فصل چهارم– اصول و قواعد

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 7/e,* by Roger S. Pressman

Presented by: Hassan Tanabi
Hassan.Tanabi@Gmail.com

چه اصولی؟!!

- اصولی که روند اجرای فرایند (تمرین) را هدایت می کنند.
- فرایند، نقشه راه را برای رسیدن به موفقیت مهیا می کند. **تمرین** جزییات مورد نیاز برای حرکت در این مسیر را نشان می دهد.

Software Engineering Practice

- کمک می کند مفاهیم و قواعد مورد نیاز برای حرکت در این مسیر را بیاموزیم.

تمرین Practice

- در زمینه مهندسی نرم افزار، **تمرین** مجموعه فعالیت های فنی است که باید شما از روز اول، تا روز آخر، در هر لحظه انجام دهید تا به عنوان نرم افزار، از یک ایده یک محصول واقعی را استخراج کنید.

- فعالیت های فنی که تمام محصولات تعریف شده توسط مدل فرایند را تولید می کنند.

- المان های تمرین:
- **قواعد، مفاهیم**، روش ها، ابزارها

دانش مهندسی نرم افزار

- شما اغلب شنیده اید که در زمینه دانش توسعه نرم افزار، نیمی از آنچه را که طی ۳ سال آموخته اید، امروزه کهنه و منسوخ است.
- در زمینه دانش مرتبط با تکنولوژی این نظر درست است، اما نوعی دیگر از دانش توسعه نرم افزار – نوعی که من به عنوان "اصول مهندسی نرم افزار" می شناسم- اینگونه منسوخ نمی شوند.

- این اصول مهندسی نرم افزار به یک برنامه نویس حرفه ای در طول مسیر کاری اش به او خدمت می کند.

Steve McConnell

مقایسه مشتری و کاربر نهایی

- هر دو از مهمترین ذینفعان هستند.
- در برخی موارد مشتری و کاربر نهایی یکی هستند، اما در برخی پروژه ها متفاوت می باشند.

- یک مشتری شخص یا گروهی می باشد که:
  - اساسا درخواست ساخت نرم افزار را دارد.
  - اهداف کلی برای نرم افزار را مشخص می کند.
  - نیازمندیهای اولیه را مشخص می کند.
  - بودجه پروژه را تأمین می کند.
- کاربر نهایی شخص یا گروهی است که:
  - به صورت عملی از نرم افزار برای دستیابی به اهداف مورد نظر استفاده می کند.
  - جزییات اجرایی نرم افزار را مشخص می کند

اصول هدایت فرایند

۱- چالاک باشید:

- باید تفکر پایه ای چالاکی در رویکرد شما حاکم باشد.
- رویکرد فنی خود را تا حد ممکن ساده کنید.
- تا حد ممکن تصمیمات محلی بگیرید.

۲- در هر گام برروی کیفیت تمرکز کنید:

- شرط خروج از هر فعالیت، کار و یا وظیفه باید تمرکز برروی کیفیت محصول تولیدی باشد.

۳- برای تطبیق آماده باشید:

- فرایند یک مذهب نیست و تعصب در آن راه ندارد. هرگاه نیاز است، رویکرد خود را براساس مشکل، افراد یا خود پروژه تنظیم کنید.

اصول هدایت فرایند (ادامه)

۴- یک تیم موثر بسازید:

- فرایند در مهندسی نرم افزار مهم است، اما افراد نیز حضور دارند. تیم خود سازماندهی بسازید که اعتماد و احترام متقابل در آن وجود دارد.

۵- راهکارهایی به منظور برقراری ارتباط و هماهنگی در نظر بگیرید:

- پروژه ها به دلیل اینکه برخی اطلاعات مهم از قلم میافتند و یا ذینفعان نمی توانند تلاش هایشان را یکپارچه و هماهنگ کنند به شکست منجر می شوند.

۶- تغییرات را مدیریت کنید:

- رویکرد شما ممکن است رسمی یا غیر رسمی باشد، اما باید راهکاری به منظور مدیریت تغییرات وجود داشته باشد.

اصول هدایت فرایند (ادامه)

۷- ارزیابی ریسک:

- طی مراحل توسعه نرم افزار خیلی چیزها می تواند اشتباه شود، ضروری است که طرح ها را بررسی و ارزیابی کنید.

