به نام پروردگار دانایی

طراحی سیستم‌های شی‌گرا

مقدمه‌ای بر اصول شی‌گرایی – بخش اول

سید کاوه احمدی

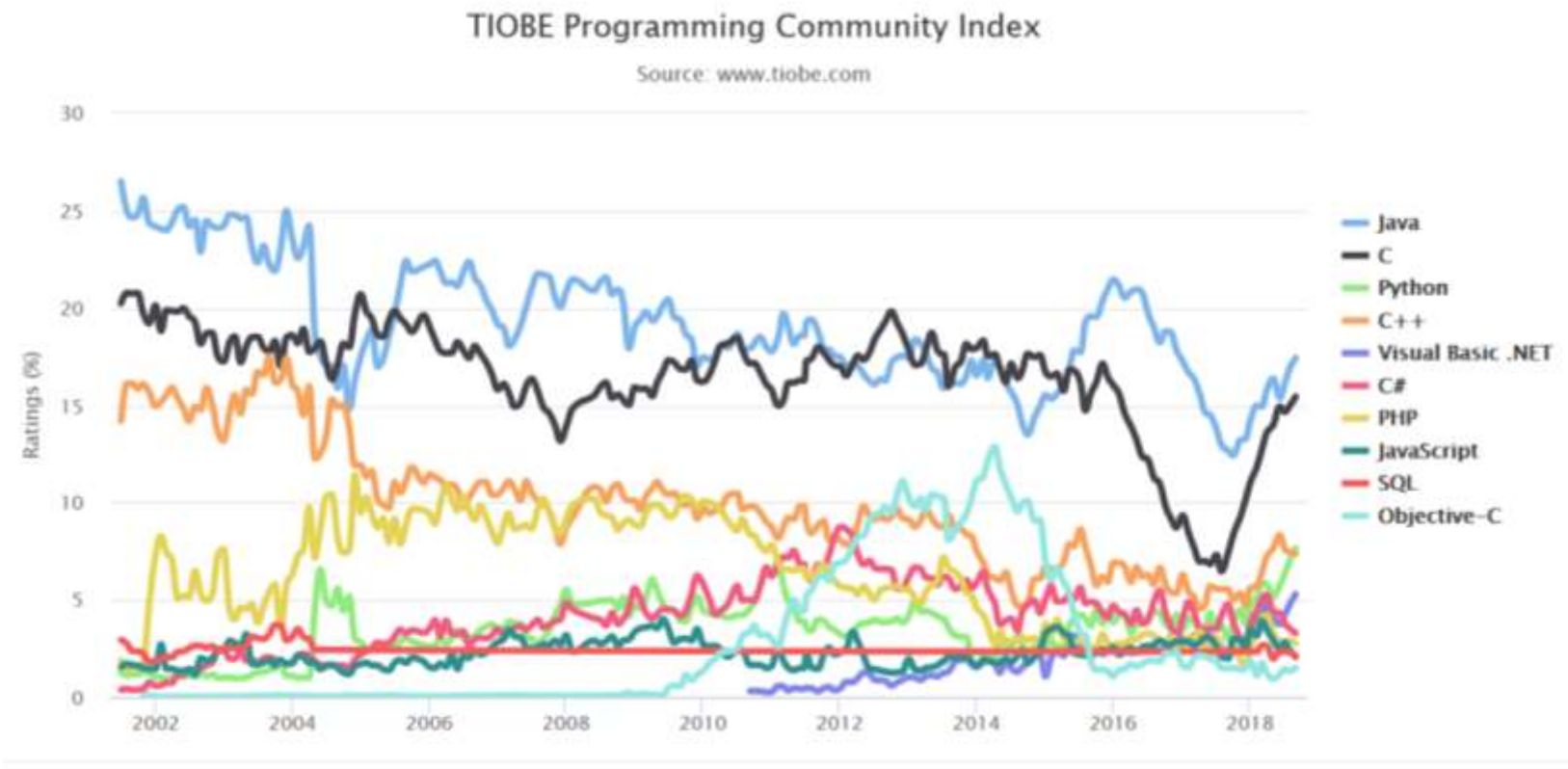
از چه زبان برنامه‌نویسی‌ای استفاده می‌کنیم؟

- این درس، کلاس سینتکس نیست.
- انتخاب زبان برنامه نویسی بر عهده شماست.
  - تمامی مباحث کمابیش در زبان‌های برنامه‌نویسی شی‌گرا وجود دارند.
- از آنجایی که مجبوریم برای نمایش مفاهیم از یک زبان برنامه‌نویسی بهره بگیریم، از زبان جاوا در کلاس استفاده می‌کنیم.

چرا جاوا

- آشنایی با یک خانواده جدید از زبان‌های برنامه‌نویسی.
- جاوا همه جا اجرا می‌شود!
- زبان اول در کارهای آکادمیک و پروژه‌های `Enterprise`.
- رعایت صحیح کلیه اصول شی‌گرایی
- اندروید!

چرا جاوا



- TIOBE programming community index is a measure of popularity of programming languages
- TIOBE stands for "The Importance of Being Earnest" which is taken from the name of a comedy play written by Oscar Wilde at the end of the nineteenth century.
- **The index is calculated from the number of search engine results for queries containing the name of the language**

چرا جاوا: BlueJ

- یک محیط توسعه مجتمع (IDE: Integrated Development Environment)

  برای یادگیری برنامه‌نویسی جاوا

  - هدف از توسعه این IDE آموزش برنامه‌نویسی شی‌گرا است
  - در طول درس از BlueJ استفاده خواهیم کرد.

- IDEهای مناسب برای کاربردهای حرفه‌ای:
  - **NetBeans**
  - **JetBrains IntelliJ IDEA**
  - **Eclipse**

- یک کارشناس نرم‌افزار باید چه چیزهایی بداند؟

انواع نرم‌افزار

- برنامه (Program Software)
  - مجموعه‌ای از دستورات/توابع/کلاس‌ها که برای انجام عملیاتی خاص نوشته می‌شود.
    - برنامه‌ای که اعداد اول را باز می‌گرداند.
    - برنامه‌ای که کوتاه‌ترین مسیر را در یک گراف پیدا می‌کند.
    - برنامه جستجو در ...

