

# فرآیند

درس مهندسی نرم افزار ۲

# مفاهیم کلیدی

- فرآیند نرم افزار
- نگاه لایه‌ای به مهندسی نرم افزار
- مراحل عمومی مهندسی نرم افزار
- فرآیند توسعه نرم افزار
- مدل‌های فرآیند نرم افزار
- رابطه محصول و فرآیند

# فرآیند نرم افزار

## ■ تعریف اولیه

- **چارچوبی برای انجام وظایف** که برای دستیابی به نرم افزاری با کیفیت بالا مورد استفاده قرار می گیرد

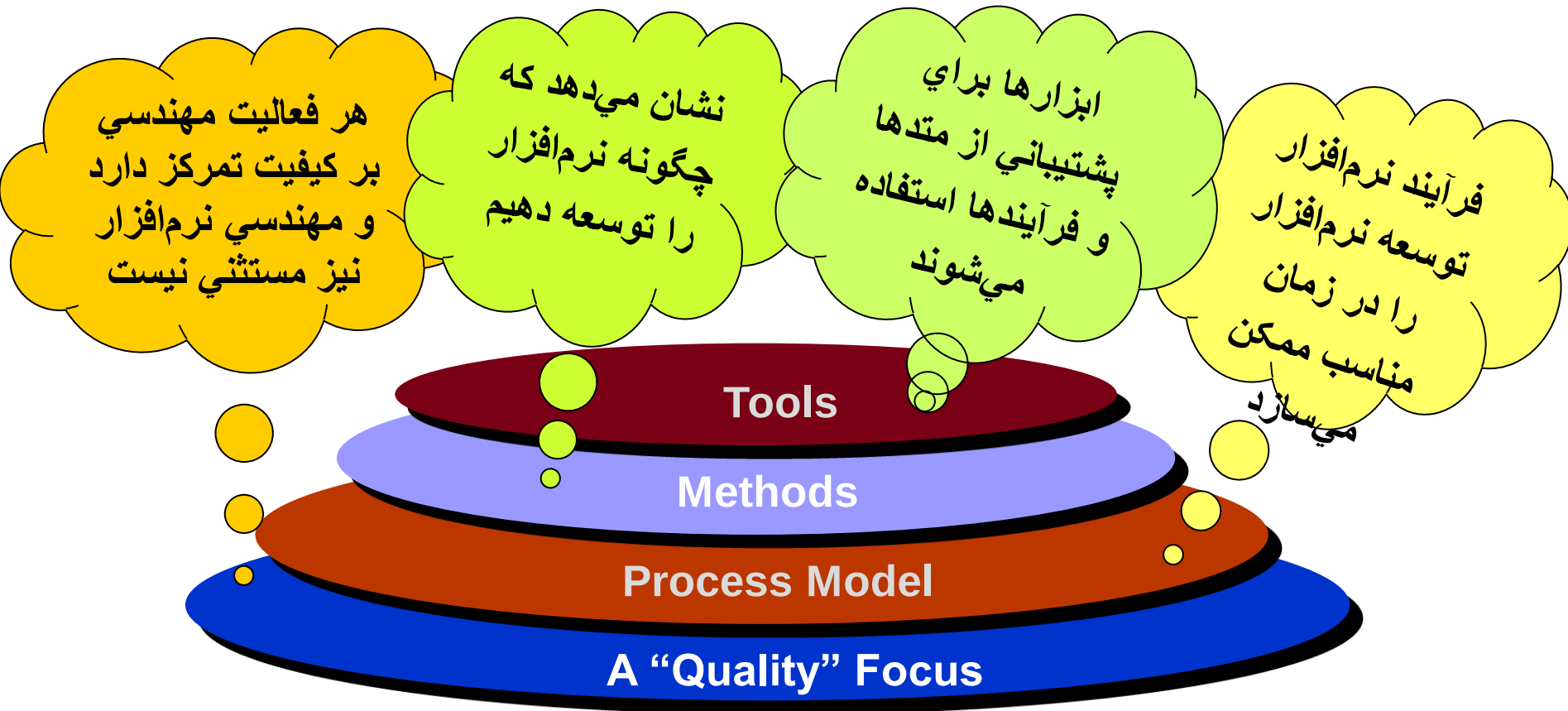
## ■ فرآیند نرم افزار مترادف مهندسی نرم افزار

- **بله:** فرآیند روشی را که نرم افزار **مهندسی می شود**، تعیین می کند
- **خیر:** مهندسی نرم افزار شامل **مجموعه ای از فناوری هاست** (متدها و ابزارها) که در فرآیند مورد استفاده قرار می گیرند

مهندسي نرم افزار توسط افراد خلاق و مطلع که در یک فرآیند تعریف شده  
و بالغ که برای تولید محصولی که تقاضا شده و یا بازار نیاز دارد، کار  
می کنند، انجام می شود

# نگاه لایه‌ای به مهندسی نرم‌افزار

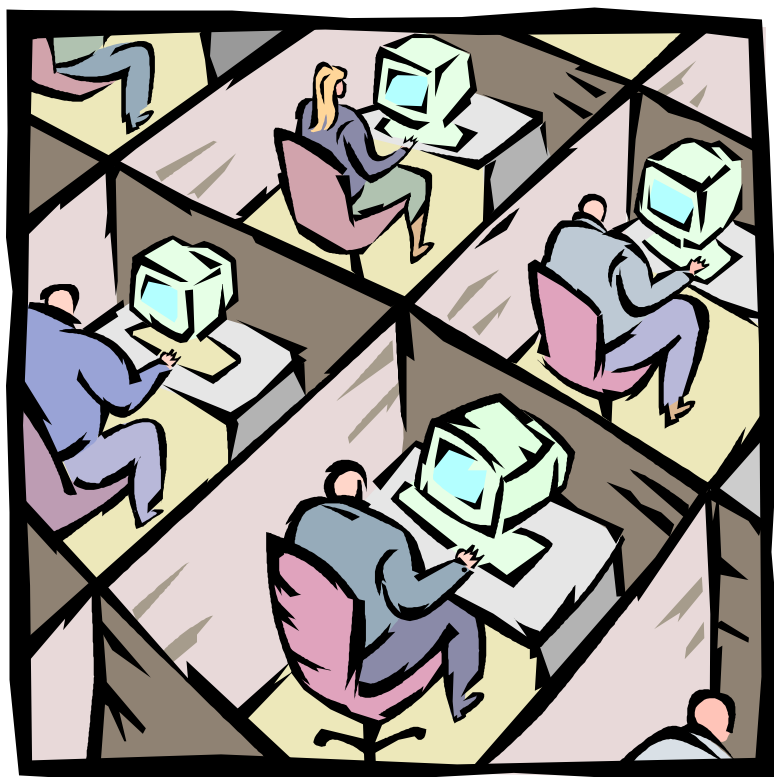
■ مهندسی نرم‌افزار از چهار لایه تشکیل شده است



# نگاه لایه‌ای به مهندسی نرم‌افزار (ادامه)

■ کیفیت از افراد تیم شروع می‌شود

- آگاهی
- بهترین تجربیات
- تعهد
- نگاه بالا به پایین (تحت امر بودن)
- نگاه پایین به بالا (توجه به تیم)



# نگاه لایه‌ای به مهندسی نرم‌افزار (ادامه)

- هر فرآیند چارچوبی برای مجموعه‌ای از نواحی کلیدی فرآیند (*Key Process Areas*) تعریف می‌کند
- نواحی کلیدی فرآیند،
  - ارائه تعاریف پایه‌ای برای مدیریت پروژه‌های نرم‌افزاری
  - ایجاد زمینه‌ای برای
    - اجرای متد و تولید محصولات
    - دستیابی به کیفیت
    - مدیریت تغییرات

# مراحل عمومی مهندسی نرم افزار

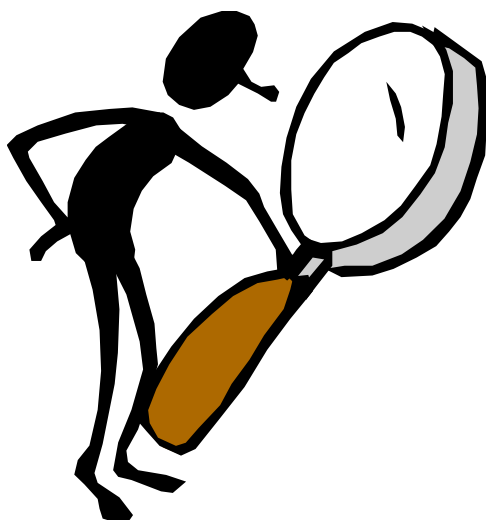
## ■ مرحله تعریف (*Definition Phase*)

### • تشخیص اینکه

- چه اطلاعاتی باید پردازش شود؟
- چه وظایفی باید انجام شود؟
- چه اجزاهایی وجود دارد؟
- ...

