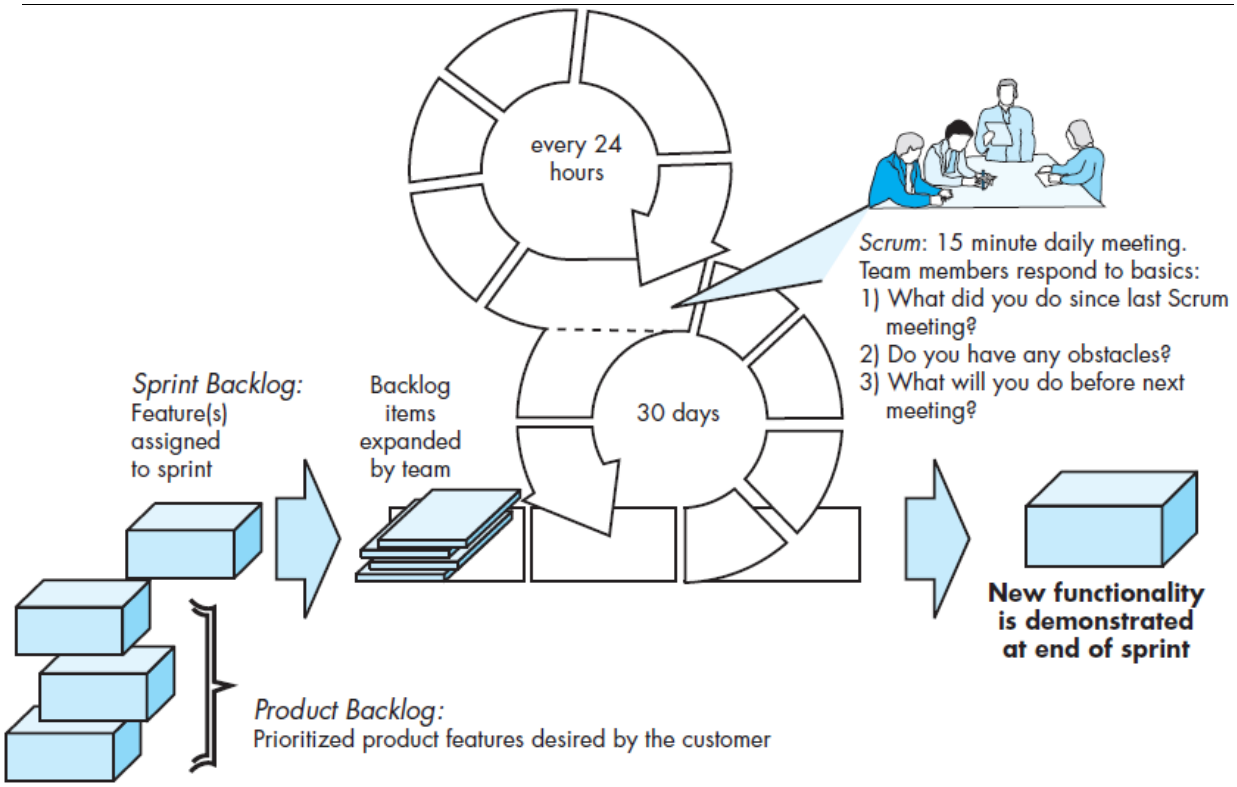
موارد پنج گانه فوق به نکات ذیل اشاره دارند :

- مطالعه امکان سنجی
- مطالعات حرفه ای
- نمایش مدل به مشتری برای اخذ تاییدیه
- انواع دیاگرام ها نظیر ... Use Case , Activity
- پیاده سازی

علت نامگذاری روش فوق به Dynamic چیست ؟

چون مراحل کار شماره گذاری نشده است و می توان تقدم و تاخر را در انجام امور در نظر گرفت.

Scrum



شکل 3-5 : نمودار پردازش Scrum : صفحه 78

لغت Scrum به فعالیتی است که در خلال بازی راگبی انجام میشود و در اصطلاح به معنای گردهم آمدن برا شوری و مشورت می باشد.

نمونه ای از پرسش هایی که در Scrum رخ می دهد :

- What did you do since the last team meeting?
- What obstacles are you encountering?
- What do you plan to accomplish by the next team meeting?

- از دیدار قبل تاکنون چه فعالیت هایی را انجام داده اید ؟
- با چه موانعی مواجه هستید ؟
- از حالا تا جلسه آینده چه برنامه هایی مد نظر دارید ؟
- و ...

WebRef

Useful Scrum information and resources can be found at **www.controlchaos.com.**

سوالات فوق در جلسات مدیریت رخ می دهد که طی موعد زمانی مقرر مثلا **30** دقیقه به مرور هنجارهای پیش رو می پردازیم.

KEY POINT

Scrum incorporates a set of process patterns that emphasize project priorities, compartmentalized work units, communication, and frequent customer feedback.

نکته جالب اینست که در ابتدای جلسه توزیع فرم صورت می گیرد و آنالیز مبحث از طریق پاسخ های اعضا صورت می گیرد. این مساله زمانی رخ می دهد که ناگهان در فاز مدلسازی دچار مشکل شده ایم. بدین منظور یکی از افراد ارشد را به تیم اضافه می کنیم تا کنترل را در دست گیرد و مشکل را حل کند.

جلسات به صورت یک روز در میان خواهد و حدود **15** تا **20** دقیقه به مباحثه می پردازیم.

فرمها باعث میشوند مشکلات گروه در یک بازه زمانی خاص برطرف شوند.

نکته : اعضا تیم و مدیر پروژه باید در جلسات حاضر باشند.

SOFTWARE TOOLS

Agile Development

Objective: The objective of agile development tools is to assist in one or more aspects of agile development with an emphasis on facilitating the rapid generation of operational software. These tools can also be used when prescriptive process models (Chapter 4) are applied.

Mechanics: Tool mechanics vary. In general, agile tool sets encompass automated support for project planning, use case development and requirements gathering, rapid design, code generation, and testing.

Representative tools:[13]

Note: Because agile development is a hot topic, most software tools vendors purport to sell tools that support the agile approach. The tools noted here have characteristics that make them particularly useful for agile projects.

OnTime, developed by Axosoft (**www.axosoft.com**), provides agile process management support for various technical activities within the process.

Ideogramic UML, developed by Ideogramic (**http://ideogramic-uml.software.informer.com/**) is a UML tool set specifically developed for use within an agile process.

Together Tool Set, distributed by Borland (**www.borland.com**), provides a tools suite that supports many technical activities within XP and other agile processes.

Feature Driven Develop Method – FDDM

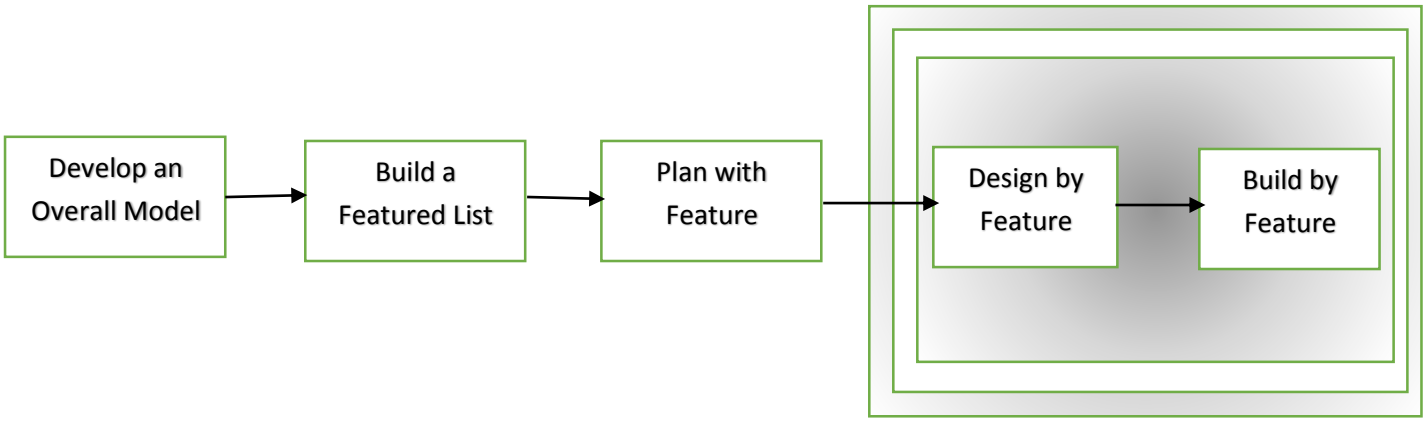
انجمن در تعریف **Feature** چنین آورده است :

Feature is a Client valued Function that can be implemented in 2 week or less

در واقع تابعی ارزشمند برای کاربر می باشد که می تواند ظرف مدت دو هفته یا کمتر پیاده سازی شود.