۸- محصولی تولید کنید که برای بقیه با ارزش باشد:

- تمام محصولات که در مراحل میانی تولید می شوند به افراد دیگر ارسال می شوند. نیازمندی ها به طراح، طرح به کدنویس و ...
- مطمئن شوید که محصول تولید شده اطلاعات لازم را دارد، ابهامی در آن وجود ندارد و ...

اصول هدایت تمرین

۱- تقسیم و غلبه:

- آنالیز و طراحی همیشه باید به مسئله های کوچکتر تقسیم شوند. (SoC)
- هر زیر مسئله وظیفه مندی های دارد که اغلب می توان به صورت جداگانه آن را توسعه داد.

۲- استفاده از انتزاع:

- انتزاع راه ساده سازی یک المان پیچیده از سیستم می باشد که برای برقراری ارتباط در یک فاز استفاده می شود.
- در مهندسی نرم افزار سطوح مختلفی از انتزاع استفاده می شود. در آنالیز و طراحی از سطوح بالاتر انتزاع شروع می کنیم و به مرور زمان به سمت سطوح پایین تر حرکت می کنیم.

اصول هدایت تمرین (ادامه)

۳- تلاش برای سازگاری:

- یک زمینه آشنا استفاده از نرم افزار را ساده تر می کند.
- برای مثال در طراحی یک منو برای سایت بهتر است از رنگبندی های سازگار استفاده کنیم و یا از آیکون های آشنا کمک بگیریم.

۴- تمرکز بر روی انتقال اطلاعات:

- توجه ویژه به رابط های آنالیز، طراحی، ساخت و تست.

۵- ساختن نرم افزاری که پیمانه ای موثر را نمایش دهد:

- جدا کرن وظایف (اصل اول) فلسفه خاصی را برای نرم افزار برپا می کند. پیمانه ای بودن مکانیزمی برای درک بهتر این فلسفه ارائه می دهد.

اصول هدایت تمرین (ادامه)

۶- جستجوی الگوها:

- هدف الگوها ایجاد ادبیاتی مشترک برای کمک به توسعه دهندگان در حال مشکلات تکراری است که در اکثر پروژه های نرم افزاری دیده می شود.

۷- هروقت ممکن است، مشکل و راه حل آن را از منظرهای مختلف نمایش بدهید:

- دید وسیعتری به ما می دهد، برای مثال نمایش نیازمندی ها با دو مدل رفتاری و وظیفه.

- **۸- به خاطر داشته باشید که کسی از نرم افزار نگهداری خواهد کرد:**
- نرم افزار تصحیح می شود (خطا)، افزوده می شود (نیاز ذینفعان)، تطبیق داده می شود (شرایط محیطی)

اصول تعامل

۱- گوش کردن:

- تلاش کنید به بر روی صحبت های گوینده تمرکز کنید، بجای اینکه برای واکنش به آن آماده شوید.
- تلاش برای شفاف سازی

۲- قبل از گفتگو آماده شوید:

- قبل از ملاقات با بقیه زمانی به منظور یادگیری مسئله صرف کنید، در مورد موضوع مطالعه کنید و اطلاعات لازم را کسب کنید.

۳- یک نفر باید این فعالیت را تسهیل کند:

- تمام گفتگوها به منظور (۱) حرکت در راستای مورد نظر، (۲) وساطت در تضادها، (۳)تضمین رعایت سایر اصول، باید یک رهبر داشته باشد.

اصول تعامل (ادامه)

۴- ارتباط چهره به چهره بهترین راه است

۵- یادداشت برداری و ثبت تصمیمات:

- یک نفر باید وظیفه ثبت تمام نکات مهم و تصمیمات را داشته باشد.

۶- تلاش برای همکاری:

- همکاری و توافق وقتی اتفاق می افتد که دانش اعضا با یکدیگر برای توصیف اجزای سیستم و وظایف آن ادغام گردد.

۷- تمرکز داشته باشید؛

- پس از پایان و نتیجه گیری در مورد یک موضوع سراغ موضوع دیگر بروید

۸- اگر چیزی غیر شفاف است، تصویری از آن رسم کنید

اصول تعامل (ادامه)

۹- (a) اگر برروی چیزی توافق کردید از آن موضوع بگذرید (b) اگر نتوانستید به توافق برسید بازهم بگذرید. (c) اگر وظیفه یا ویژگی خاصی غیرشفاف است و نمیتوان در آن لحظه آن را روشن نمود، بازهم بگذرید.