### • سه کار اساسی

- مهندسی اطلاعات یا سیستم
- برنامه ریزی پروژه
- تحلیل نیازمندی ها





# مراحل عمومی مهندسی نرم افزار (ادامه)

## ■ مرحله توسعه (*Development Phase*)

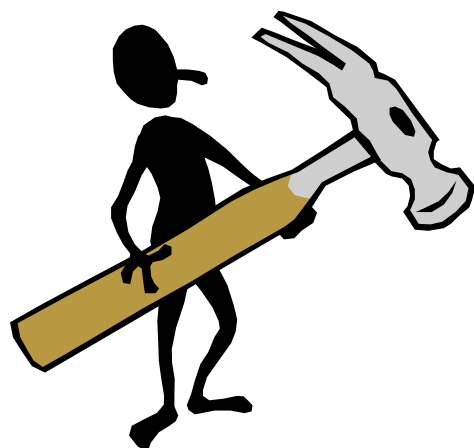
### • تعیین اینکه

- ساختار داده‌ها چگونه باشد؟
- وظایف چگونه پیاده‌سازی شوند؟
- طراحی رابط چگونه باشد؟

□ ....

### • سه کار اساسی

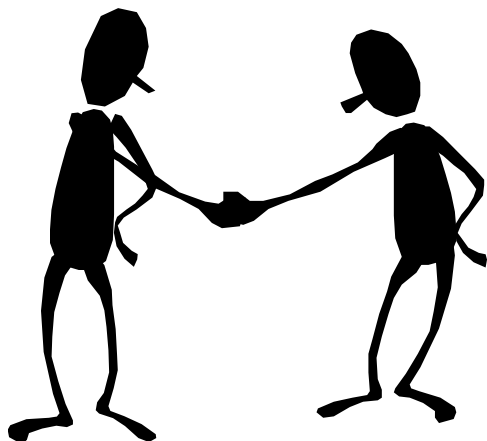
- طراحی نرم افزار
- تولید کد
- آزمایش نرم افزار



# مراحل عمومی مهندسی نرم افزار (ادامه)

## ■ مرحله حمایت (*Support Phase*)

- تغییر نرم افزار
  - همگام با تکامل محیط
  - با تغییر نیازمندی ها
- انجام وظایف دو مرحله قبل بر روی نرم افزار موجود
- کارهای حمایتی
  - آموزش نرم افزار
  - رفع مسائل و مشکلات کاربران



چهار نوع تغییر در هر نرم افزار بوجود می آید که عبارتند از اصلاح  
(*Correction*)، تطبیق (*Adaptation*)،  
ارتقاء (*Enhancement*) و پیشگیری (*Prevention*)

# فعالیت‌های حمایتی (*Umbrella Activities*)

- فعالیت‌های عمومی با فعالیت‌های حمایتی زیر کامل می‌شوند
  - مدیریت پروژه‌های نرم‌افزاری (*Software Project Management*)
  - واریسی‌های فنی (*Formal Technical Reviews*)
  - تضمین کیفیت نرم‌افزار (*Software Quality Assurance*)
  - مدیریت پیکربندی نرم‌افزار (*Software Configuration Management*)
  - آماده‌سازی و تولید محصولات کاری (*Work product preparation and production*)
  - مدیریت قابلیت استفاده مجدد (*Reusability Management*)
  - سنجش (*Measurement*)
  - مدیریت خطر (*Risk management*)

# فرآیند نرم افزار

- هر فرآیند نرم افزار می تواند به صورت مجموعه ای از فعالیت های چارچوب مشترک فرآیند و فعالیت های حمایتی دیده شود

# فرآیند نرم افزار (ادامه)

■ فعالیت‌های چارچوب می‌توانند بر هر پروژه‌ای اعمال شوند و  
تطبیق داده شوند، اما ...

■ تطبیق فعالیت‌های چارچوب بستگی دارد به:

- نوع پروژه

- خصوصیات پروژه

- نیازمندی‌های تیم پروژه



# فرآیند نرم افزار – مدل بلوغ قابلیت

■ اهمیت بلوغ فرآیند نرم افزاري

■ مدل بلوغ قابلیت (*Capability Maturity Model*)

- ارائه شده توسط Software Engineering Institute
- نشان دهنده کارایی فرآیند مهندسي نرم افزار سازمان
- معياري براي ارزيابي ميزان بلوغ فرآیند مدیریت و توسعه نرم افزار و محصولات در سازمان
- شامل مجموعه‌اي از فعاليت‌هاي کلیدی در پنج سطح مختلف

# فرآیند نرم افزار – مدل بلوغ قابلیت (ادامه)

## ■ سطوح مدل بلوغ قابلیت

### • سطح ۱ – اولیه (*Initial*)

- پروژه‌های نرم افزار از فرآیند خاصی تبعیت نمی‌کنند
- موفقیت به تلاش افراد بستگی دارد

### • سطح ۲ – تکرارپذیر (*Repeatable*)

- فرآیندهای مدیریت پروژه برای کنترل هزینه، زمان و عملکرد برقرار شده است
- نظم لازم برای استفاده از تجربیان پروژه‌های موفق قبلی برقرار شده است

### • سطح ۳ – تکرارپذیر (*Defined*)

- فرآیند توسعه نرم افزار استاندارد (متدولوژی) خریداری/توسعه داده شده است
- در تمام پروژه‌ها از این فرآیند توسعه استفاده می‌شود