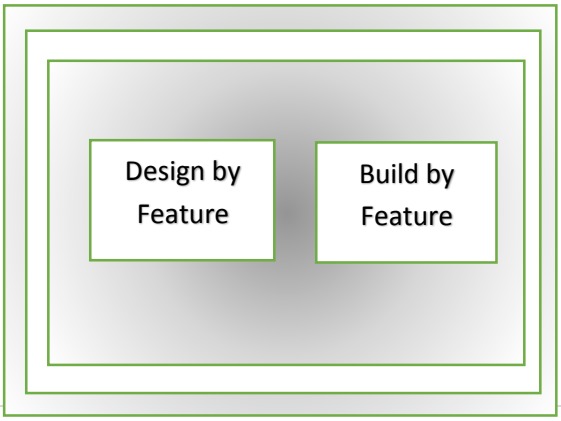
نکته : **Feature** خود سیستم نیست و کاریست اضافی که روی دوش سیستم نهاده ایم.

مراحل کاری این روش به شکل ذیل است :



- اول : تعریف مدل اصلی : طراحی کلی سیستم
- دوم : طراحی فهرستی از **Feature** ها.
- سوم : زمانبدی بر اساس مورد دوم
- چهارم : تعریف صریح تمامی موارد موجود در مرحله دوم یعنی **Feature** ها.
- پنجم : ساخت تمامی موارد موجود در مرحله دوم.

وجود مستطیل های مکرر در مرحله چهارم و پنجم به معنی تکرار می باشد. **(Iteration)**



خصوصیات روش Featured

- **سادگی طراحی Feature ها به دلیل سادگی**
- **امکان جایگذاری در بخش های مختلف سیستم**
- **امکان قرارگیری در ساختار شی گرایی (کل به جزء) در محل های مناسب**
- **سادگی آزمایش**

Agile Modeling

روش مبتنی بر AM

در تعریف آمده است :

Agile Modeling is a Practice Based Methodology for Effective Modeling and Documentation of Software based System.

دو عنصر مهم در این تعریف عبارتند از :

- **Modeling**
- **Documentation**

این روش برای مستندسازی و مدل سازی سیستم های نرم افزاری بکارگرفته میشود.

نکته : Practice Base به تجربی بودن این روش اشاره دارد.

نکات قابل توجه :

- **Model with a Purpose**
- **Use Multiple Models**
- **Travel Light**
- **Content is More Important than Representation**

مورد اول : هدفمندی
مورد دوم : انتخاب مدل های چندگانه برای پروژه به دور از تک مدلی
مورد سوم : حذف ابزارهای غیر ضروری و انتخاب ابزارهای ضروری برای پروژه
مورد چهارم : اهمیت نسبی محتوا نسبت به نوع ارایه نرم افزار با وجود اهمیت نوع نمایش پروژه

نکته : Mr. Pressman در ویرایش هشتم دو مورد دیگر را هم به موارد چهارگانه فوق اضافه کرده که عبارتند از:

- **Know the models and the tools you use to create them**
- **Adapt locally**

مورد پنجم به این مساله اشاره دارد که ما می بایست نقاط ضعف و قوت مدل ها و ابزارهای مورد استفاده را بدانیم.

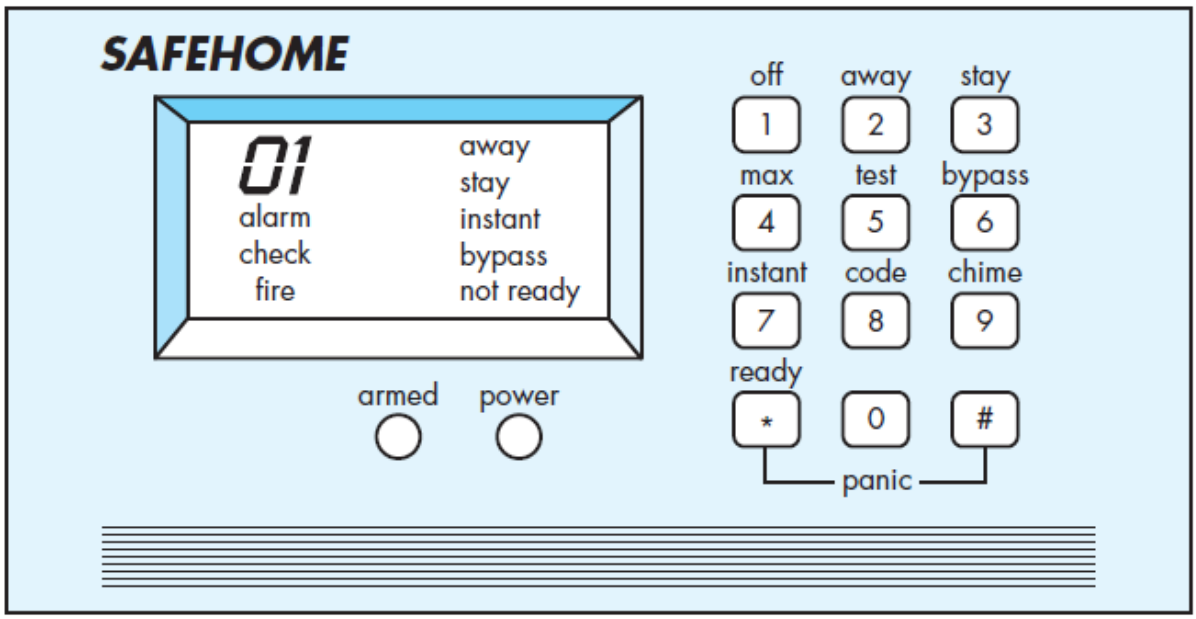
مورد ششم هم به قابلیت سازگاری مدل با توجه به نیازهای تیم Agile اشاره دارد.

Scenario- based	Flow Oriented	Behavioral Elements	Class Based
Use case-text	DFD	State Diagram	Cases Diagram
Use case Diagram	CFD	Sequence Diagram	Analysis Package
Swim Lane Diagram	Processing narratives		CRC Model
			Collaboration Daigram

نکته : در فاز Modeling می بایست چهار نمودار بکشیم.

بررسی فاز Scenario Based Elementsدر فاز Modeling :

گام نخست در Communication : آشنایی با سیستم است. شکل فوق بیانگر اینست که هر کدام از این نمودارها در کدام بخش شکل مثلثی شکل بکار می آید. می خواهیم با نمونه نرم افزارهای دزدگیر آشنا شویم. می خواهیم نرم افزاری بنویسیم که سخت افزارهای مختلف دزدگیر را اداره و کنترل کند. یک کنترل پنل هم داریم.



شکل 1-8 صفحه 159 : Control Panel of Safe Home

در فاز **Communication** با افراد مختلف درباره سیستم صحبت می کنیم و نیازهای دزدگیر را پرس و جو می کنیم. گام بعدی رفتن به سوی طراحی نرم افزار است.

یادآوری : برای درک دیاگرام ها باید با مفهوم **Actor** آشنا شویم.

Actor: عناصری خارج از سیستم اند.

- نرم افزار
- سخت افزار
- ابزار
- نیروی انسانی

وجود سیستم برای نیل **Actor** به هدفش ضروری است.

نکته : **Actor** ها به دو بخش اصلی و فرعی تقسیم میشوند.

نکته : **Actor** فرد یا افرادی است که سیستم به آنها سرویس می دهد.

مثال **Actor** در سیستم داروخانه : دکتر داروساز . بیمار . کارمند . رییس بخش . رجوع کنندگان به داروخانه و شخصی که نسخه را برای تحویل دارو بررسی میکند.

نکته : دارو و نسخه **Actor** نیست و **Data** به حساب می آیند.

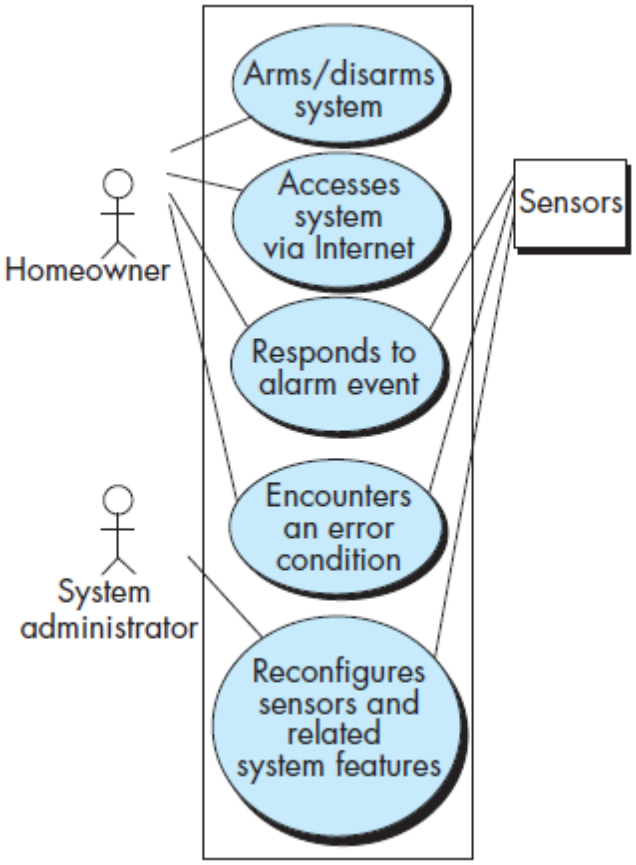
نکته : توقعات **Actor** همان **Use Case** های سیستم ما محسوب می شوند.