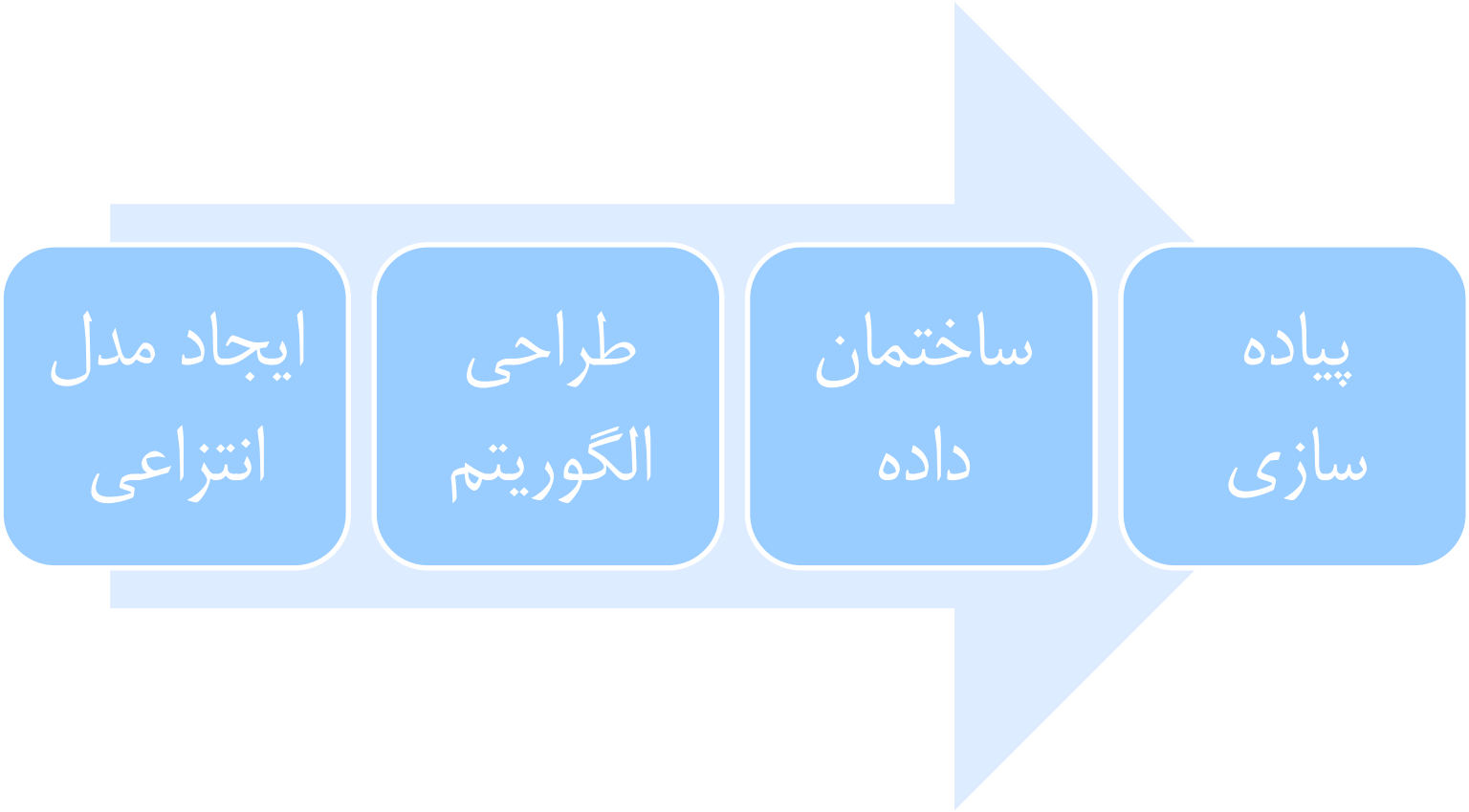
انواع نرم‌افزار

- نرم‌افزار کاربردی و تجاری (Application Software)
  - نرم‌افزاری شامل چندین برنامه که برای پاسخگویی به یک نیاز خاص نوشته می‌شود.
    - عمدتا مبتنی بر پایگاه داده
    - نرم‌افزار حسابداری، نرم‌افزار تحلیل ریسک، فتوشاپ، بازی‌ها و...

انواع نرم‌افزار

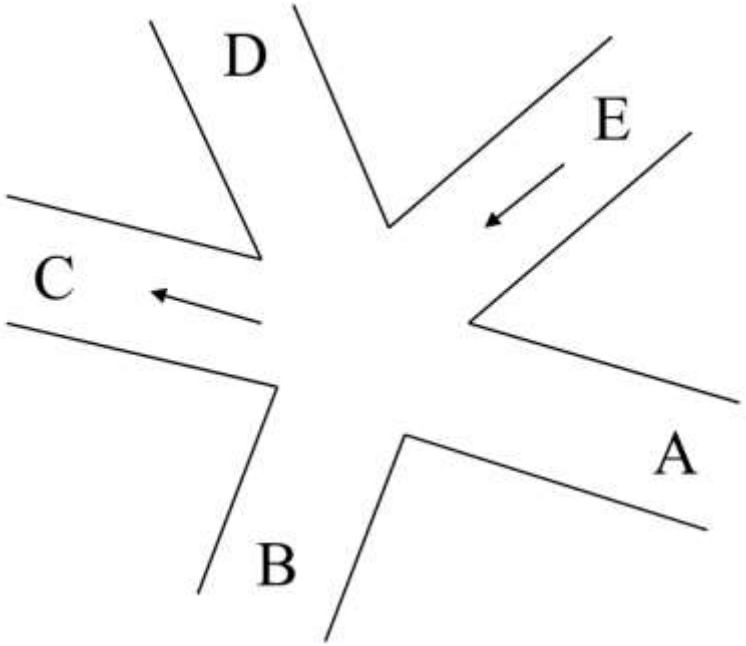
- نرم‌افزار سیستمی (System Software)
  - نرم‌افزارهایی که برای پنهان کردن پیچیدگی سخت‌افزار به کار گرفته می‌شود.
    - سیستم عامل، درایورهای سخت‌افزارها و...