# فرآیند نرم افزار – مدل بلوغ قابلیت (ادامه)

## ■ سطوح مدل بلوغ قابلیت (ادامه)

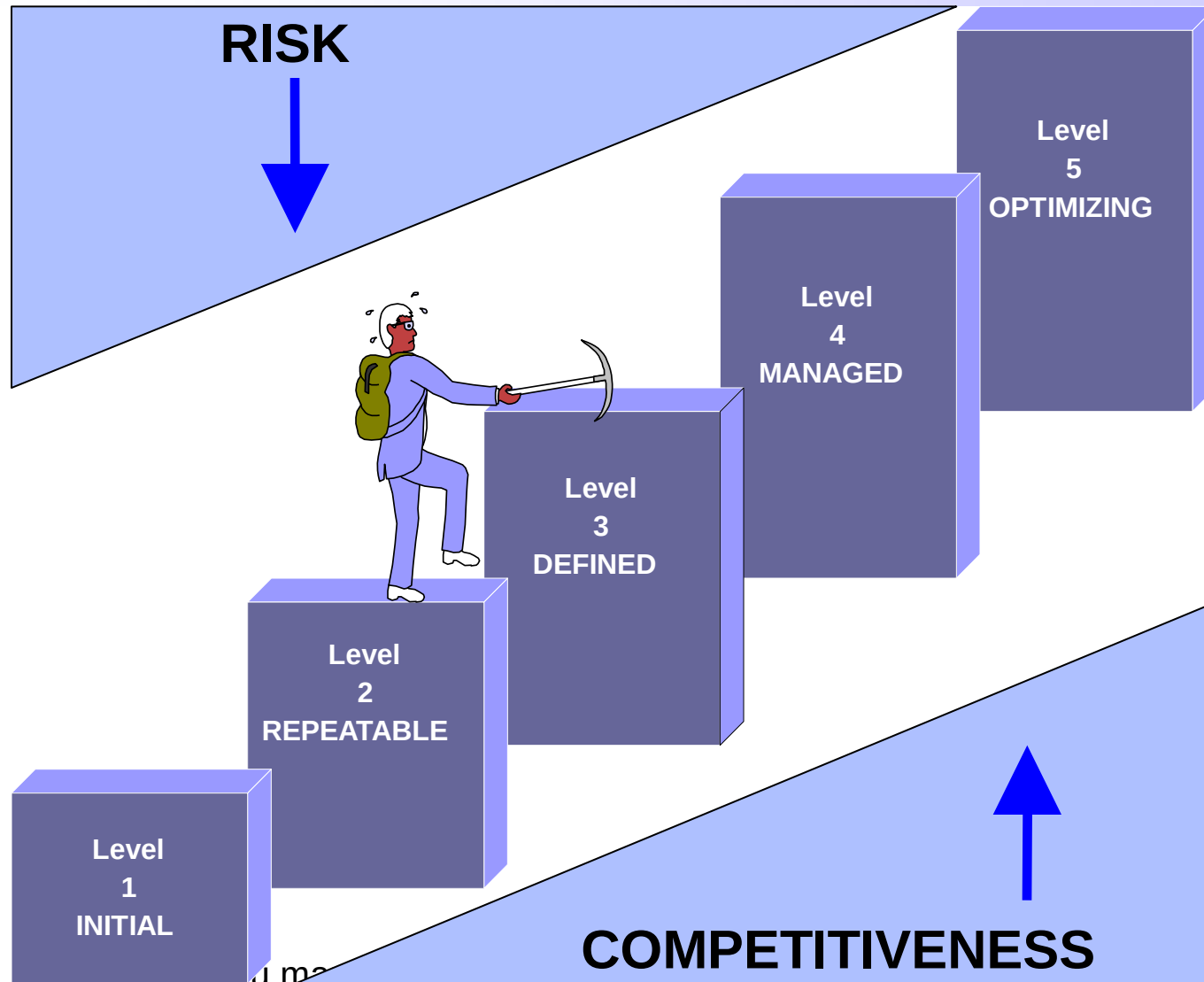
### • سطح ۴ – مدیریت شده (*Managed*)

- اهداف قابل اندازه گیری برای فرآیند و کیفیت محصول ایجاد شده است
- فرآیند و محصول با توجه به معیارها قابل درک و فهم هستند

### • سطح ۵ – در حال بهینه سازی (*Optimizing*)

- فرآیند توسعه نرم افزار متداوماً در حال کنترل و بهینه سازی است
- معیارهای بهبود همان معیارهای برقرار شده در سطح چهارم هستند