- **Arm / Disarm Sys**: فعال یا غیر فعال کردن سیستم
- **Access Sys Via Internet** : اتصال به سیستم از طریق وب
- **Respond to Alarm Event** : پاسخ دادن به هشدار
- **Encounter Error Condition** : رویارویی با اشکال و تصمیم مجدد
- **Reconfigure Hardware of the Sys**: پیکره بندی مجدد سیستم از نو.

نکته : دو **Actor** اصلی داریم و یک **Actor** فرعی.

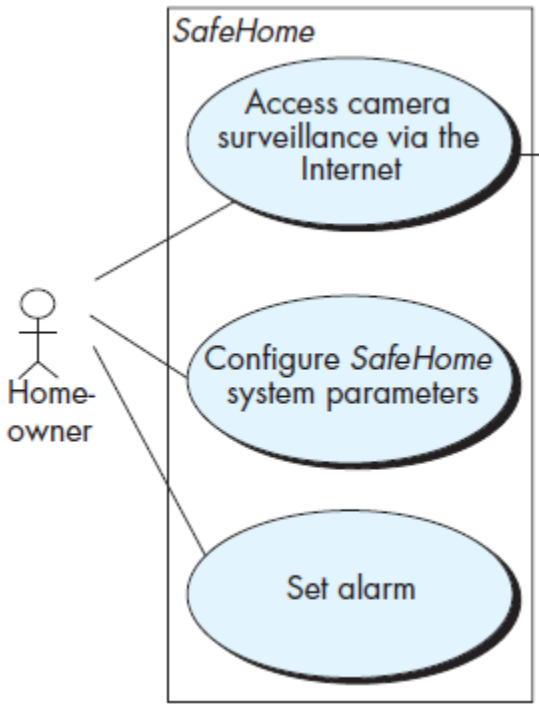
Actor اصلی: **Home Owner & Sys Admin**

Actor فرعی: **Camera & Sensors**



نمودار 2-8 صفحه 153

یادآوری : UML به ما کمک میکند تا نتیجه Communication را با یک ابزار نشان دهیم و Use Case Diagram به شکل بصری نشان داده شود. در UML ابزار Rational Rose ابزاریست برای کشیدن نمودارها.



- **Camera Surveillance via the Internet**
- **بررسی فعال و سالم بودن دوربین از طریق اینترنت**
- **Configure Safe Home System Properties**
- **پارامترها و شاخص های سیستم**

این نرم افزار Safe Home نام دارد.

Software Tools

Use Case Development

Objective: Assist in the development of use cases by providing automated templates and mechanisms for assessing clarity and consistency.

Mechanics: Tool mechanics vary. In general, use case tools provide fill-in-the-blank templates for creating effective use cases. Most use case functionality is embedded into a set of broader requirements engineering functions.

Representative Tools:[15]

The vast majority of UML-based analysis modeling tools provide both text and graphical support for use case development and modeling.

Objects by Design (**www.objectsbydesign.com/tools/umltools_byCompany.html**) provides comprehensive links to tools of this type.

برای سیستم در مجموع هشت مرحله ای فوق را در دست داریم و هر مرحله نیازمند کد نویسی است و هر کدام نیز نیازمند **Use Case & Activity Diagram** می باشد.

نکته : جزئیات فراموش نشود.

نکته : نمودارهای **UML** با یک دایره توپر مشکی شروع و با یک دایره خالی مشکی تمام میشود. شکل بیضی نشان دهنده فعل است.

در **Activity Diagram** که در صفحه **180** رسم شده است ابتدای کار **Log in** به سیستم مد نظر است . این یعنی اعتبار سنجی **(Authentication)** . کاربر تا **5** بار مجاز به وارد کردن اشتباه رمز می باشد. پس باید نرم افزاری برای اعتبار سنجی نام کاربری و رمز عبور باشد.

سناریو :

شروع : سیستم ما را شناسایی کرد. **(Valid)**

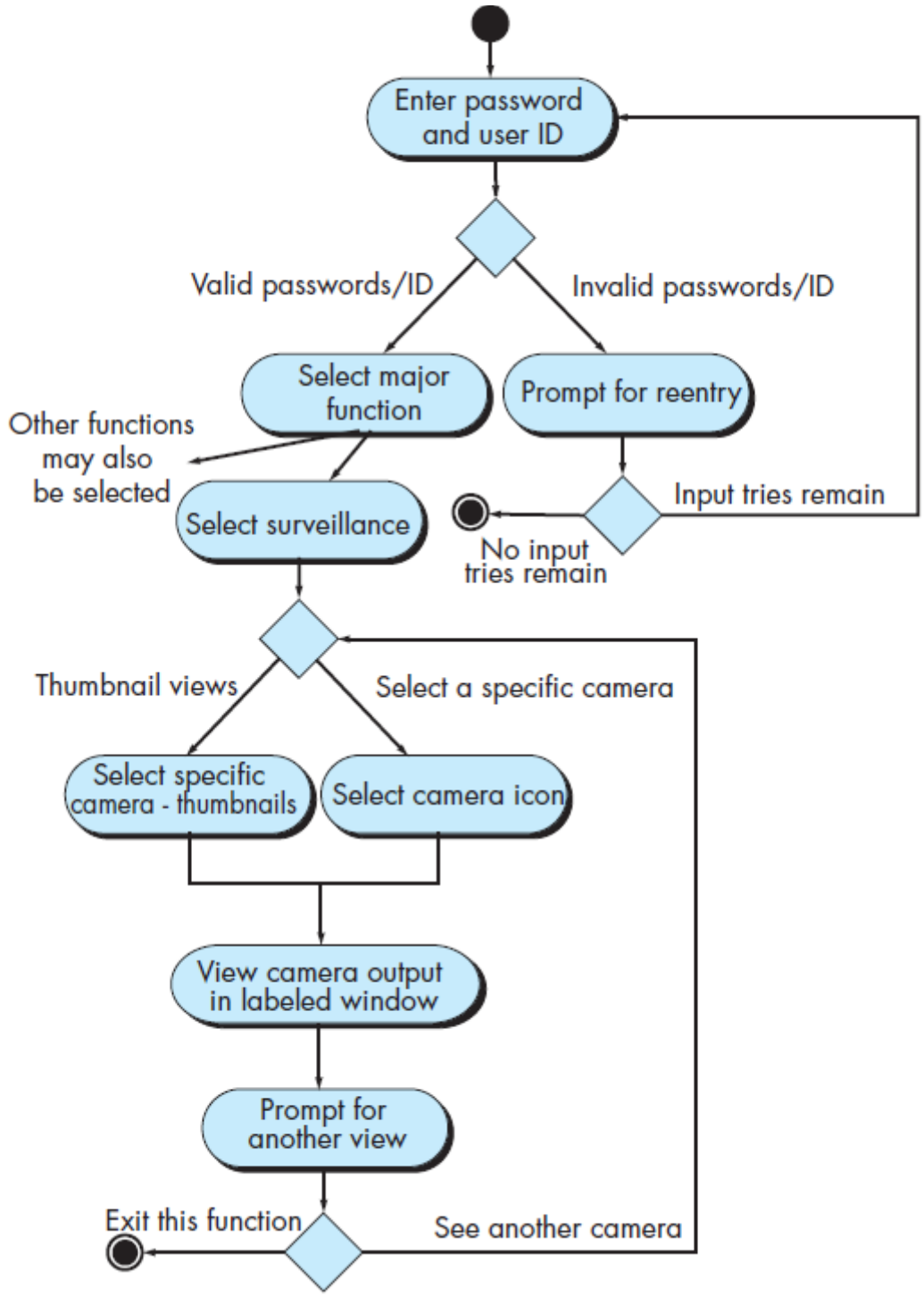
مرحله یک : **Select Major Function** (مثل وقتی که رمز عبور را در **ATM** صحیح وارد می کنیم و **ATM** ما را به صفحه عملیات بانکی رهنمون میکند.

مرحله دو : **Select Surveillance** (کلیک می کنیم تا صفحه خاص باز شود). این صفحه نشان می دهد آیا دوربین ها فعال هستند یا خیر ! (شمایی کوچک از تمام دوربین ها نمایش می یابد).