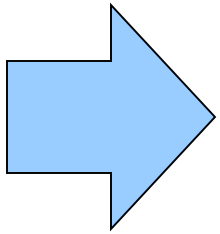
مراحل حل مسئله

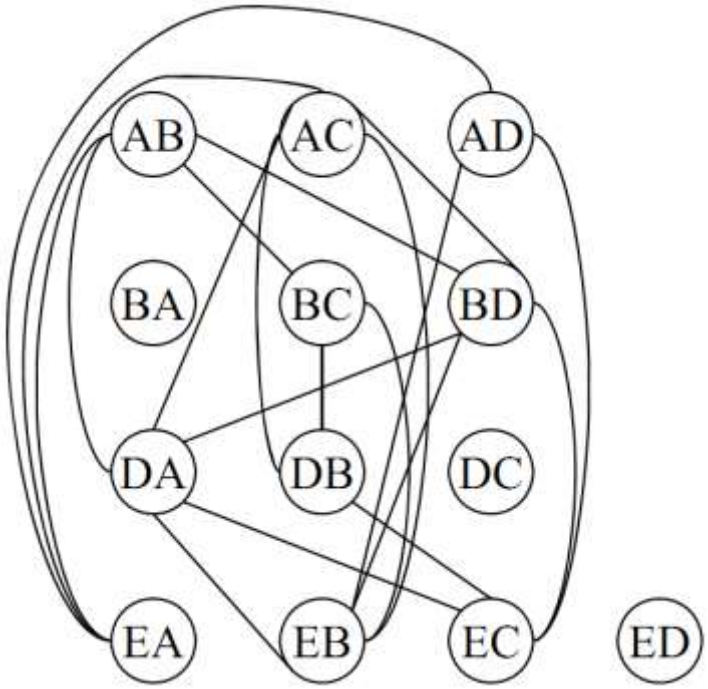


مراحل حل مسئله

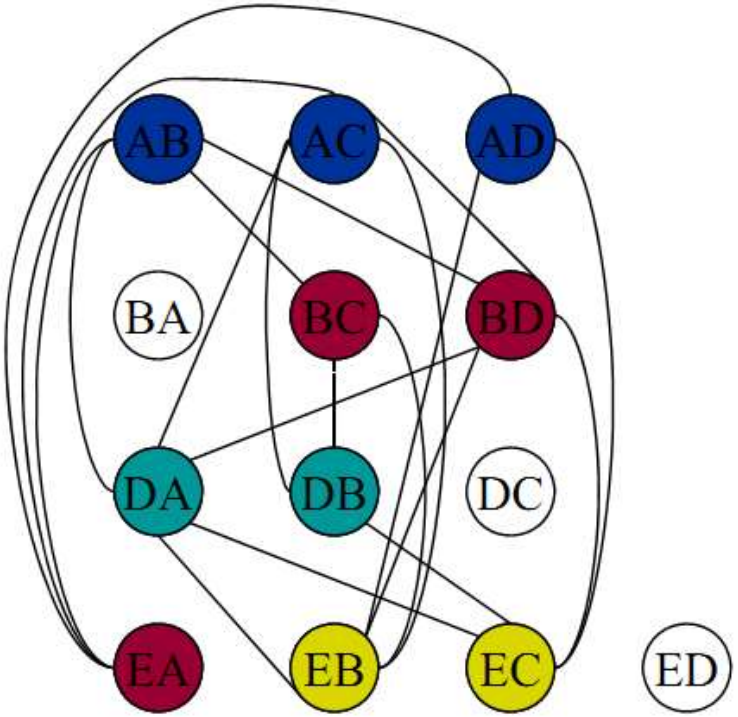
- هدف:
  - کمترین تعداد زمان‌های چراغ
  - عبور بیشترین ترافیک در هر زمان







مراحل حل مسئله



- **انتزاع:** در نظر نگرفتن جزئیات غیر ضروری
  - عرض خیابان‌ها
  - میزان ترافیک خیابان‌ها
- **مدل:**
  - مثلا AD به معنی گردش از A به D است
  - رسم یک یال بین گردش‌هایی که همزمان میسر نیست
- **الگوریتم:** این مسئله با الگوریتم رنگ آمیزی گراف‌ها قابل حل است.
  - رئوس مجاور نباید همرنگ باشند
- **ساختمان داده:**
  - چگونه گراف را پیاده‌سازی کرده آنرا رنگ آمیزی کنیم؟

[محمد قدسی – دانشگاه شریف]

یک توسعه دهنده نرم‌افزار با تجربه چه چیزهایی می‌داند؟

- مشخصات فنی برخی زبان‌های برنامه‌نویسی
- مهمتر: قواعد عمومی که در عمده زبان‌های برنامه‌نویسی مشترک است.
- مهارت‌ها:
  - طراحی
  - اشکال زدایی
  - آزمون
  - Code Review / Refactoring
  - مستندسازی
  - آشنایی با چارچوب‌های توسعه نرم افزار و ابزارهای مرتبط
  - مدیریت ابزارهای Source Control
  - آشنایی با DevOps و ابزارهای مرتبط

یادگیری در تمام عمر

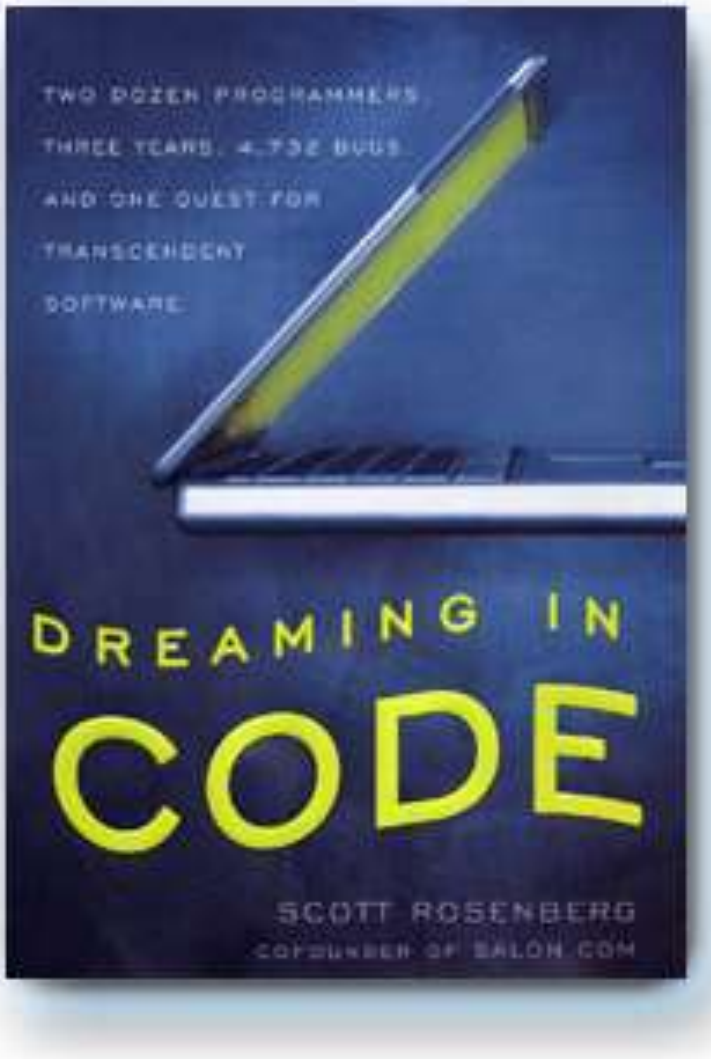
- آموزش و یادگیری این مهارت‌ها مشکل است.
  - باید با تجربه به دست بیاید
- توسعه دهنده نرم‌افزار خوب هیچگاه از یادگیری دست بر نمی‌دارند. بدون توجه به اینکه چه مدت است این کار را انجام می‌دهند.
  - یادگرفتن روش‌های جدید فکر کردن و نگرش به مسئله
- هیچ راه کوتاهتری برای رسیدن به این هدف در هیچ جایی وجود ندارد.
  - هرچه بیشتر تلاش کنید، بیشتر نتیجه می‌گیرید.