# فرآیند نرم افزار - مدل بلوغ قابلیت (ادامه)



# مدل‌های فرآیند نرم‌افزار

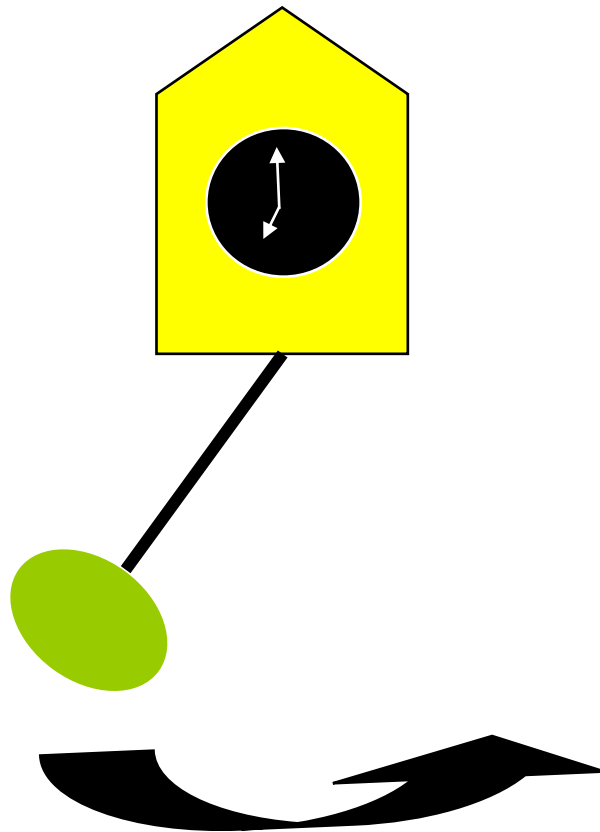
## ■ فرآیند و محصول

### ■ Product

- ADT
- OOP

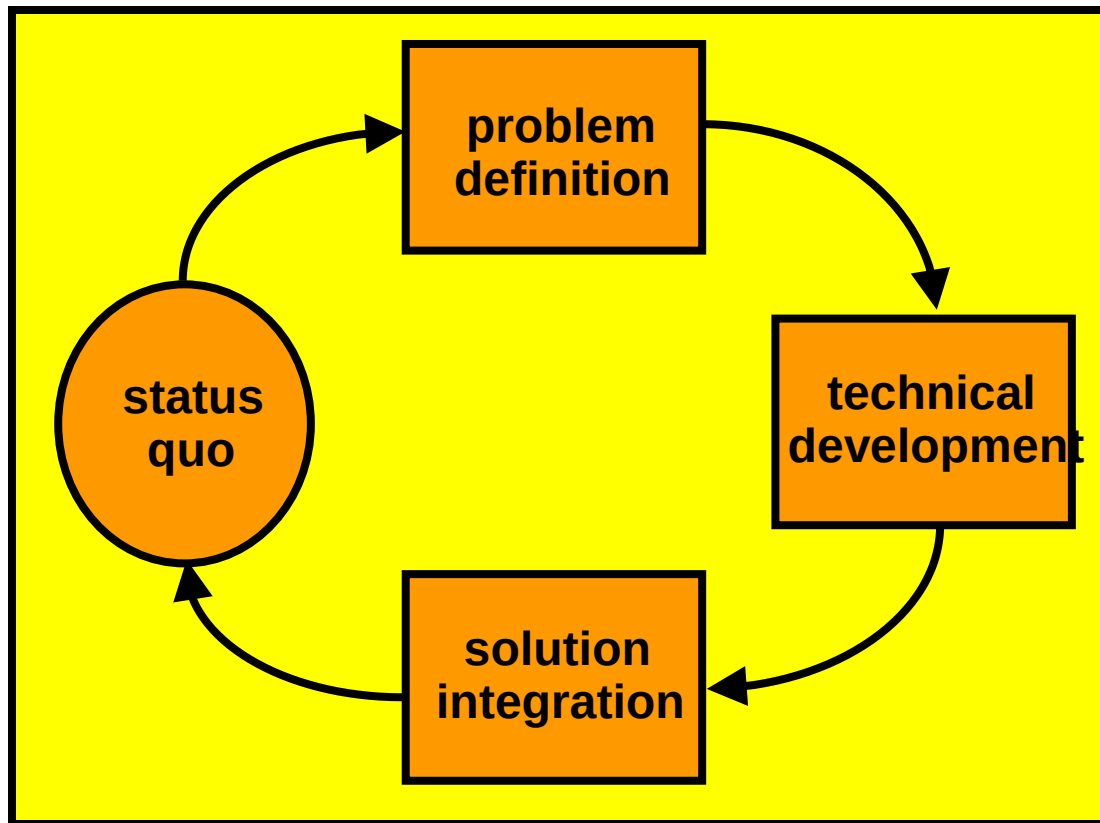
### ■ Process

- Structured Analysis
- OOAD
- Agile



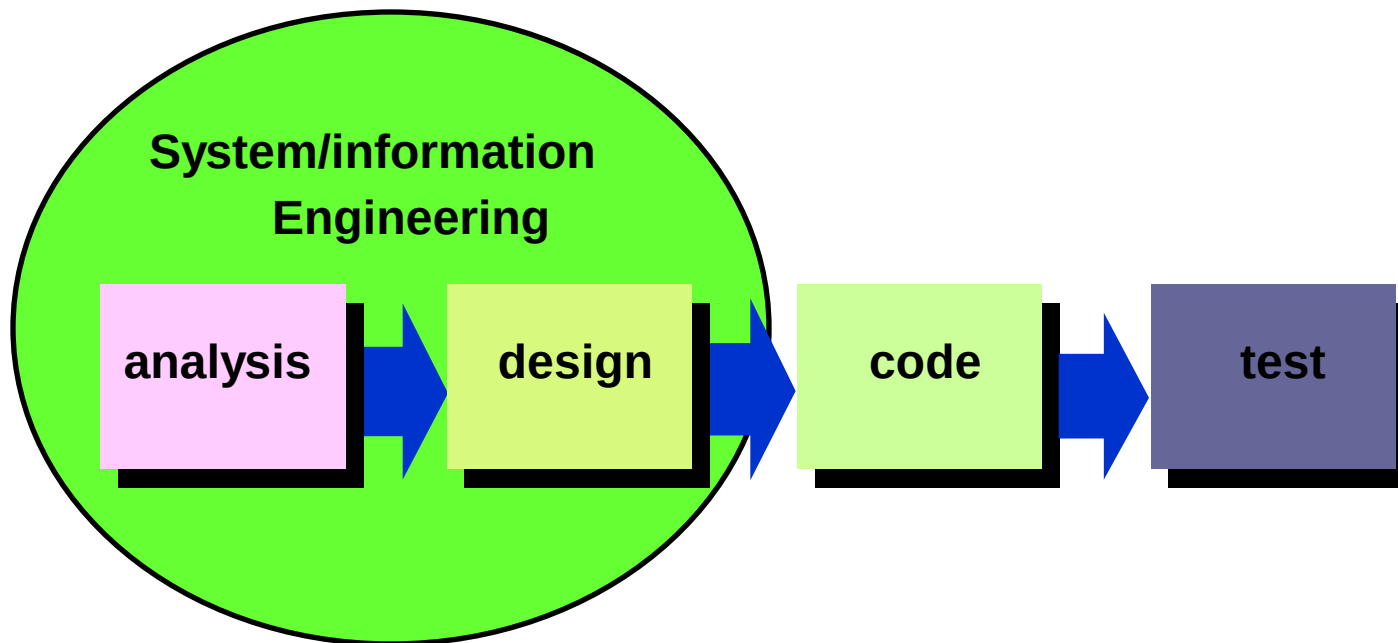
# مدل‌های فرآیند نرم‌افزار (ادامه)

■ فرآیند به‌عنوان حل‌کننده مسئله



# مدل‌های فرآیند نرم‌افزار (ادامه)

## ■ مدل خطی (*Linear Model*)



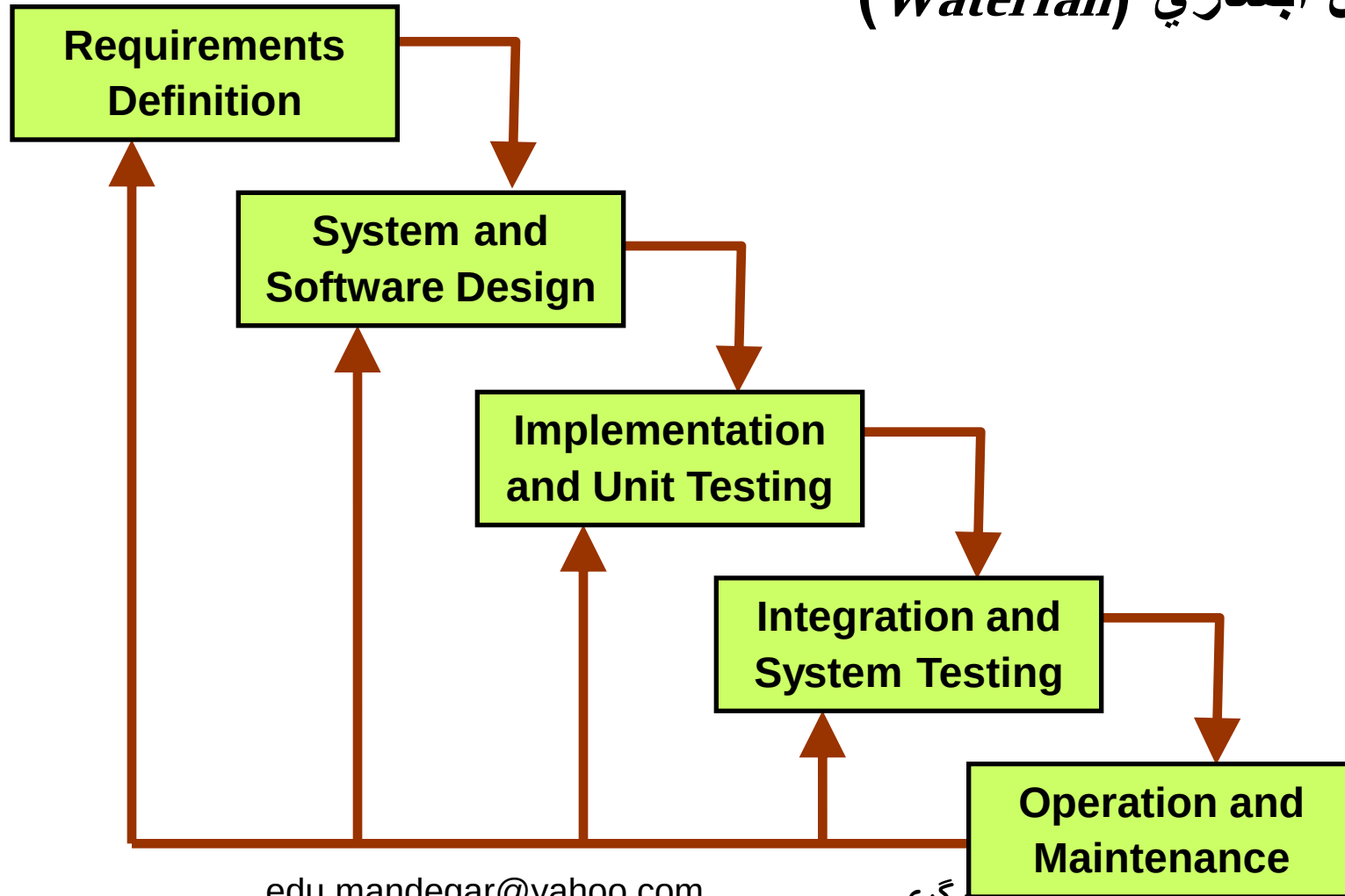
# مدل‌های فرآیند نرم‌افزار (ادامه)

## ■ مشکلات مدل خطی

- پروژه‌های واقعی کمتر دارای ترتیب خطی هستند و از مدل خطی پیروی نمی‌کنند
- اغلب برای مشتریان دشوار است که همه نیازمندی‌ها را در ابتدا بیان کنند
- مشتری باید تا انتهای پروژه صبر کند تا نرم‌افزار قابل اجرا ببیند

# مدل‌های فرآیند نرم‌افزار (ادامه)

■ مدل آبشاری (Waterfall)