۱۰- مذاکره رقابت یا بازی نمی باشد. وقتی مفید است که هر دو طرف برنده باشند.

اصول برنامه ریزی

۱- مشخص کردن محدوده پروژه:

- ممکن نیست بدون اینکه بدانید به کجا می روید بتوانید از یک نقشه راه استفاده کنید. محدوده، مقصد را برای تیم نرم افزاری مشخص می کند.

۲- مشتری را در فعالیت های برنامه ریزی شرکت دهید:

- مشتری اولویت بندی و محدودیت های پروژه را مشخص می کند.

۳- تکراری بودن برنامه ریزی را درک کنید:

- برنامه ریزی حکاکی روی سنگ نیست. با شروع کار خیلی از موارد تغییر می کنند.

اصول برنامه ریزی (ادامه)

۴- براساس آنچه که می دانید تخمین بزنید:

- تخمین میخواهد برآوردی از هزینه، تلاش و مدت زمان انجام کار را برمبنای وضعیت فعلی ارائه دهد.

۵- هنگام برنامه ریزی ریسک را در نظر بگیرید:

- اگر به ریسکی با احتمال وقوع بالا برخوردید، لازم است در برنامه ریزی آن را مد نظر قرار دهید

۶- واقع بین باشید:

- افراد ۱۰۰٪ روز را فعالیت نمی کنند.

۷- سطح جزییات را در هنگام برنامه ریزی در نظر بگیرید.

اصول برنامه ریزی (ادامه)

۸- مشخص کنید چگونه می خواهید کیفیت را تضمین کنید:

- برنامه باید مشخص کند چگونه تیم کیفیت را تضمین می کند.

۹- مشخص کنید چگونه می خواهید تغییرات را مرتفع کنید

- برای مثال مشتری هرزمان بخواهد میتواند تغییرات اعمال کند؟هزینه؟!!

۱۰- برنامه را به صورت دوره ای بررسی کنید و تنظیمات لازم را اعمال نمایید

- پروژه های نرم افزاری با برنامه ریزی ثابت در یک زمان شکست می خورند

اصول مدل سازی

- در مهندسی نرم افزار دو دسته مدل می توان ایجاد کرد:
- **مدل های نیازمندی ها**
  - نیازمندی های مشتری را در سه حوزه نمایش می دهد:
    - اطلاعات
    - وظیفه مندی
    - رفتاری
- **مدل های طراحی**
  - ویژگیهای نرم افزارکه به ساخت کارای آن کمک میکند:
    - معماری
    - رابط کاربری
    - جزییات سطح کامپوننت

اصول مدل سازی نیازمندیها

۱- دامنه اطلاعات مسئله باید مشخص شود

- تمام اطلاعاتی که به سیستم وارد می شوند (از کاربر نهایی، از سیستم های دیگر، ...).
- تمام اصلاعاتی که از سیستم خارج می شوند (از رابط کاربری، گزارشات، نمودارها، ...)

۲- وظایفی که نرم افزار انجام می دهد باید مشخص شوند

- وظایف را می توان در سطوح مختلف انتزاع بیان کرد، از هدف کلی تا تمام جزییات موجود.

۳- رفتار نرم افزار باید نمایش داده شود

- تعامل نرم افزار با محیط
- ورودی های داده شده توسط کاربر نهایی
- ...

اصول مدل سازی نیازمندیها (ادامه)

۴- مدل هایی که اطلاعات، وظایف و رفتارهای نرم افزار را بیان می کند باید تقسیم شوند.

- یک مسئله پیچیده به چند زیر مسئله تقسیم می شود، هر زیر مسئله نسبتا ساده تر می باشد. (SoC)

۵- آنالیز باید از جزییات اطلاعات ضروری آغاز و به سمت جزییات پیاده سازی حرکت کند.

- مدل سازی نیازمندیها از توصیف مسئله از نگاه کاربر نهایی آغاز می شود

اصول مدل سازی طراحی

۱- مدل طراحی باید قابل ردگیری در مدل نیازمندیها باشد.