- اهمیت طراحی در توسعه نرم‌افزار
- اهمیت شی‌گرایی در طراحی و توسعه نرم‌افزار
- اصول شی‌گرایی تعریف شده برای نیل به طراحی مناسب
- ارتباط این اصول با برنامه‌نویسی شی‌گرا

بحران نرم افزار

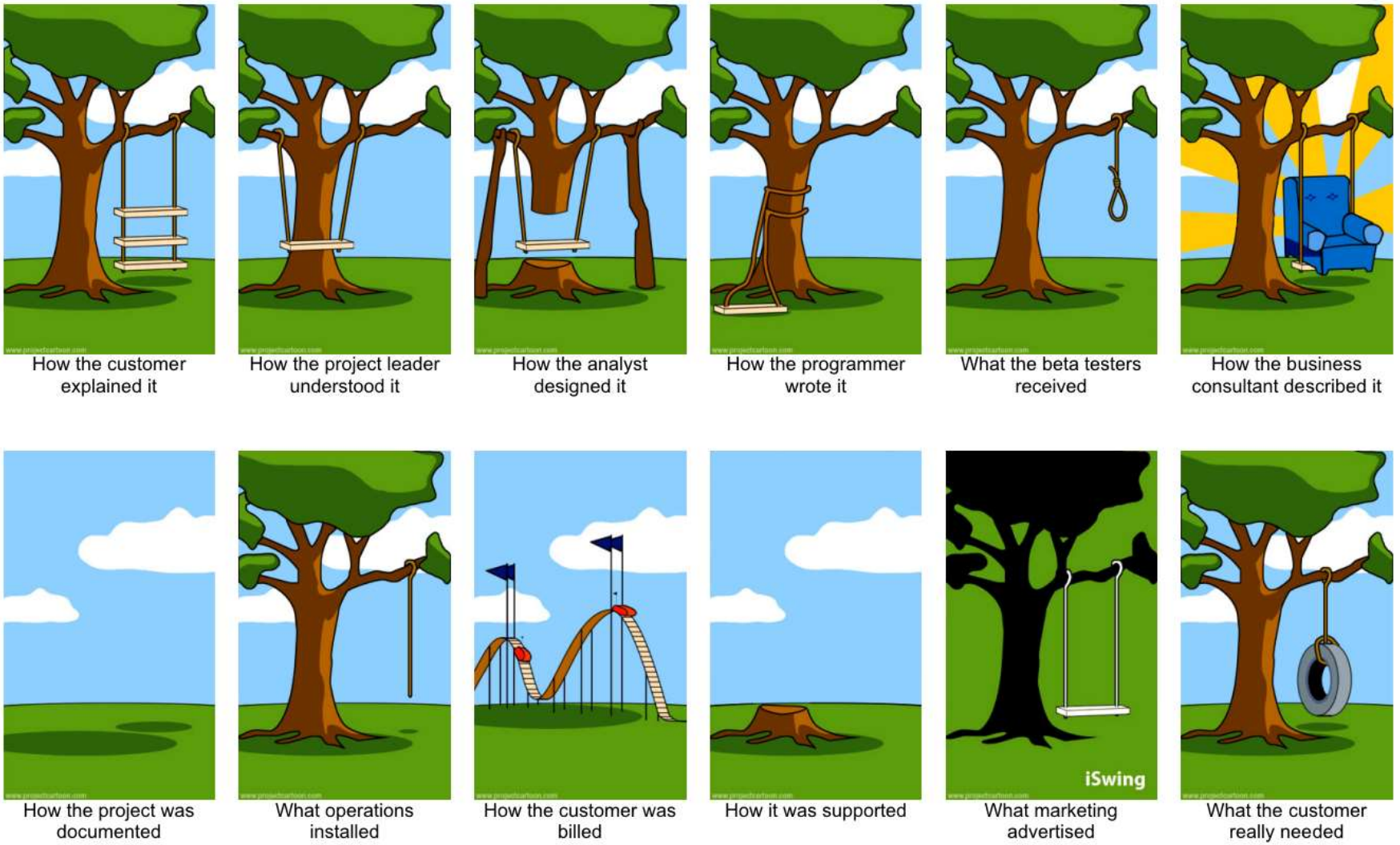
- هزینه‌های هنگفت تولید نرم افزار
- عدم تحویل به موقع
- عدم تامین نیازمندی‌های کاربر
- کیفیت پایین و نامطمئن
- سختی نگهداری بعلت کیفیت پایین طراحی

بحران نرم افزار



- دوجین برنامه نویس
- سه سال
- ۴۷۳۲ باگ
- تلاشی برای ایجاد نرم افزاری عالی!
  - پروژه چندلر (`chandlerproject.org`)

توسعه برنامه کاربردی در عمل!



چه زمانی باید فاتحه مهندسی نرم افزار را خواند؟

- آخ...! یادم رفت!
- اِ... اینجوری هم می‌شد!
- عجب...! پس این را هم می‌خواستید؟!
- ببخشید! به جا نمی‌آورم! شما...؟! (در مواجه با یکی از اشخاص کلیدی مشتری)
- ای تف به این روزگار...! توی پروژه قبلی هم همین اشتباه رو کرده بودیم!
- الان که ۱۰۰ درصد را تحویل گرفته‌اید، خواهش می‌کنم حداقل پیش پرداخت پروژه را بدهید!
- تا اونجا که یادم می‌آید، این تیکه کد را دو سه بار دیگه هم `Copy-Paste` کرده بودم!
- رو سیستم خودمون که درست کار می‌کنه!
- مشکل از شبکه است!
- قبلا این مساله را به ما گفته بودین؟!
- این موضوع در `RFP` ذکر نشده بود!
- ببخشید حجم ریالی قرارداد شما با این درخواست مطابق نیست!

- `[http://weblog.radmanitd.com/index.php/archives/2916]`

چه زمانی باید فاتحه مهندسی نرم افزار را خواند؟

این برنامه دیگه به درد نمی‌خوره

باید از اول بنویسیمش!