# مدل‌های فرآیند نرم‌افزار (ادامه)

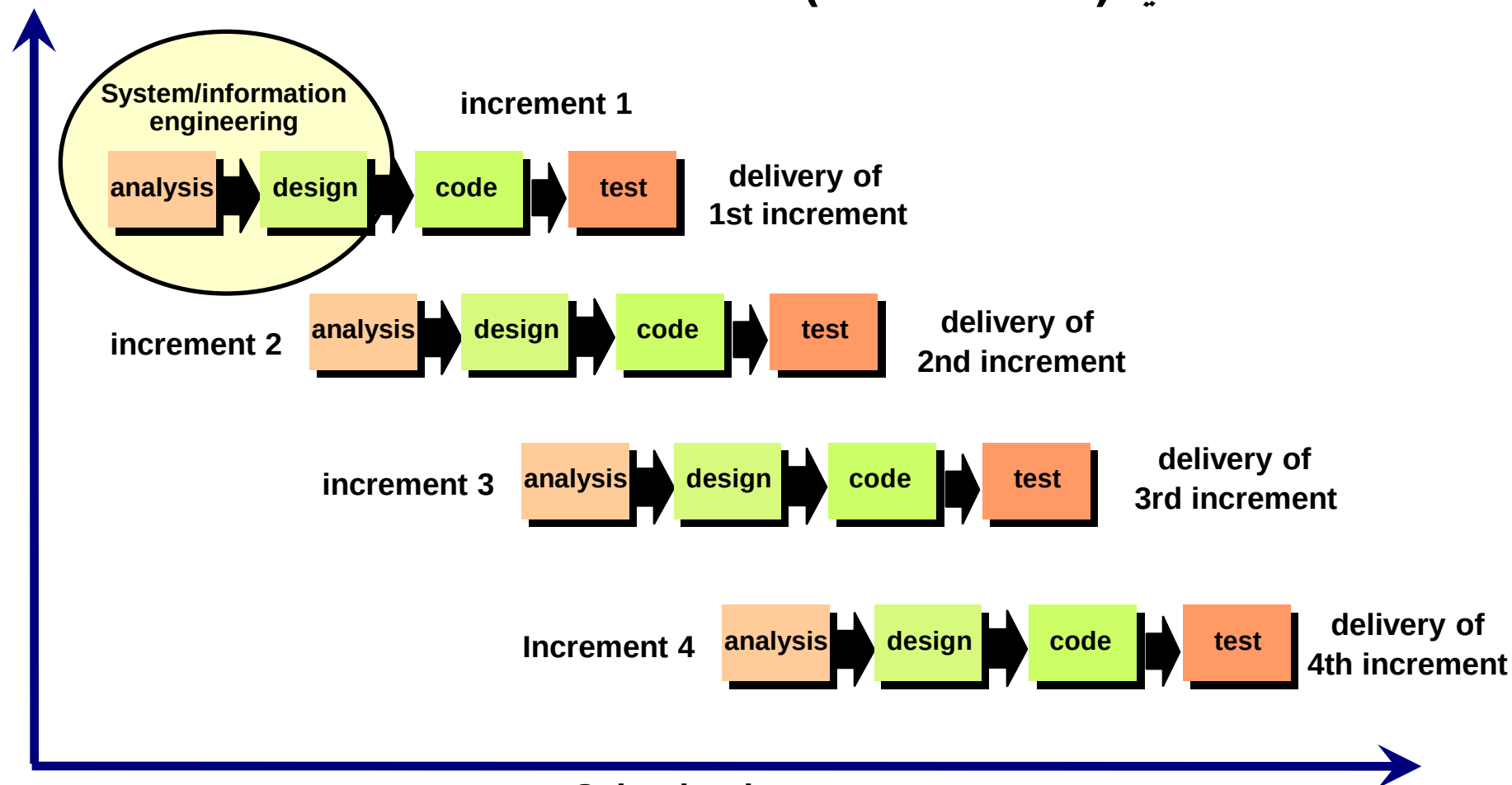
## ■ مشکلات مدل آبشاری

- تقسیم‌بندی نامنعطف پروژه به مراحل که تغییر نیازمندی‌های مشتریان را در نظر نمی‌گیرد
- وقتی مناسب است که
  - تمام نیازمندی‌ها در ابتدا شناسایی شده باشند
  - نیازمندی‌ها در هنگام توسعه تغییر نکنند



# مدل‌های فرآیند نرم‌افزار (ادامه)

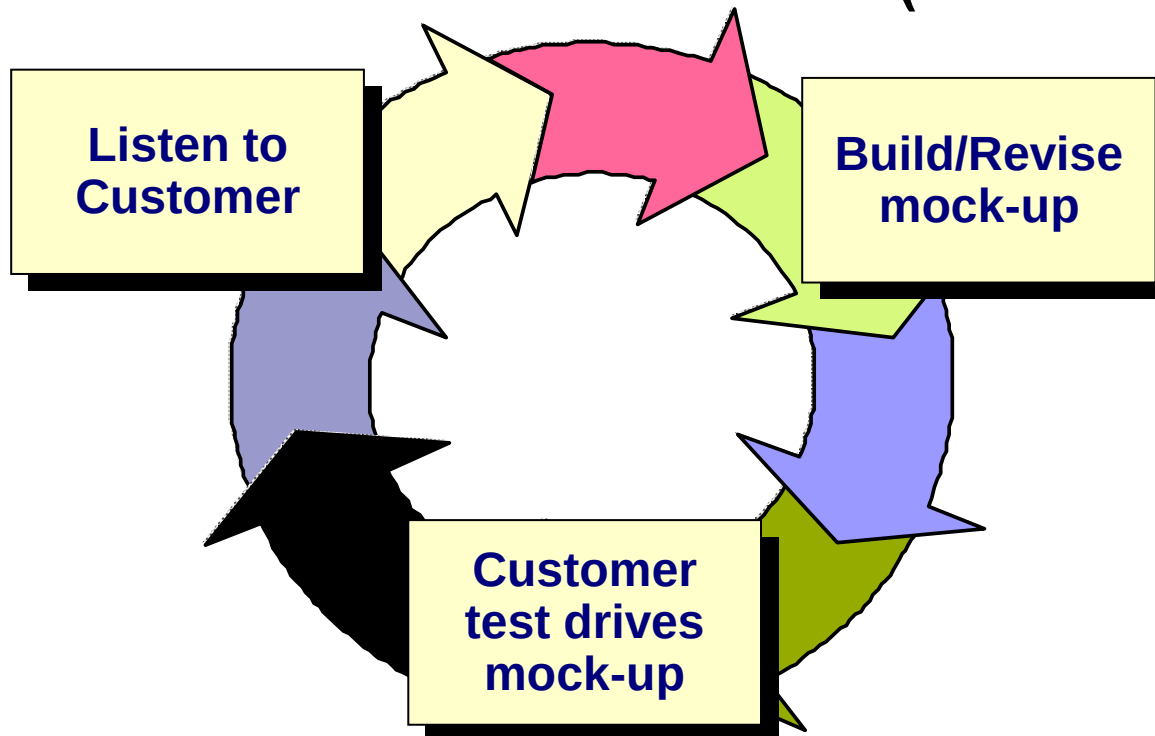
## ■ مدل افزایشی (*Incremental*)



# مدل‌های فرآیند نرم‌افزار (ادامه)

## ■ مدل نمونه‌سازی (*Prototyping*)

- نمونه تکاملی (*Evolutionary*)
- دورانداختنی (*Throwaway*)