مرحله سوم : **Select Specific Camera-thumbnails** (انتخاب دوربین خاص با لمس آن)

مرحله چهارم : **View Camera Output in Labeled Window** (نمایش خروجی دوربین در صفحاتی که از پیش نام گذاری شده اند)

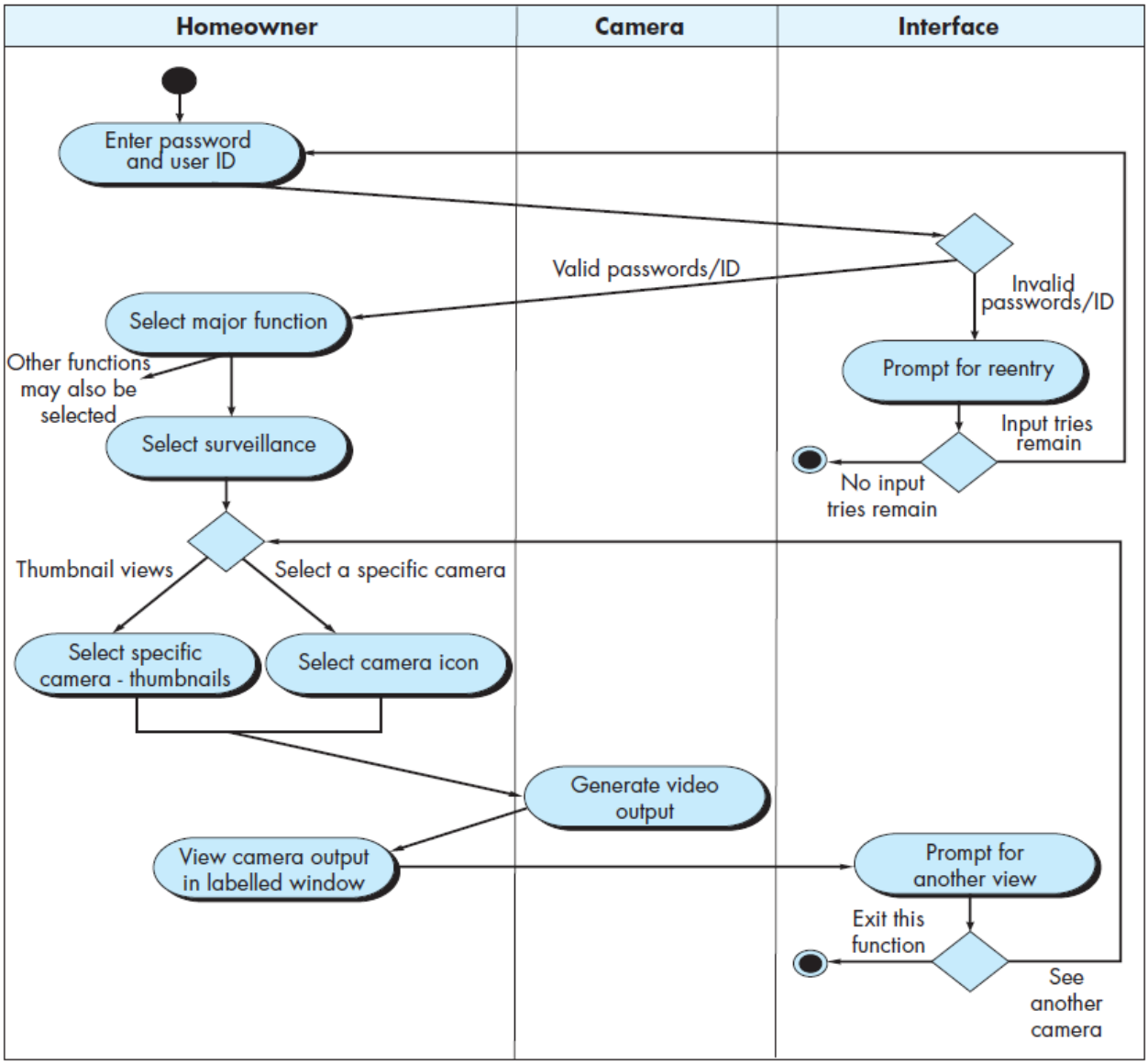
مرحله پنج : **Prompt for Another View** (مرحله های جدید و انتخاب دوربین های بعدی)



شکل 5 -9 صفحه 180

دخالت Actor ها در سیستم باعث تبدیل Activity Diagram به Swim Lane Diagram می شود. در Swim Lane Diagram نشان می دهیم در انجام کدام کار کدام Actor چه قسمتی از کار را انجام می دهد و تعامل بین Actor ها نمایش داده میشود.

FIGURE 9.6 Swimlane diagram for Access camera surveillance via the Internet—display camera views function.

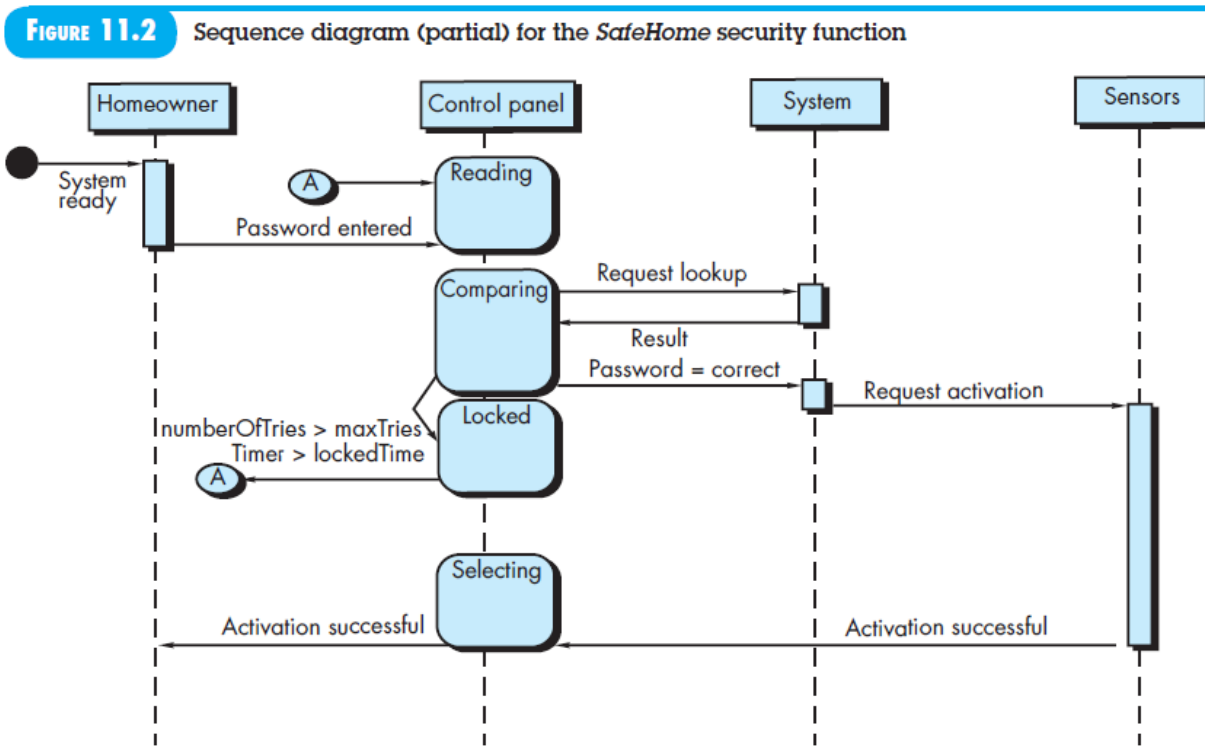


شکل 6 – 9 صفحه 181 کتاب

3 Activity Diagram وجود دارد :

Home Owner کاری را شروع کرده و نتیجه آنرا در Interface می بینیم. در صورت تایید به صفحه دیگری می رویم و در صورت اخطار چراغ قرمز ظاهر می شود. Swim Lane نشان می دهد که در انجام یک کار کدام یک از Actor ها نقش دارند. Swim Lane همان Activity Diagram می باشد با جزئیات بیشتر. در نمودار Swim Lane نقش زمان را نمی بینیم.

در این نمودار توالی کار و نقش زمان را مشاهده می کنیم و گردش کار کامل تر بیان می شود.



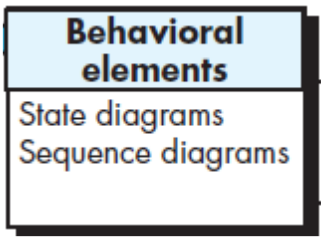
شکل 2-11 صفحه 206 کتاب

- ❖ مرحله یک : Home Owner
  - شروع کننده می باشد. آماده سازی سیستم (Sys Ready) مستلزم گذر زمان است.
- ❖ مرحله دو : Enter Password
- ❖ مرحله سه : Password Reading
- ❖ مرحله چهار : Comparing Password
  - Password Comparing to Find a Correct One (May Repeat Several)
- ❖ مرحله پنج : Locked (if Password Is Uncorrected)
- ❖ مرحله شش : Request Activation
  - بعد از ورود به سیستم عملیات هایی بایستی صورت گیرد
    - تنظیم سنسورها
      - سنسورهای حرارتی ضربه ای و ...

قانون : شروع کننده کار در واقع خاتمه دهنده کار است. Sequence Diagram

❖ **Behavioral Elements** :

- **State Diagram**
- **Sequence Diagram**



این گروه نمودار رفتار سیستم را نشان می دهند.