چرا نوشتن نرم افزارهای تجاری مشکل است؟

- پیچیدگی مساله
  - اگر موضوعات ساده باشند که برای حل آنها به نرم افزار فکر نمی‌کنیم!
  - نیازمندی‌های گوناگون و متضاد
  - مکانیزه کردن تعاملات انسانی

چرا نوشتن نرم افزارهای تجاری مشکل است؟

- تغییر رخ می‌دهد
  - تغییر ناشی از رفع ایرادات نرم افزار
  - تغییر به منظور بهبود عملکرد
  - افزودن امکانات جدید
  - تغییر نیازمندی‌ها
  - تغییر قواعد کسب و کار
  - تغییر قوانین حاکم بر فضای مسئله

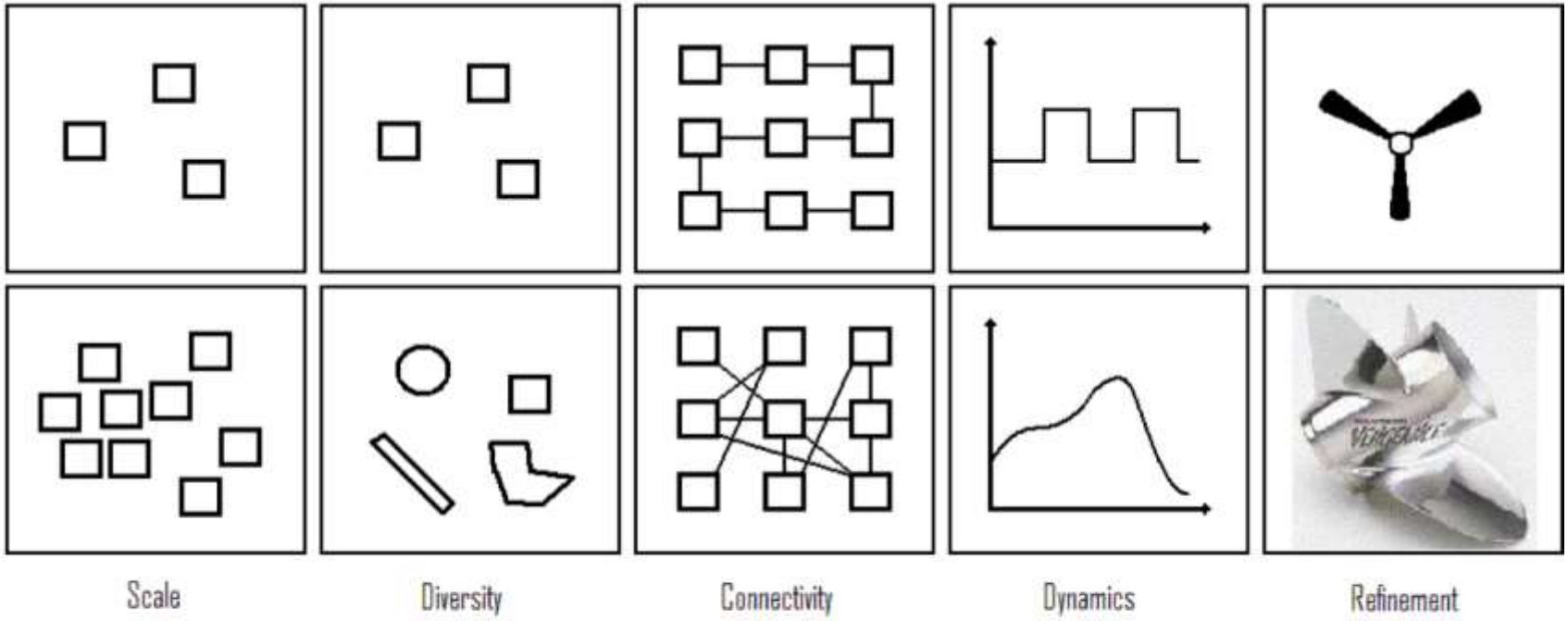
چرا نوشتن نرم افزارهای تجاری مشکل است؟

- انواع تغییر در نرم‌افزار
  - Corrective
  - Adaptive
  - Perfective
  - Preventive

چرا نوشتن نرم افزارهای تجاری مشکل است؟

- پیچیدگی فرآیند تولید
  - ماهیت مسائل ذاتا پیچیده است. ماهیت نرم افزار نیز همینطور!
  - ذهن انسان پیچیده است و پیچیدگی را دوست دارد!
  - استاندارد نبودن فرآیندهای مهندسی نرم افزار
    - فعالیت‌های مهندسی نرم افزار مستلزم طراحی‌های نوآورانه است و به شکل روتین قابل انجام نیست.
    - مهندسی نرم افزار هنوز در حال بکارگیری به روش غیر اصولی (non-systematic) است.
    - نسبت به سایر تخصص‌های مهندسی سازماندهی کمتری دارد
    - نسبت به برخی از رشته‌های مهندسی، استانداردهای کمتری دارد
    - بستن طراحی و کامل شدن آن تا انتهای پروژه به طول می‌انجامد
    - دستاورد اصلی آن مهیتی غیرقابل لمس دارد (محصول انتزاعی است و نه فیزیکی)

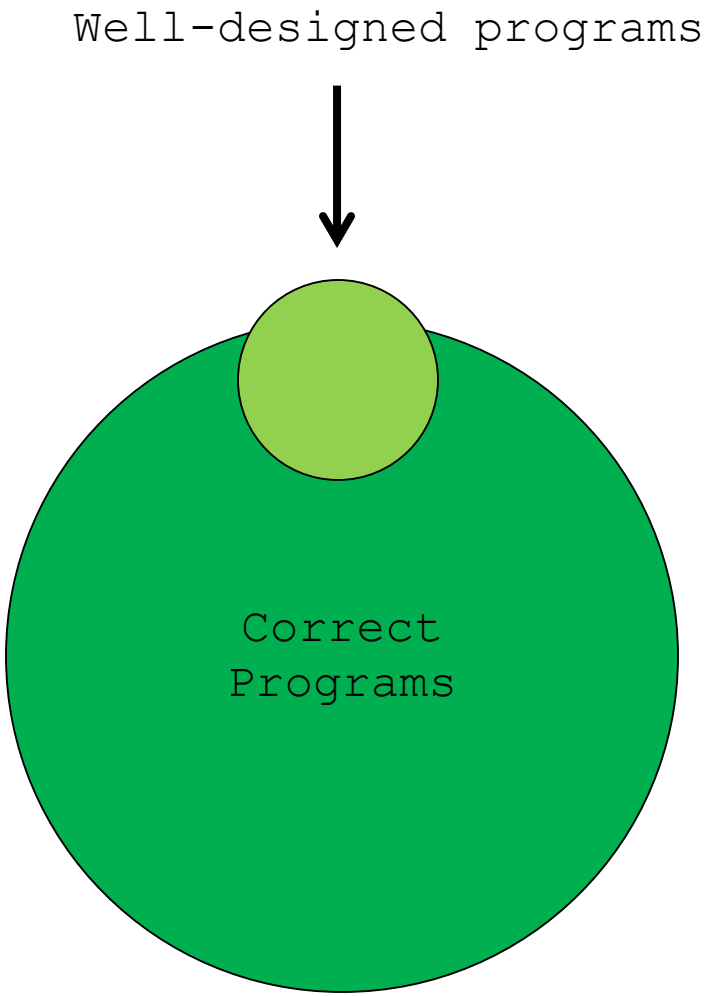
چرا نوشتن نرم افزارهای تجاری مشکل است؟



چرا نوشتن نرم افزارهای تجاری مشکل است؟

- نمی‌توان پیچیدگی را از بین برد بلکه باید آنرا کنترل کرد.
  - پیچیدگی نرم افزار با پیچیدگی سیستم‌های طبیعی و محصولات فیزیکی ساخت دست بشر متفاوت است و یک خاصیت ذاتی سیستم‌های نرم افزاری بزرگ می‌باشد.