# مدل‌های فرآیند نرم‌افزار (ادامه)

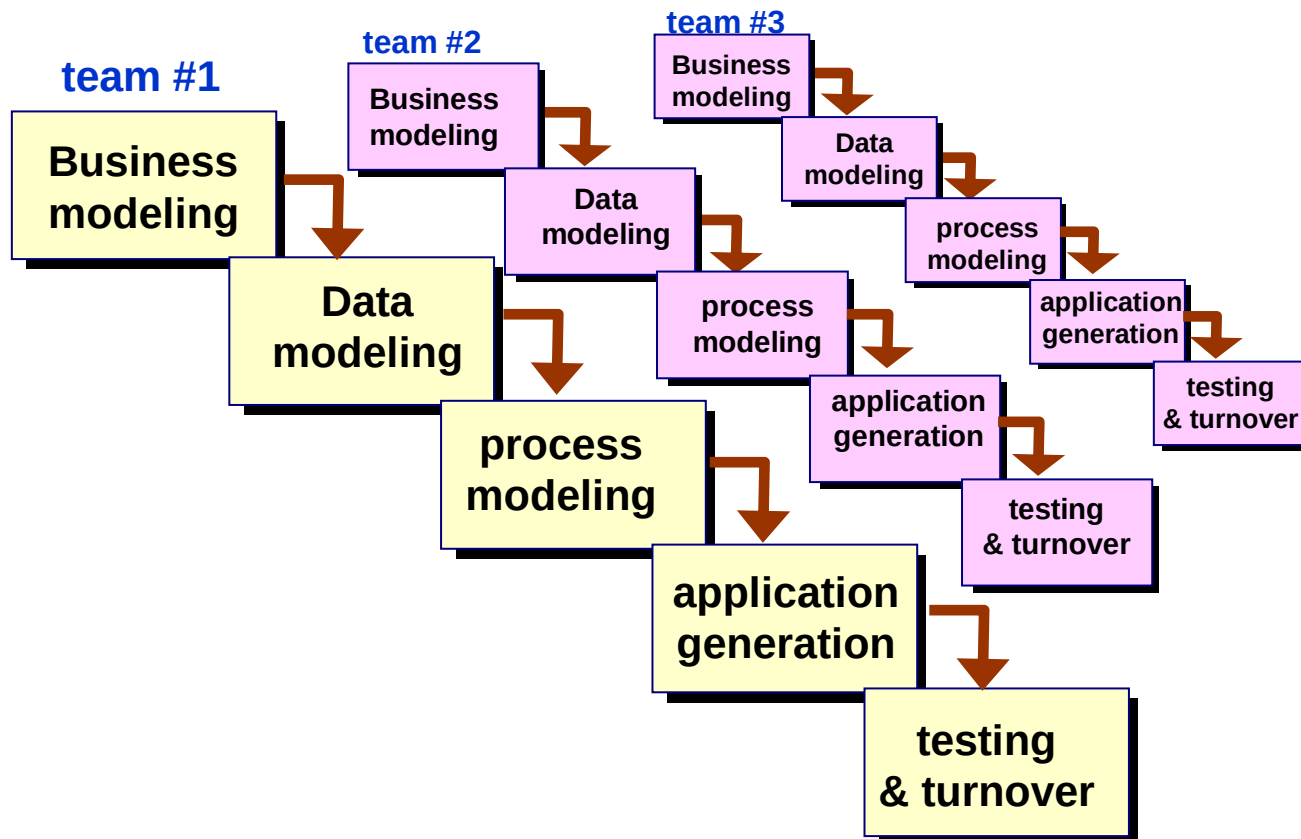
## ■ مشکل مدل نمونه‌سازی

- مشتری نسخه‌های مختلفی از نرم‌افزار می‌بیند!
- انتخاب ابزارهای اولیه توسعه می‌تواند منجر به توسعه نامناسب شود
  - توسعه‌دهندگان برای ارائه سریعتر نمونه، ابزار یا محیطی را به عنوان پیش‌فرض در نظر می‌گیرند که می‌تواند در ادامه مشکلاتی ایجاد نماید



# مدل‌های فرآیند نرم‌افزار (ادامه)

■ توسعه کاربرد سریع (*Rapid Application Development*)



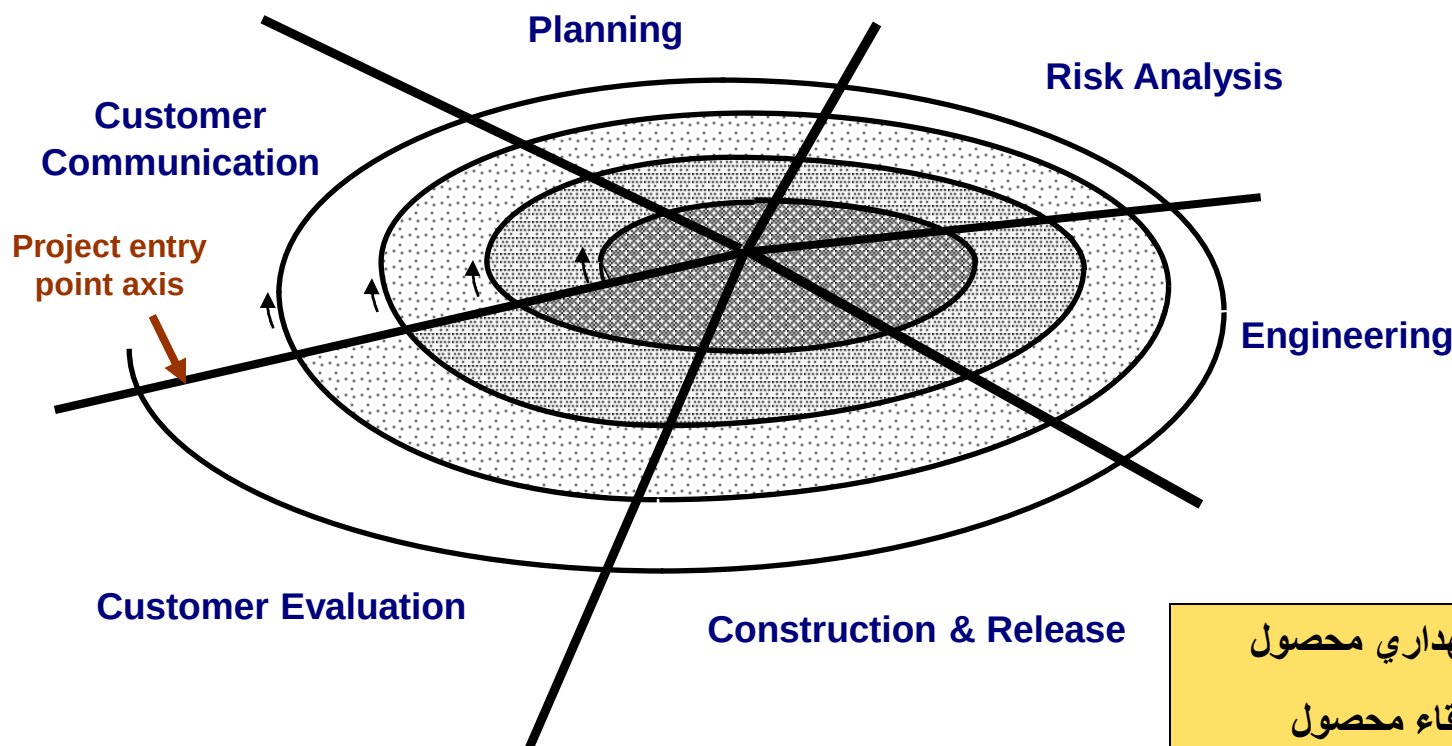
# مدل‌های فرآیند نرم‌افزار (ادامه)

## ■ مشکل مدل RAD

- در پروژه‌های بزرگ و قابل گسترش، این روش نیازمند منابع انسانی برای ایجاد تیم‌های مورد نیاز است
- این مدل تعهد زمانی زیادی برای تیم توسعه‌دهنده و مشتریان ایجاد می‌کند
- مخصوص نرم‌افزارهای خاص است
- در صورت بالا بودن ریسک فنی این روش مناسب نیست

# مدل‌های فرآیند نرم‌افزار (ادامه)

## ■ مدل حلزونی (Spiral Model)



پروژه‌های نگهداری محصول



پروژه‌های ارتقاء محصول

پروژه‌های توسعه محصول جدید

پروژه‌های طراحی مفهومی

# مدل‌های فرآیند نرم‌افزار (ادامه)

## ■ مشکلات مدل حلزونی

- (همانند سایر مدل‌ها) متقاعد کردن مشتری در مورد قابل کنترل بودن روش تکاملی دشوار است!
- این مدل نیاز به تجربه در ارزیابی ریسک دارد
- تجربه کمی در مورد اجرای این مدل وجود دارد، برخلاف مدل خطی و نمونه‌سازی. به همین سبب زمانی نیاز است تا تیم توسعه این روش را یاد بگیرند و اجرا کنند!