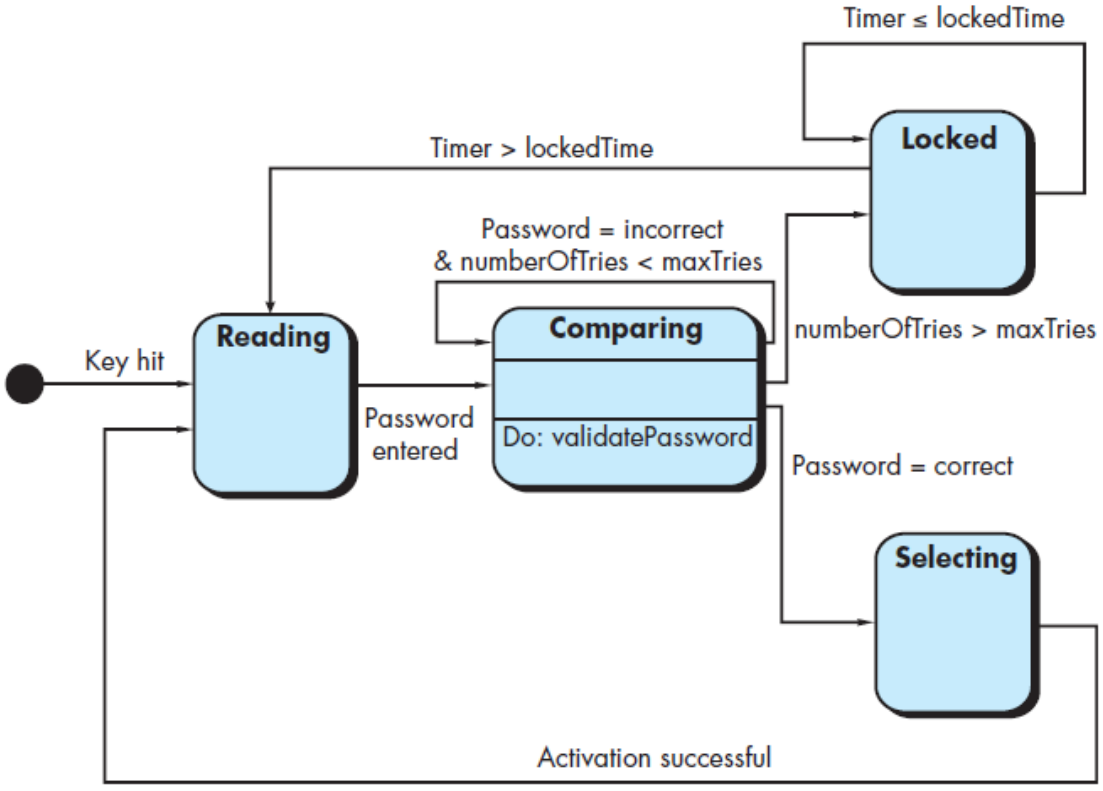
❖ **Flow – Oriented Elements**

این نوع نمودار رابطه سیسم با محیط عمیاتی را نمایش می دهد.

نام دیگر: DFD (Data Flow Diagram)

نکته : سیستم را با خط بدون گوشه یا بیضی و محیط را با خط گوشه دار یا مستطیل نشان می دهند.

FIGURE 11.1
State diagram for the ControlPanel class



نمودار State Diagram برای کلاس Control Panel صفحه 205

SOFTWARE TOOLS

Generalized Analysis Modeling in UML

Objective: Analysis modeling tools provide the capability to develop scenario-based models, class-based models, and behavioral models using UML notation.

Mechanics: Tools in this category support the full range of UML diagrams required to build an analysis model (these tools also support design modeling). In addition to diagramming, tools in this category (1) perform consistency and correctness checks for all UML diagrams, (2) provide links for design and code generation, (3) build a database that enables the management and assessment of large UML models required for complex systems.

Representative Tools:[3]
The following tools support a full range of UML diagrams required for analysis modeling:

ArgoUML is an open source tool available at **argouml.tigris.org**.

Enterprise Architect, developed by Sparx Systems (**www.sparxsystems.com.au**).

PowerDesigner, developed by Sybase (**www.sybase.com**).

Rational Rose, developed by IBM (Rational) (**http://www-01.ibm.com/software/rational/**).

Rational System Architect, developed by Popkin Software now owned by IBM (**http://www-01.ibm.com/software/awdtools/systemarchitect/**).

UML Studio, developed by Pragsoft Corporation (**www.pragsoft.com**).

Visio, developed by Microsoft (**http://office.microsoft.com/en-gb/visio/**).

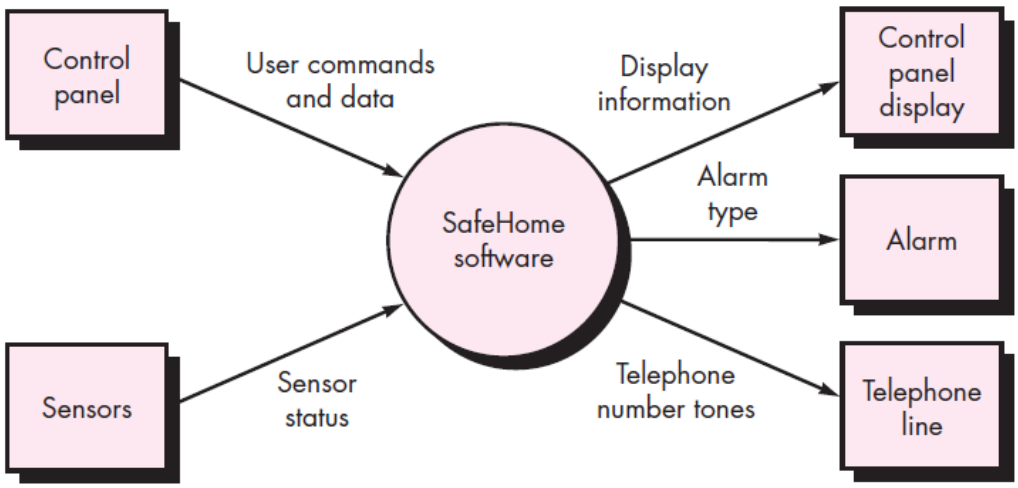
Visual UML, developed by Visual Object Modelers (**www.visualuml.com**).

Software Tools صفحه 207 کتاب

سطوح DFD

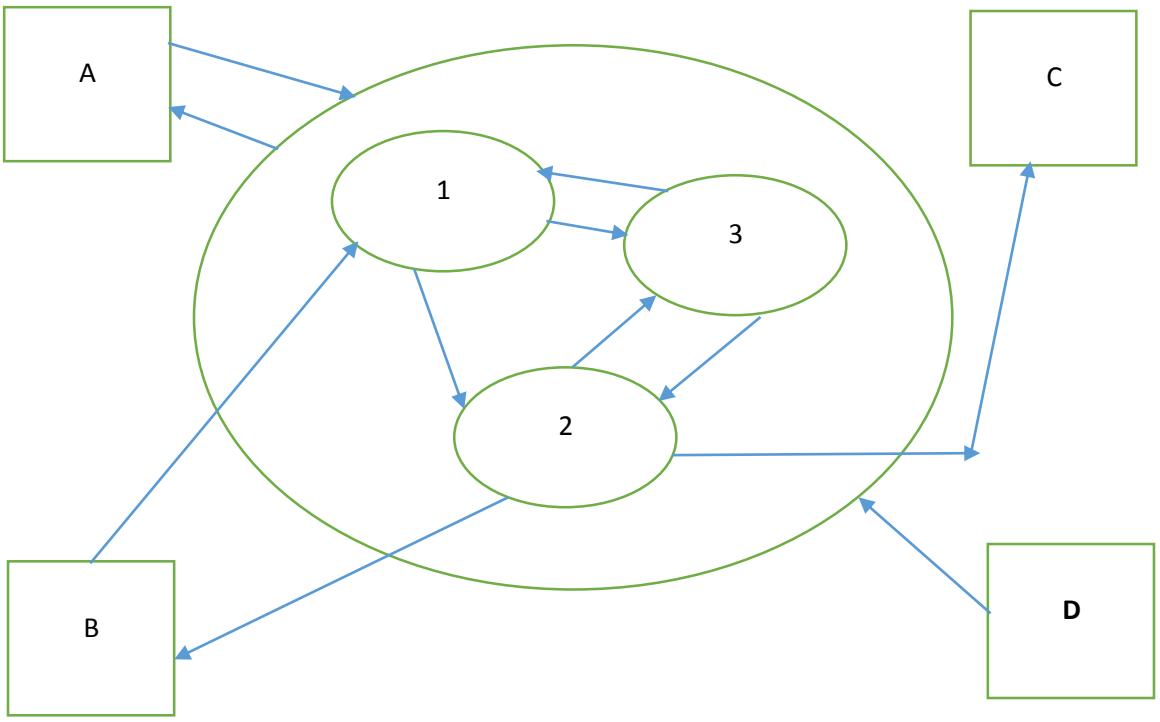
سطح 0 : رابطه کل سیستم با محیط

FIGURE 7.1
Context-level DFD for the *SafeHome* security function



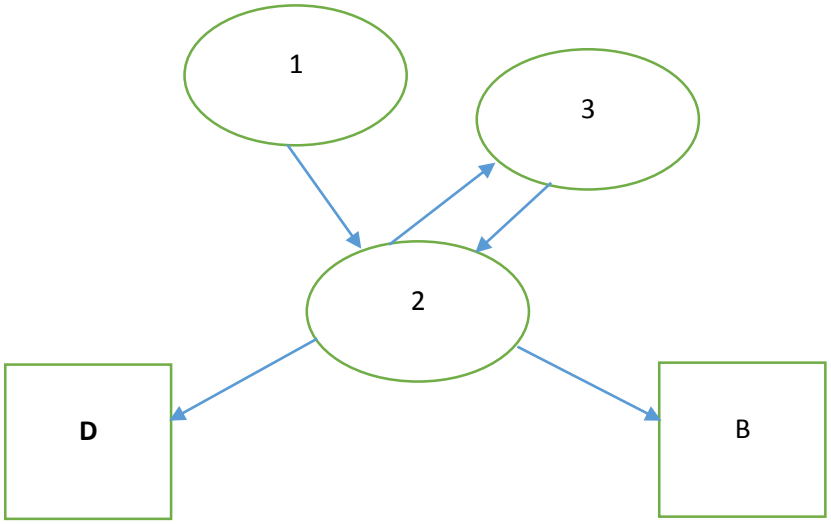
شکل 1- 7 صفحه 188 ویراست 7

سطح 1 : رابطه زیر سیستم ها و محیط



سه زیر سیستم داریم در نتیجه سه DFD سطح یک داریم.