اهمیت طراحی خوب در نرم‌افزار



- کدامیک؟
  1. یک برنامه که درست کار می‌کند؟
  2. یک برنامه که به خوبی طراحی شده است؟

- بینهایت برنامه وجود دارد که کارهای مورد انتظارشان را به خوبی انجام می‌دهند
- تعداد محدودی به خوبی طراحی شده‌اند

اهمیت طراحی خوب در نرم‌افزار

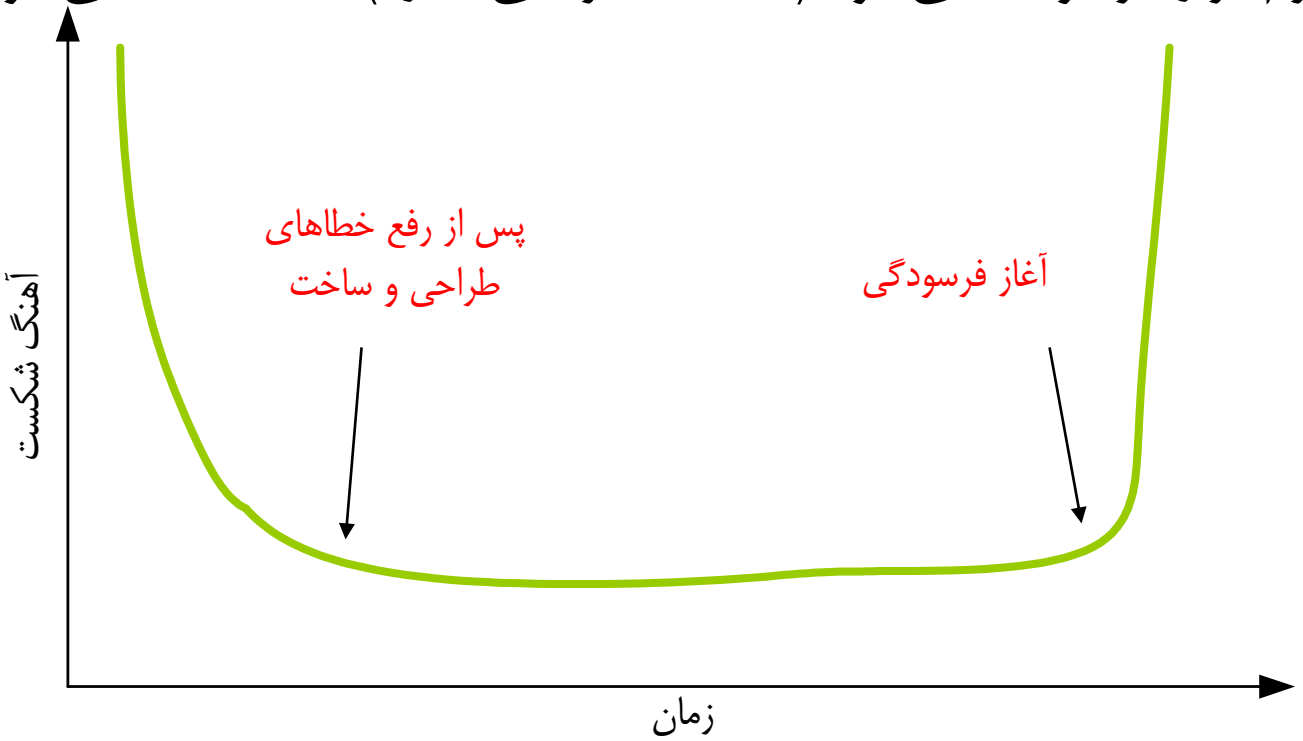
- با یک طراحی خوب
  - نوشتن یک برنامه با طراحی خوب راحت‌تر است.
  - توسعه، به‌روز رسانی و تغییر ساده‌تر است.

اهمیت طراحی خوب در نرم‌افزار

- جلوگیری از فاسد شدن سریع نرم افزار
- کنترل پیچیدگی

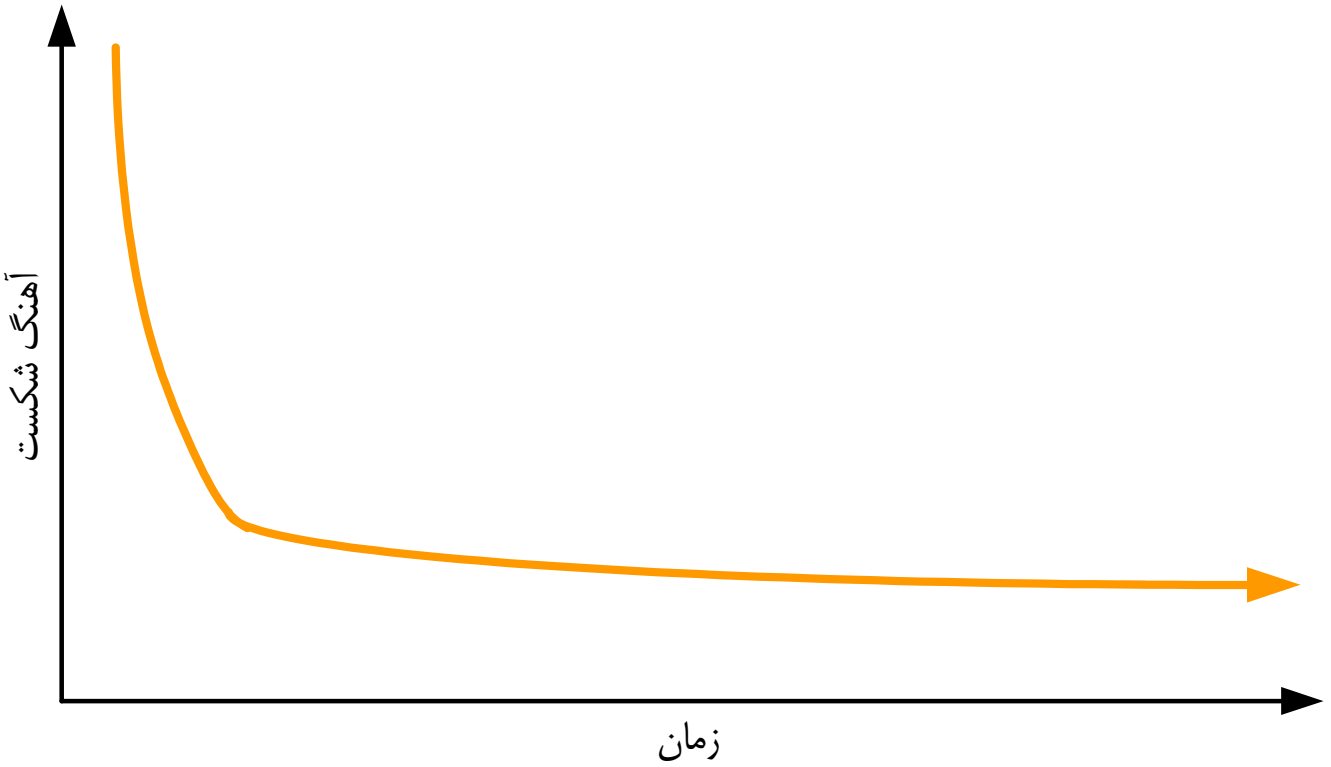
اهمیت طراحی خوب در نرم‌افزار

- در نرم‌افزار فقط یک چیز است که تغییر نمی‌کند: **تغییر**
- بنابراین نرم‌افزار فرسوده نمی‌شود (ماهیت فیزیکی ندارد) بلکه فاسد می‌شود.