- مدل نیازمندی اطلاعات، وظیفه مندیها و رفتار سیستم را نمایش می دهد. مدل طراحی هرکدام از این اطلاعات را به یک معماری و یا زیرسیستم ترجمه می کند، هرکدام از این المان ها باید در مدل نیازمندی قابل ردگیری باشند.

۲- همیشه معماری سیستمی که باید ساخته شود را در نظر بگیرید.

- معماری نرم افزار اسکلت سیستم می باشد که بر روی رابط کاربری، ساختار داده ها و ... تأثیرگذار است. به همین دلیل برای طراحی باید از معماری آن آغاز کنیم.

۳- طراحی داده به اندازه طراحی توابع پردازش اهمیت دارد.

- یک طراحی داده خوب می تواند جریان برنامه را ساده تر کند، طراحی و پیاده سازی کامپوننت ها را آسان تر می کند و بطور کلی پردازش را کاراتر می گرداند.

اصول مدل سازی طراحی (ادامه)

۴- رابط کاربری (داخلی و خارجی) باید با دقت طراحی شود

- یک رابط کاربری خوب طراحی شده ادغام و تست را ساده تر می کند.

۵- رابط کاربری باید بر اساس نیازهای کاربر نهایی تنظیم شود. اگرچه همواره باید تأکید برروی استفاده ی آسان باشد.

- رابط کاربری خروجی قابل مشاهده نرم افزار می باشد. یک رابط کاربری ضعیف اغلب در دید مشتری به معنی نرم افزار بد است! (بدون توجه به وظایفی که انجام می دهد!)

۶- طراحی کامپوننت ها باید مستقل از وظیفه مندی ها باشد.

۷- کامپوننت ها باید تاحد ممکن غیروابسته به یکدیگر و به محیط بیرونی باشند.

- هرچقدر میزان اتصال کامپوننت ها بیشتر باشد میزان انتشار خطا نیز افزایش می یابد و نگهداری نرم افزار سخت تر می گردد.

۸- مدل های طراحی باید به راحتی قابل فهم باشند.

- هدف طراحی تعامل اطلاعات برای کدنویس ها، کسانی که نرم افزار را تست می کنند و کسانی که در آینده از نرم افزار نگهداری می کنند است. طراحی نامفهوم این تعامل را غیرموثر می کند.

اصول مدل سازی طراحی (ادامه)

۹- طرح باید متناوبا توسعه داده شود. با هر تکرار، طراح باید به دنبال سادگی بیشتر باشد.

- تکرار های آخر باید طرح را تا حد ممکن ساده کنند.

- **با رعایت این اصول، طراحی ساخته شده تمام فاکتورهای داخلی و خارجی کیفی را دارا می باشد.**
- **فاکتورهای خارجی کیفیت:**
  - ویژگیهایی که توسط کاربر نهایی قابل مشاهده هستند. مانند: سرعت، صحت، قابلیت استفاده
- **فاکتورهای داخلی کیفیت:**
  - این فاکتورها برای مهندسین نرم افزار اهمیت دارد. طراحی با کیفیت بالا از نظر تکنیکی و فنی

اصول مدل سازی چالاک

۱- هدف اصلی تیم نرم افزاری ساخت نرم افزار است نه مدل ها

- مدل ها باید در راستای ارائه زودهنگام نرم افزار باشند.

۲- سبک سفر کنید، به همان اندازه که نیاز دارید مدل ایجاد کنید.

۳- بدنبال ساخت ساده ترین مدل ها برای نمایش مسئله یا نرم افزار باشید.

۴- مدل هایی بسازید که بتوان تغییرات را در آنها اعمال نمود.

۵- قادر باشید برای هر مدل ایجاد شده یک هدف صریح بیان کنید.

- اگر نمیتوانید این کار را انجام دهید وقت خود را برای آن مدل صرف نکنید.

اصول مدل سازی چالاک (ادامه)

۶- مدل را براساس سیستم موجود تنظیم کنید.

- مدلی که برای یک بازی طراحی می کنید با مدل طراحی شده برای یک سیستم جاسازی شده متفاوت است.

۷- تلاش کنید مدل های سودمند بسازید، اما بدنبال ساختن مدل های کامل نباشید.