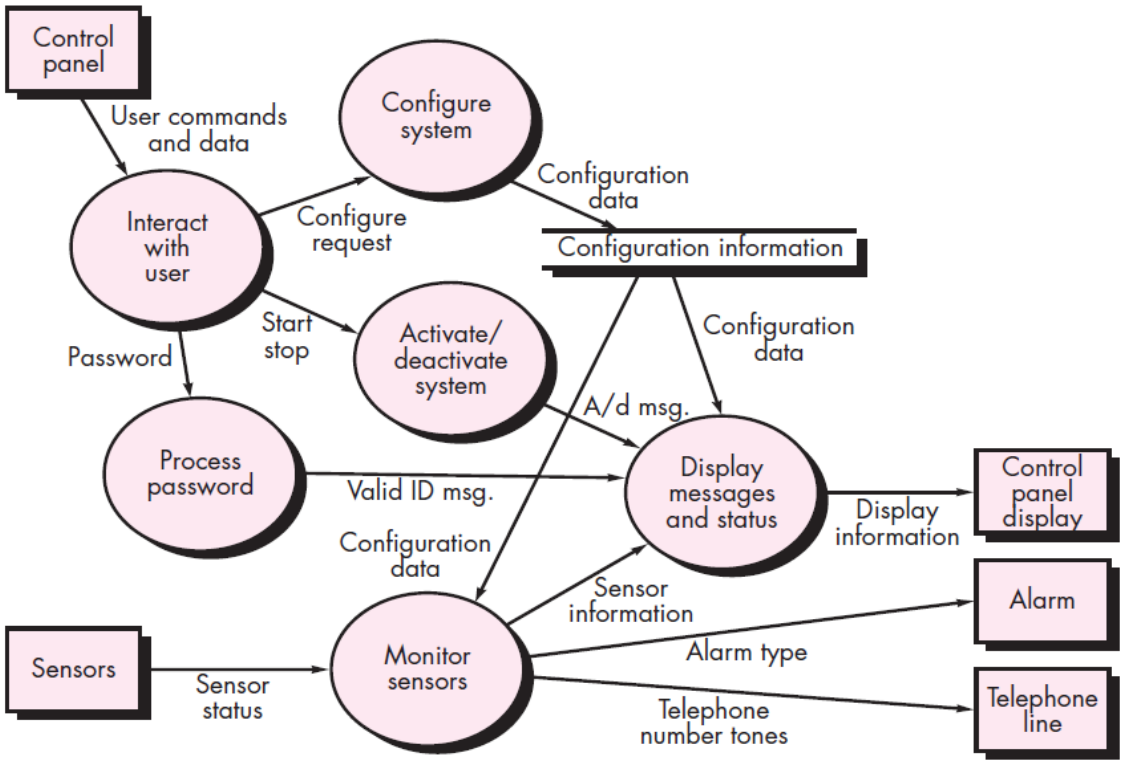
سطح 2 : اگر هر زیر سیستم یک DFD مثل سطح 0 داشته باشد نتیجه DFD سطح 2 خواهد شد. یعنی زیر سیستم 2 یک DFD 0 میخواهد.



آنالیز سیستم :

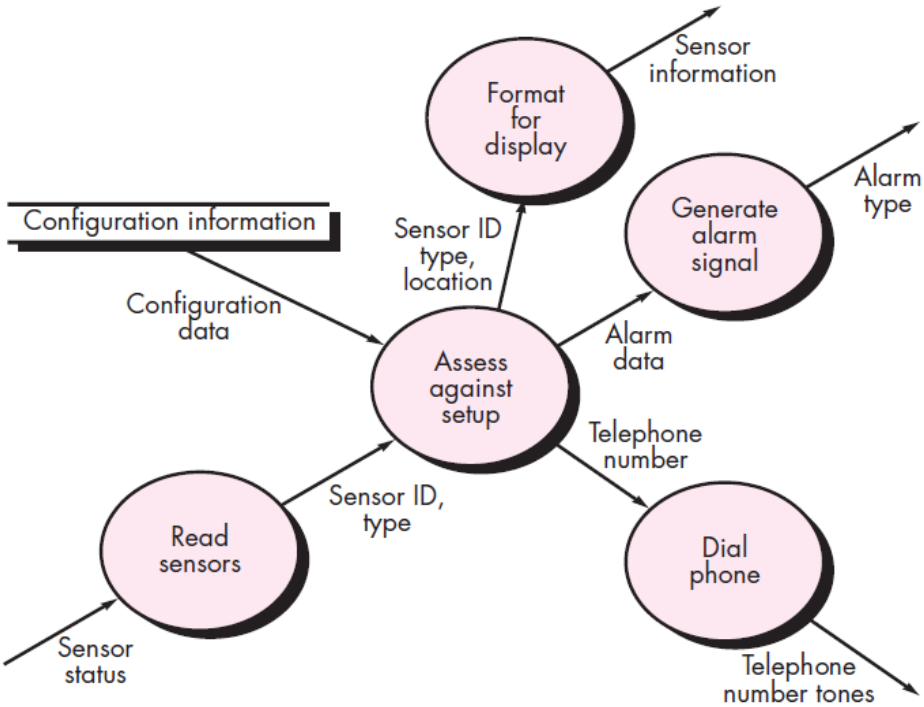
DFD 1 & 2 دارای Subsystem 6 هستند.

FIGURE 7.2
Level 1 DFD for *SafeHome* security function



شکل 2-7 صفحه 190 ویراست 7 کتاب

FIGURE 7.3
Level 2 DFD that refines the *monitor sensors* process



شکل 3-7 صفحه 190 ویراست 7 کتاب

Personal Software Process – PSP

Team Software Process – TSP

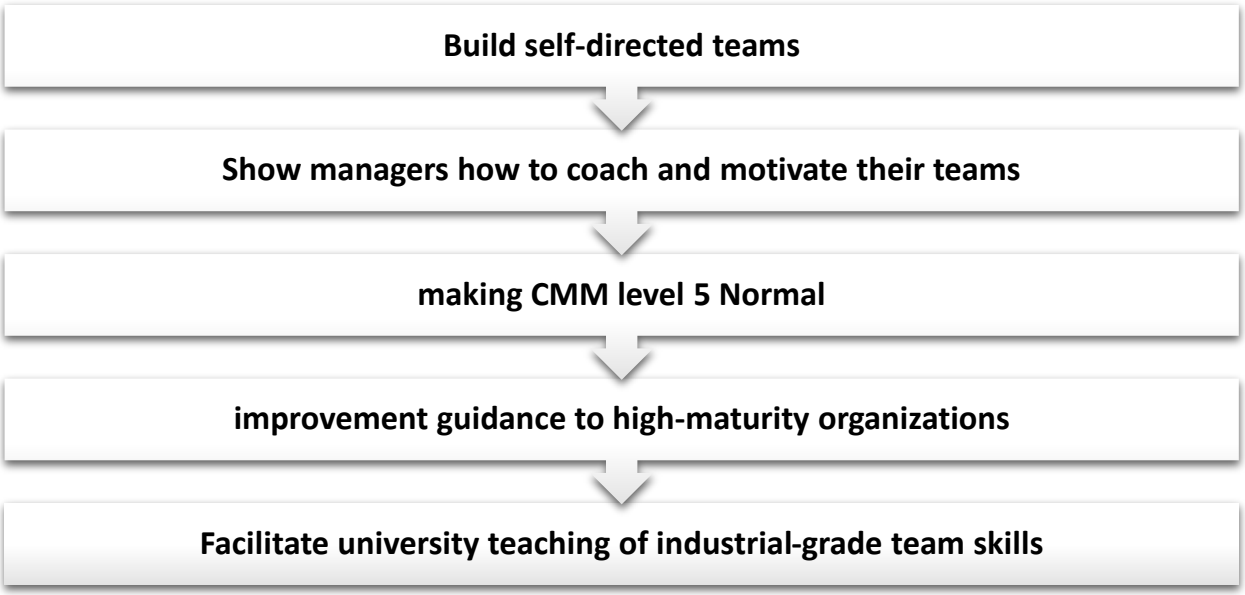
- **PSP** :

معیارهای اندازه گیری نرم افزارهای تولید شده را در دستور کار خود قرار می دهیم در عین حال از کیفیت هم غافل نخواهیم شد.