# مدل‌های فرآیند نرم‌افزار (ادامه)

## ■ مدل توسعه همروند

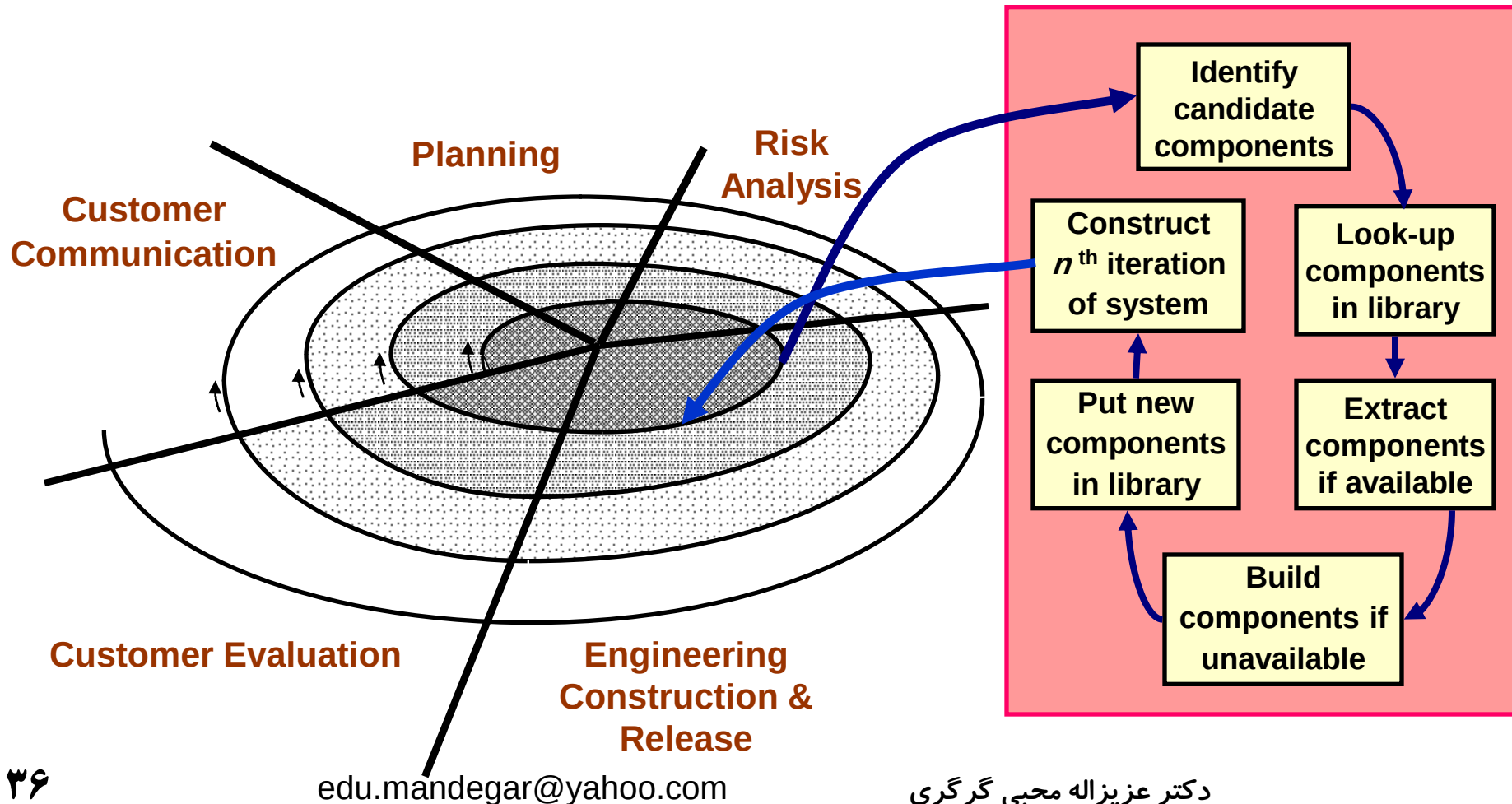
- بیشتر برای نرم‌افزارهای Client/Server مورد استفاده قرار می‌گیرد
- برای توسعه هر نوع نرم‌افزاری مناسب است و تصویر صحیحی از وضعیت جاری پروژه ارائه می‌دهد
- درک مدل توسعه برای تیم توسعه دشوار است





# مدل‌های فرآیند نرم‌افزار (ادامه)

■ توسعه براساس مؤلفه‌ها (*Component-based Development*)



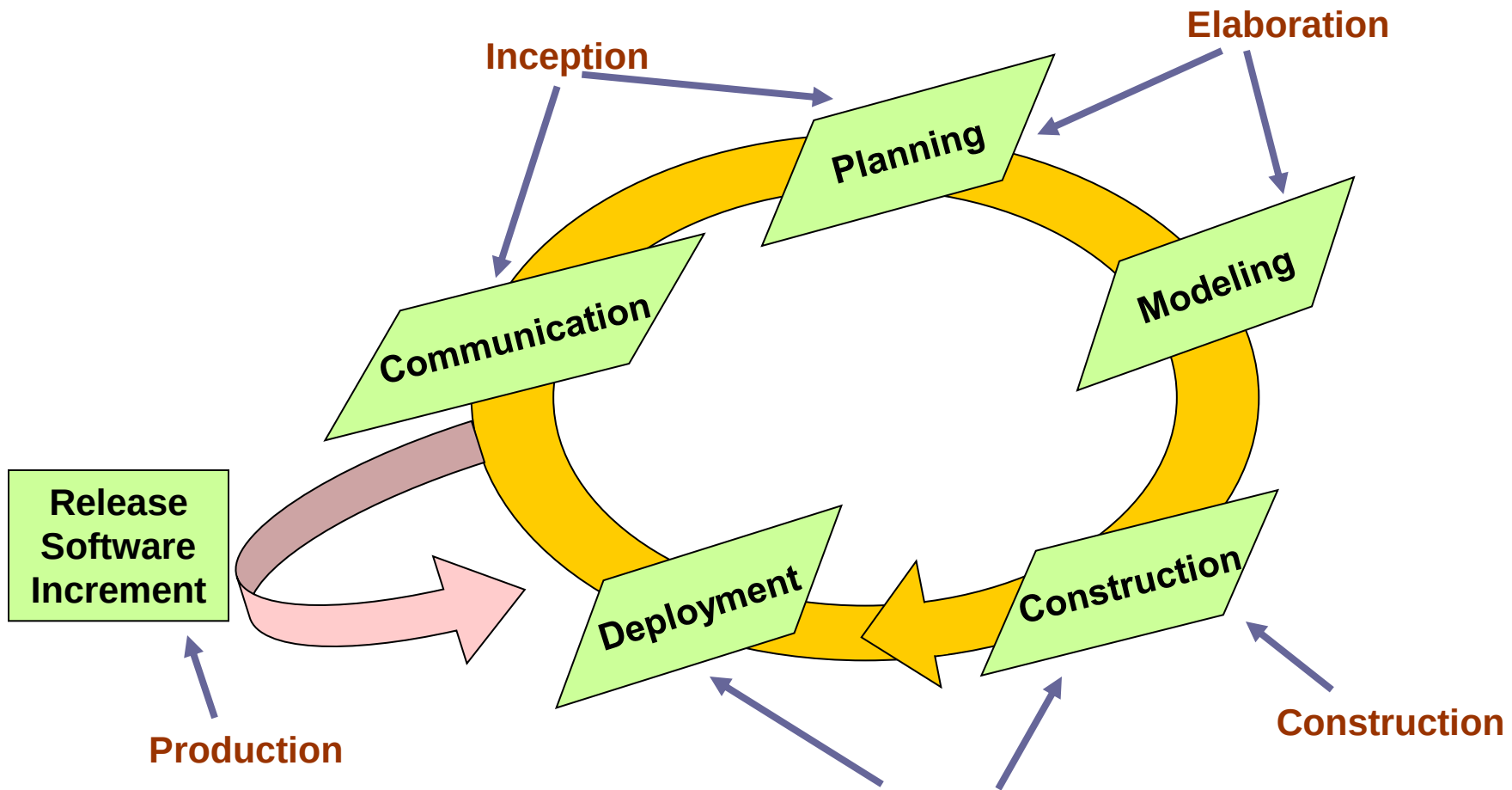
# مدل‌های فرآیند نرم‌افزار (ادامه)