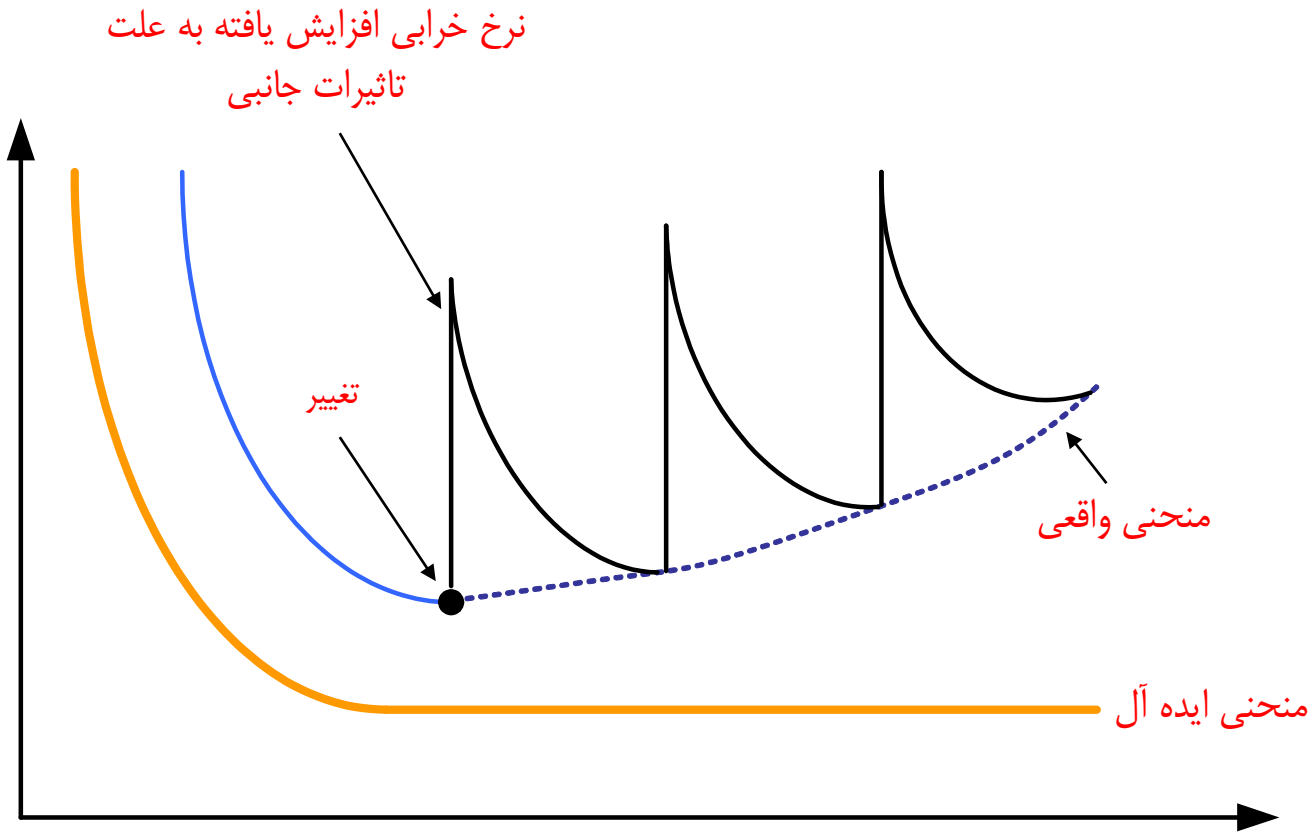
منحنی نرخ خرابی سخت افزار نسبت به زمان

اهمیت طراحی خوب در نرم‌افزار



منحنی ایده‌آل آهنگ شکست نرم افزار نسبت به زمان

اهمیت طراحی خوب در نرم‌افزار



منحنی نرخ خرابی واقعی نرم افزار نسبت به زمان

- Simplicity is prerequisite for reliability.
- Simplicity is a great virtue but it requires hard work to achieve it and education to appreciate it. And to make matters worse: **complexity sells better.**