برای رسیدن به دو اصل فوق 5 گام را پیش رو داریم :

- **1**: افرادی که اهل برنامه ریزی باشند.
- **2**: دانش **UML**
- **3**: کار امروز افراد باید بهتر از دیروز باشند.(در **Progress** باشند).
- **4**: شناسایی خوب کلاس ها و متد ها (شناخت تفاوت تولید نرم افزار با تولید **Log** های آن).

- **TSP**:



در TSP به اهداف ذیل توجه می کنیم :

- **Self – Directed : ساخت تیم هایی با قابلیت خود رهبری**
  - **پیگیر کارها باشند**
  - **برنامه ریزی کنند.**
  - **اهداف را مشخص کنند.**
  - **در اهداف کاری نقش موثری داشته باشند.**
- **Show managers how to coach and motivate their team :**
  - **آموزش تشویق تیم توسط مدیران به منظور ابقا تیم در اوج**
  - **چه روشی ؟ روش های کلاسیک**
- **Facilitate university teaching of industrial-grade team skills**
  - **امکان آموزش به اعضای تیم در دانشگاه به منظور ارتقا مهارت صنعتی کاربردی**

تهیه نمودارها

- **اصول هادی تجربیات (Principle that Guide Process)**

	Principle	**Description**
1	Be Agile	**چست و چابک باش (پویا – حلزونی)**
2	Focus on Quality at Every Step	**توجه مداوم به کیفیت**
3	Ready to Adapt	**توجه مداوم به سازگاری**
4	Build an Effective Team	**یک تیم موثر باشیم (ریز بین)**
5	Manage Changes	**مدیریت تغییرات**
6	Assess Risks	**ارزیابی خطرات (CMMI)**
7	Establish mechanisms for communication and coordination	**مکانیزم ارتباطات و هماهنگی**
8	Create work products that provide value for others	**خلق محصولات با ارزش**

- Principles That Guide Practice
- Principle 1. Divide and conquer
- Principle 2. Understand the use of abstraction.
- Principle 3. Strive for consistency
- Principle 4. Focus on the transfer of information
- Principle 5. Build software that exhibits effective modularity
- Principle 6. Look for patterns.
- Principle 7. When possible, represent the problem and its solution from
- a number of different perspectives
- Principle 8. Remember that someone will maintain the software

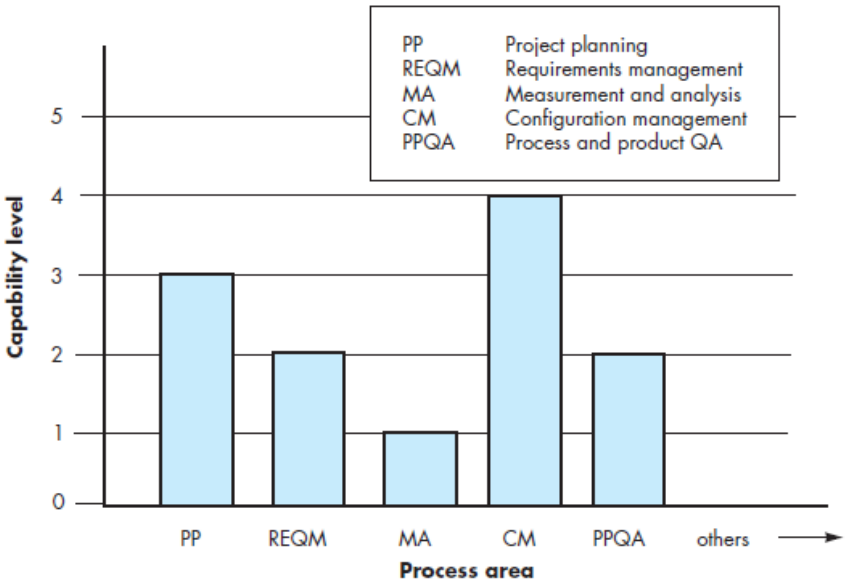
نکته : به تفاوت اصول Principle that Guide Process و Principle that Guide Practice

توجه کنید.(اینکه هر کدام دارای 8 مورد مجزا هستند دچار اشتباه نشود).

FIGURE 37.2

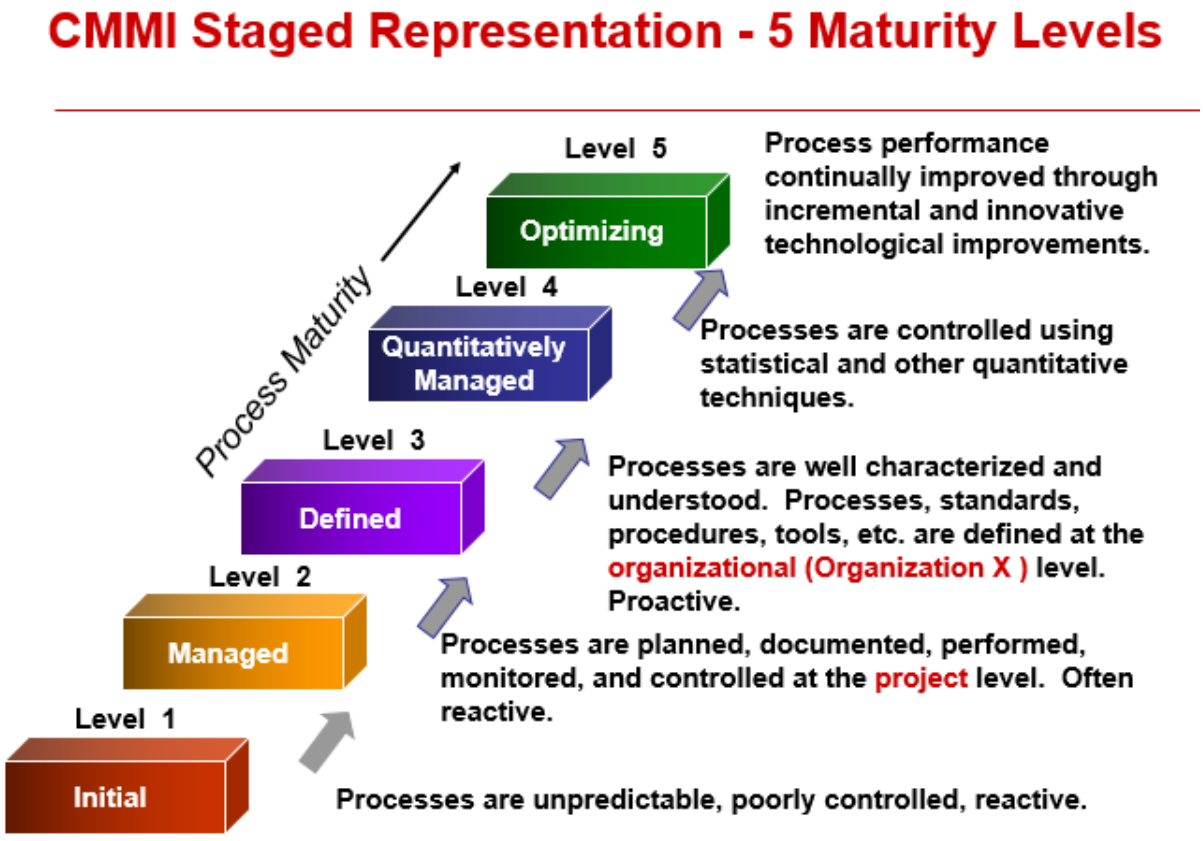
CMMI Process Area Capability Profile

Source: [Phi02].



شکل 2-37 صفحه 829 کتاب

شکل شمایی از مدل CMMI می باشد.



- **اصول مهم در CMMI**
  - **روحیه رقابتی**
  - **مدیریت خطرات**
    - **خطرات انتخاب اشتباه نیرو**
    - **خطرات انتخاب اشتباه ابزار**
    - **خطرات هزینه های بی جا**
    - **و...**

ارزشیابی خطرات یعنی بدانیم خطرات به چه اندازه هزینه بر است.

- **اصول اصلی در Process**
- **A**

- **Communication Principles :**
- Principle 1. Listen
- Principle 2. Prepare before you communicate
- Principle 3. Someone should facilitate the activity
- Principle 4. Face-to-face communication is best.
- Principle 5. Take notes and document decisions
- Principle 6. Strive for collaboration
- Principle 7. Stay focused; modularize your discussion
- Principle 8. If something is unclear, draw a picture
- Principle 9.
  - (a) Once you agree to something, move on.
  - (b) If you can't Agree to something, move on
  - (c) If a feature or function is unclear and cannot be clarified at the moment, move on
- Principle 10. Negotiation is not a contest or a game. It works best when both parties win.

اصل یک : گوش کنید (یک زبان داده اند و دو گوش)

اصل دو : پیش از گفت و گو با مشتری ابزار و ذهن تان را آماده کنید.

اصل چهار : حضوری صحبت کنید. (ملاقات های Face to Face بهترین شیوه برقراری ارتباط است)

اصل هشت : استفاده از اشکال و متدهای ویژوال (در صورتی که موضوع شفافیت کافی نداشت)

اصل ده : گفت و گو یک بازی با امکان برد و باخت عامیانه نیست. در Team Work مهمترین اصل Interactivity است.