## ■ توسعه براساس مولفه

- افزایش قابلیت استفاده مجدد
- ۷۰ درصد کاهش در زمان توسعه
- ۸۴ درصد کاهش در هزینه پروژه
- شاخص بهره‌وری ۲۶.۲ (روش‌های متداول ۱۶.۹)

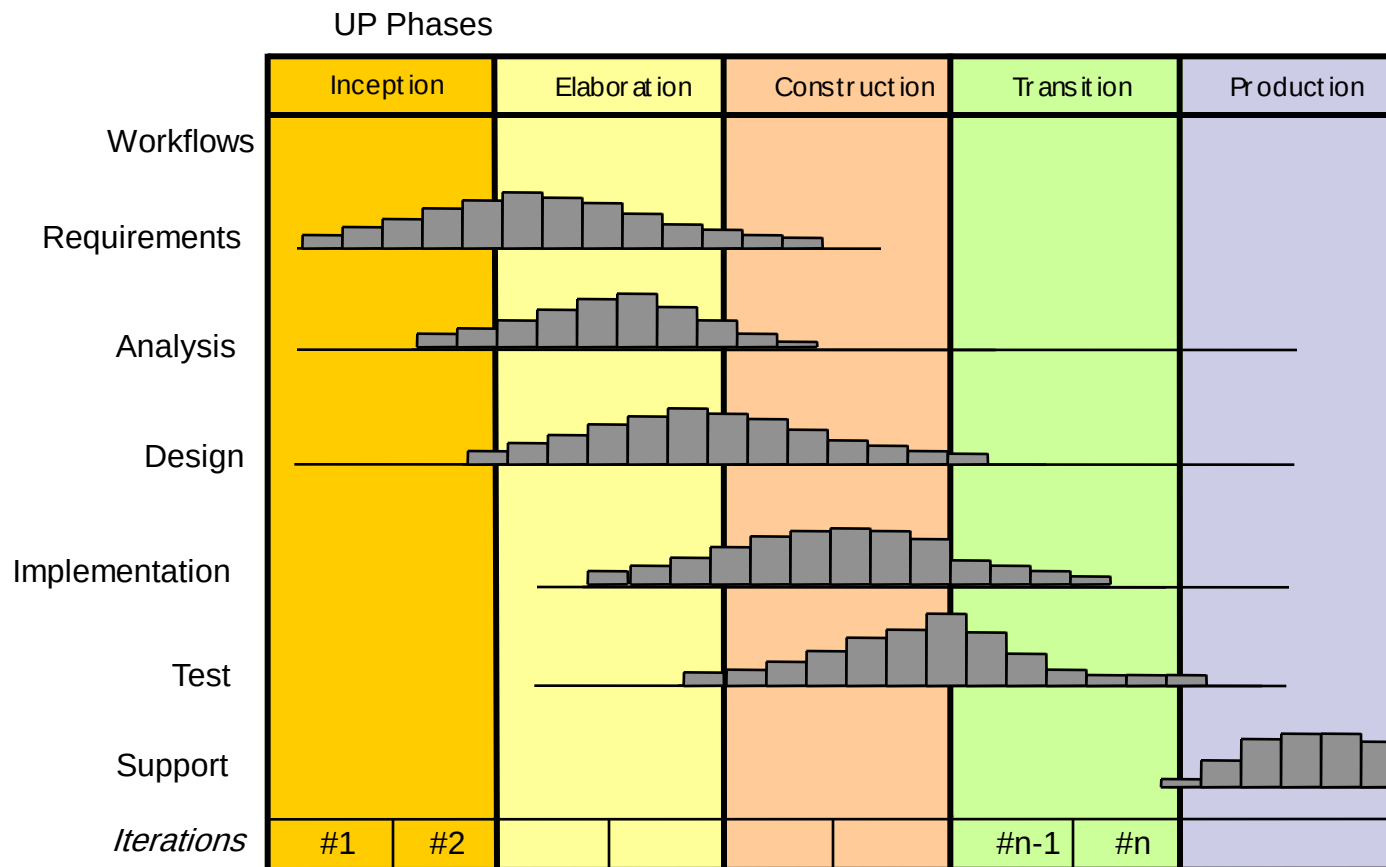
# مدل‌های فرآیند نرم‌افزار (ادامه)

## ■ فرآیند یکپارچه (Unified Process)



# مدل‌های فرآیند نرم‌افزار (ادامه)

## ■ فازهای فرآیند یکپارچه



# مدل‌های فرآیند نرم‌افزار (ادامه)

## ■ روش‌های رسمی (*Formal methods*)

- مجموعه‌ای از فعالیت‌ها که سبب توصیف ریاضی نرم‌افزار می‌شوند
- گونه خاصی از این روش: *Cleanroom Software Engineering*
- حذف ابهام با استفاده از تعاریف دقیق ریاضی
- کاربرد این روش دشوار است
  - بسیار زمان‌بر
  - نیاز به آموزش تیم توسعه
  - مشکل ارتباط با مشتری
  - کاربرد محدود

# مدل‌های فرآیند نرم‌افزار (ادامه)

## ■ تکنیک‌های نسل چهارم

- محدوده وسیعی از ابزارهای نرم‌افزاری
- قادر به تعیین برخی از خصوصیات نرم‌افزار در سطح بالا هستند
- می‌توانند به صورت اتوماتیک و بر مبنای مشخصات تعیین شده کد برنامه را ایجاد نمایند
- می‌تواند برای پروژه‌های مختلف بکار گرفته شود
- زمان مورد نیاز برای توسعه نرم‌افزار در پروژه‌های کوچک و متوسط را بطرز چشمگیری کاهش می‌دهد
- در حال حاضر دارای کاربردهای خاص هستند

# محصول بهتر یا فرآیند بهتر

- اگر فرآیند **ضعیف** باشد، بدون شک در **توسعه محصول** مشکل خواهیم داشت
- **حساسیت** بیش از حد روی فرآیند، **خلاقیت** در توسعه را کاهش می‌دهد
  - فرآیند باید محیطی برای بروز خلاقیت ایجاد نماید
  - تیم توسعه باید علاقه‌مند به پیروی از فرآیند و تولید محصول مناسب باشند
- دانشجو باید علاقه‌مند به درس و نشستن در کلاس باشد و گونه .... !!!

پایان