۸- بدنبال قواعد ساخت مدل نباشید، اگر محتوی را به خوبی نمایش میدهد، قواعد در اولویت بعدی قرار دارند.

۹- اگر بطور حسی به این نتیجه رسیدید که مدل شما ایراد دارد، باید نگران باشید، اگرچه روی کاغذ همه چیز مرتب است.

۱۰- هرچه زودتر بازخورد مدل را دریافت کنید.

اصول ساخت

- فعالیت ساخت، بر روی مجموعه ای از وظایف **کد نویسی** و **تست** در راستای ارائه نرم افزار اجرایی و آماده تحویل به کاربر نهایی و یا مشتری تأکید دارد.

- اغلب تست در سطح کامپوننت

Unit Test

- تست یکپارچگی

Integration Testing

- تست تأیید

Validation Testing

- تست پذیرش

Acceptance Testing

اصول ساخت (ادامه)

- مفاهیم و اصول کد نویسی:
  - مشابه سبک برنامه نویسی، زبان های برنامه نویسی و ...

- مفاهیم و اصول تست:
  - طراحی آزمایشاتی که به صورت خودکار و سیستماتیک کلاس های مختلف خطا را پوشش دهد.
    - به منظور کمینه هزینه تلاش و زمان

اصول ساخت (ادامه)

Preparation Principles

- پیش از نوشتن یک خط کد مطمئن شوید که:
  - مسئله ای که قرار است حل کنید را کامل فهمیده اید.
  - مفاهیم و قواعد پایه ای طراحی را می دانید.
  - زبان برنامه نویسی را انتخاب کنید که نیازهای نرم افزاری که قرار است بسازید و محیط اجرای آن را براورده می کند.
  - محیط برنامه نویسی انتخاب کنید که ابزاری را به منظور سهولت کارتان در اختیارتان قرار می دهد.
  - واحدهای تستی بسازید که می توان با ساخت کامپوننت آن تست را برروی آن اعمال کنید.

اصول ساخت (ادامه)

Coding Principles

- بعد از شروع کد نویسی مطمئن شوید که:
  - الگوریتم خود را مجبور به قرارگیری در برنامه نویسی ساخت یافته کنید.
  - از مزیت برنامه نویسی دو نفره استفاده کنید.
  - ساختار داده ای استفاده کنید که نیازهای طراحی را برآورده کند.
  - معماری نرم افزار را درک کنید و رابط کاربری بسازید که با این معماری سازگار باشد.
  - شرط های منطقی را تاحد ممکن ساده طراحی کنید.
  - حلقه های تودرتو را طوری طراحی کنید که تست آنها آسان باشد.
  - از نامهایی برای متغیرها استفاده کنید که با معنی باشد و از استانداردهای کدنویسی محلی استفاده کنید.
  - استفاده از توضیحات، خطوط خالی و نمایش بصری که به فهم کد کمک می کند.

اصول ساخت (ادامه)

Validation Principles

- بعد از کامل کردن اولین انتشار کد، مطمئن شوید که:
  - کد را مرور کنید.
  - واحدهای تست را اجرا کنید و خطاهایی که پوشش ندادید را اصلاح کنید.
  - کد را بازنویسی کنید

Refactor

اصول تست

۱- تمام آزمایش ها باید قابل نگاشت به نیازهای مشتری باشد.

۲- قبل از شروع آزمایش آن را به خوبی طراحی کنید.

۳- اصل Pareto در تست نرم افزار برقرار است.

۴- تست در ابعاد کوچک آغاز و در مراحل بعدی در ابعاد بزرگتر انجام می شود.

۵- تست کامل برنامه امکان پذیر نمی باشد.

اصول توسعه

۱- انتظارات مشتری باید مدیریت شود

- اغلب مشتری انتظار بیشتری از آنچا ارائه داده اید دارد و این مسئله موجب ناامیدی مشتری می گردد.

۲- یک بسته کامل قابل ارائه باید آماده و آزمایش گردد.

۳- باید پشتیبانی مناسبی قبل از ارائه نرم افزار در نظر گرفته شود.

۴- دستورالعمل ها و راهنماهای مناسبی به منظور استفاده برای کاربر نهایی فراهم گردد.

۵- نرم افزار خطا دار باید ابتدا رفع خطا گردد و سپس تحویل داده شود.

خسته نباشید