- اصول اصلی در **Process**
- **B**

- **Planning Principles**
- Principle 1. Understand the scope of the project
- Principle 2. Involve stakeholders in the planning activity
- Principle 3. Recognize that planning is iterative
- Principle 4. Estimate based on what you know
- Principle 5. Consider risk as you define the plan.
- Principle 6. Be realistic
- Principle 7. Adjust granularity as you define the plan
- Principle 8. Define how you intend to ensure quality
- Principle 9. Describe how you intend to accommodate change
- Principle 10. Track the plan frequently and make adjustments as required.

اصل یک : درک باید و نباید های سیستمی

اصل دو : سهامداران را در امر جریان برنامه ریزی مطلع و وارد کن

اصل سه : بدان که برنامه ریزی امری است که باید مکررا صورت گیرد.

اصل چهار : برآورد ها بر اساس تجارب شخصی است نه متون کتاب یا افراد پروژه های خارجی

اصل پنج : ریسک های پروژه را مد نظر داشته باش

اصل شش : واقع بین باش (حقیقت گرایی در نفی ایده آلیست فنی)

اصل هفت : کنترل سر راست بودن و روان بودن پروژه (چون چشمه روان باش)

اصل هشت : تعریف و شفاف سازی چگونگی تامین کیفیت پروژه

اصل نه : تشریح تطبیق سازی تغییرات در امر پروژه

اصل ده : برنامه را به طور مداوم پیگیری کن و تغییرات لازم را مبذول کن.

- **C : Modeling**

- **Modeling Principles**

  Principle 1. The primary goal of the software team is to build software, not Create models

- Principle 2. T ravel light—don’t create more models than you need
- Principle 3. Strive to produce the simplest model that will describe the Problem or the software
- Principle 4. Build models in a way that makes them amenable to change
- Principle 5. Be able to state an explicit purpose for each model that is created
- Principle 6. Adapt the models you develop to the system at hand
- Principle 7. Try to build useful models, but forget about building perfect Models
- Principle 8. Don’t become dogmatic about the syntax of the model. If it Communicates content successfully, representation is secondary
- Principle 9. I f your instincts tell you a model isn’t right even though it Seems okay on paper, you probably have reason to be concerned
- Principle 10. Get feedback as soon as you can

اصل یک : هدف ما تولید نرم افزار است نه ساخت مدل

اصل دو : ابزار . مدل و نیروی انسانی و ... (ما زاد و اضافی ها) را رها می کنیم.(به حد کفایت)

اصل سه : تلاش در جهت تولید مدلی ساده به منظور تشریح مساله یا نرم افزار

اصل چهار : ساخت مدل با امکان آمادگی برای تغییرات

اصل پنج : قابلیت ارایه ادله کافی برای مدل ها

اصل شش : تطبیق مدل های توسعه سیستم

اصل هفت : مدل های کارآ بساز در جایی که امکان ساخت مدل های کامل مقدور نیست.

اصل هشت : تعصب ها را در رابطه با Syntax یا یک مدل خاص کنار بگذار . کارآیی موفقیت آمیز مدل در مرحله اول اولویت دارد و شکل و شمایل ارایه در مرحله دوم.

اصل نه : اگر قلبت با تو در رابطه با یک مدل خاص همراه نبود با وجود قابل قبول بودن روی کاغذ وقت آنست که به فکر چاره باشی.

اصل ده : بازخورد ها را به سرعت دریافت کن.

- **D : Analysis**

- **Analysis Principles**
- Principle 1. The **information domain** of a problem must be represented and understood.
- Principle 2. The functions that the software performs **must be deified**
- Principle 3. The **behavior of the software** (as a consequence of external Events) must be represented
- Principle 4. The models that depict information, function, and behavior must **be partitioned in a manner that uncovers detail in a layered** (or hierarchical) Fashion
- Principle 5. **The analysis task** should move from essential information toward Implementation detail

اصل یک : دامنه اطلاعاتی مساله باید به نحو احسن نمایش داده شده و درج شود.

اصل دو : عملیات کاری نرم افزار باید به نحو احسن درج شود.

اصل سه : رفتار نرم افزار باید به نحو احسن نمایش داده شود.

اصل چهار : مدل ارایه دهنده اطلاعات و توابع می بایست به شکلی تقسیم شود که جزییات را در لایه های مختلف نشان دهد.

اصل پنج : تحلیل باید بتواند ما را از اطلاعات اصلی به جزییات پیاده سازی نزدیک کند.

- **E : Design**

- **Design Modeling Principles**
- 1 : Design should be traceable to the requirements model
- 2 : Always consider the architecture of the system to be built
- 3: Design of data is as important as design of processing functions.
- 4:Interfaces (both internal and external) must be designed with Care
- 5: User interface design should be tuned to the needs of the end user. However, in every case it should stress ease of Use.
- 6:Component-level design should be functionally independent
- 7: Components should be loosely coupled to one another and to the external environment
- 8: Design representations (models) should be easily understandable
- 9: The design should be developed iteratively
- 10: Creation of a design model does not preclude an agile approach

اصل یک : طراحی باید قابلیت پیگیری و کنترل سیستم بر اساس نیازها را داشته باشد.
اصل دو : در طراحی نمودارها در در قالب معماری نرم افزاری پیش برویم.
اصل سه : طراحی داده مهم هست . حتی شاید بیشتر از طراحی موارد پروژه
اصل چهار : Interface های ورودی و خروجی بایستی با دقت طراحی شوند.
اصل پنج : UI بر اساس User Requirements رخ می دهد (با محوریت معماری نرم افزاری)
اصل شش : ترمینال ها بعد از طراحی و تایید نمودارها توسط اعضاء تیم صورت می گیرد.
نکته : سطوح برگها باید مستقل باشند(از حیث کاری)
اصل هفت : اتصال کامپوننت ها به یکدیگر و محیط خارجی باید پایین باشد.
اصل هشت : مدل طراحی شده می بایست به سادگی قابل فهم باشد.
اصل نه : طراحی می بایست به طور مکرر توسعه یابد.
اصل ده : خلق یک مدل طراحی مانع روشی چست و چابک (Agile) نخواهد شد.

- **Construction Principles**

موضوع Coding Principles

Preparation Principles: Before you write one line of code, be sure you

• Understand of the problem you're trying to solve.

• Understand basic design principles and concepts.

• Pick a programming language that meets the needs of the software to be Built and the environment in which it will operate.

• Select a programming environment that provides tools that will make Your work easier.

• Create a set of unit tests that will be applied once the component you code Is completed.

قواعد لازم پیش از کد زنی : [Preparation Principles]

- **درک مساله موردنظر**
- **انتخاب زبان مناسب**
- **انتخاب محیط کدزنی**
- **ساخت آزمون های تست**

قواعد لازم هنگام برنامه نویسی : [Programming Principles]

- **برنامه ها را حدالمقدور بدون Go To می نویسیم (رعایت اصل ساخت یافتگی)**
- **استفاده وافی از Pair Programming**
- **دستورات شرطی ساده**
- **ایجاد حلقه های تکرار با امکان تست آسان**
- **ساختمان داده مناسب**

قواعد لازم الاجرا پس از برنامه نویسی [Validation Principles]

- **برنامه قابلیت Trace در هر زمان ممکن را داشته باشد.**
- **برنامه قابلیت بهره مندی از روش Refactoring را داشته باشد.**
- **برنامه قابلیت اعمال Unit Test و Debug داشته باشد.**

- موضوع **Testing Principles**

Coding Principles: As you begin writing code, be sure you

• Constrain your algorithms by following structured programming [Boh00] Practice.

• Consider the use of pair programming.

• Select data structures that will meet the needs of the design.

• Understand the software architecture and create interfaces that are Consistent with it.

• Keep conditional logic as simple as possible.

• Create nested loops in a way that makes them easily testable.

• Select meaningful variable names and follow other local coding standards.

• Write code that is self-documenting.

• Create a visual layout (e.g., indentation and blank lines) that aids Understanding.

- **تعریف آزمون ها توسط ما قابلیت پیگیری نیازمندی ها مشتری را داشته باشد.**
- **پیدایش آزمون ها پیش از تعریف در مرحله System Test**
- **عدم مثمر ثمر بودن آزمون های زیاد و طولانی**

نکته : Testing پردازشی است که در صورت شروع به کار برنامه خطاهای سیستم را متوجه می شویم.

نکته : Good Test نوعی آزمون با قابلیت کشف خطاهای شناسایی نشده در سیستم با احتمال بالا است.

نکته : Successful Test : نوعی آزمون با قابلیت کشف خطاهای ناشناخته تابحال در سیستم است.

- **G : Deployment**
- **الزام مدیریت نیازمندی های مشتری**
- **قابلیت حمل سیستم به شرط تست کامل سیستم و آمادگی برای پیاده سازی در محیط**
- **تعریف روش های پشتیبانی نرم افزار پیش از حمل و نصب نرم افزار**

چنین گفت پیغمبر راست گوی زگهواره تا گور دانش بجوی