Edsger W. Dijkstra

First Law of Software Quality

$$e = mc^2$$

$$errors = (more\ code)^2$$

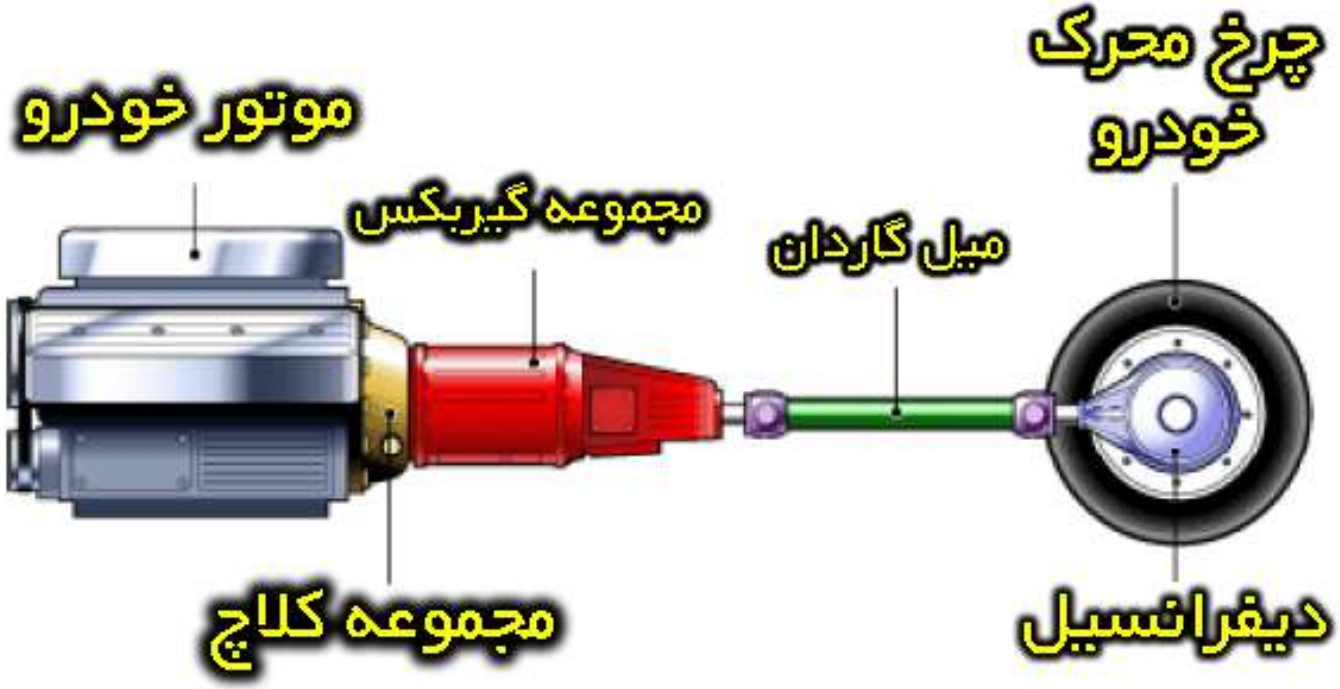
Bring Order to Chaos

- یک اتومبیل شامل بخش‌های متفاوتی است که می‌تواند به شکل مجزا اما در ارتباط با هم مورد بررسی قرار گیرند:
  - سیستم انتقال قدرت خودرو
  - سیستم سوخت رسانی خودرو
  - سیستم برق خودرو
  - و...

Bring Order to Chaos

- سیستم انتقال قدرت خودرو
  - مکانیزم‌های انتقال قدرت وظیفه دارند قدرت و گشتاوری که در موتور تولید شده را به چرخ‌ها منتقل کنند.
  - خروجی موتور خودرو یک شفت می‌باشد. برای اینکه انرژی این شفت به چرخ‌های خودرو منتقل شود، نیاز به سیستم انتقال قدرت می‌باشد. در خودروها این شفت خروجی از موتور وارد مجموعه‌ای به اسم کلاچ می‌شود.
  - کلاچ وظیفه قطع و وصل کردن ارتباط بین گیربکس و موتور را بر عهده دارد.
  - گیربکس نیرو را از موتور دریافت کرده و آن را بصورت افزایش قدرت (دنده سنگین)، یا افزایش سرعت (دنده سبک) به دیفرانسیل و در نتیجه چرخ ها منتقل می‌کند.
  - وظیفه دیفرانسیل تقسیم نیرو بین چرخ‌ها است. چرخ‌های اتومبیل با سرعت‌های متفاوتی می‌چرخند. برای مثال، هنگامی که سر پیچ با اتومبیل در حال گردش هستید، چرخ داخلی نسبت به چرخ خارجی مسافت کمتری را طی می‌کند.

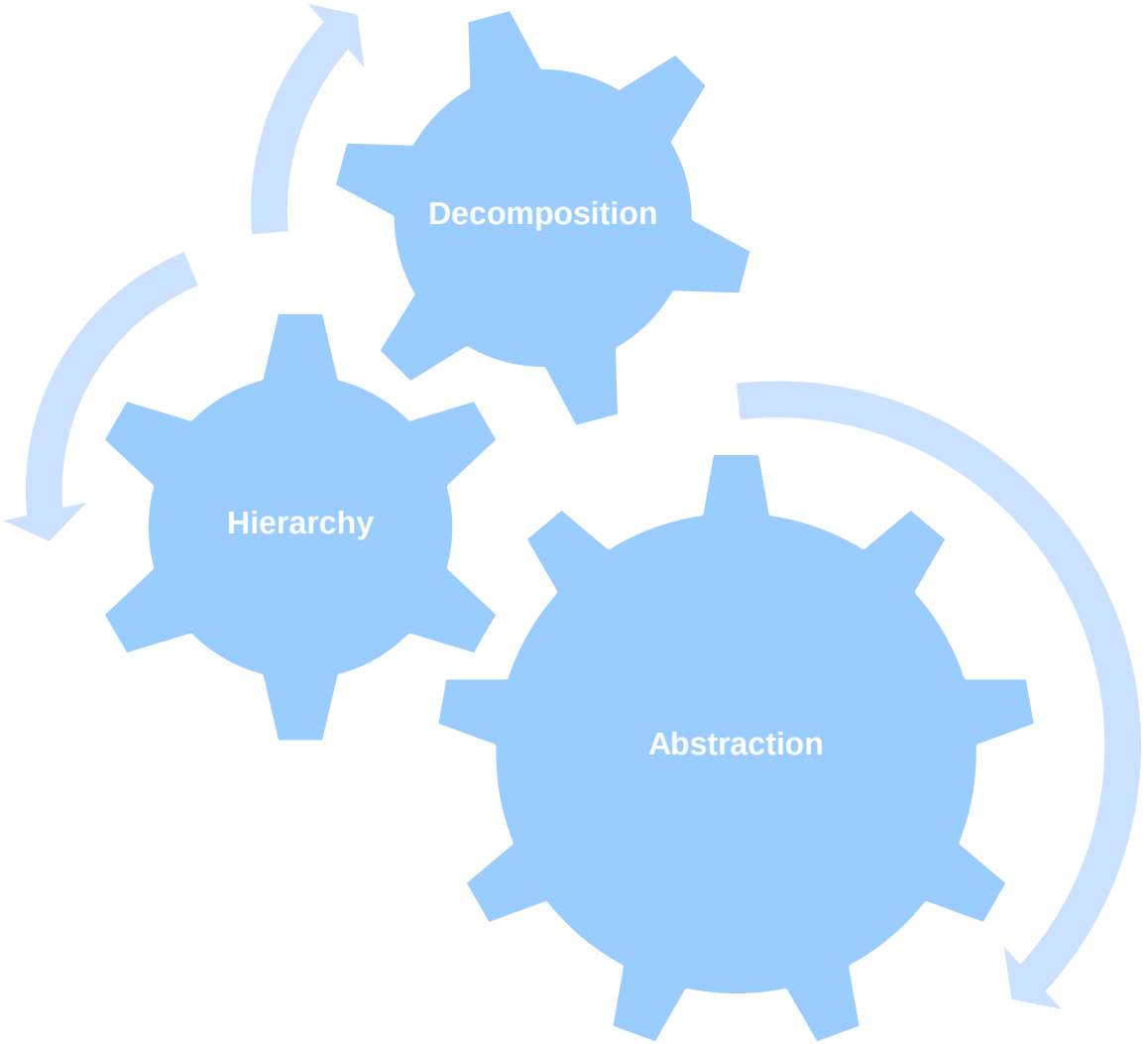
Bring Order to Chaos



Bring Order to Chaos

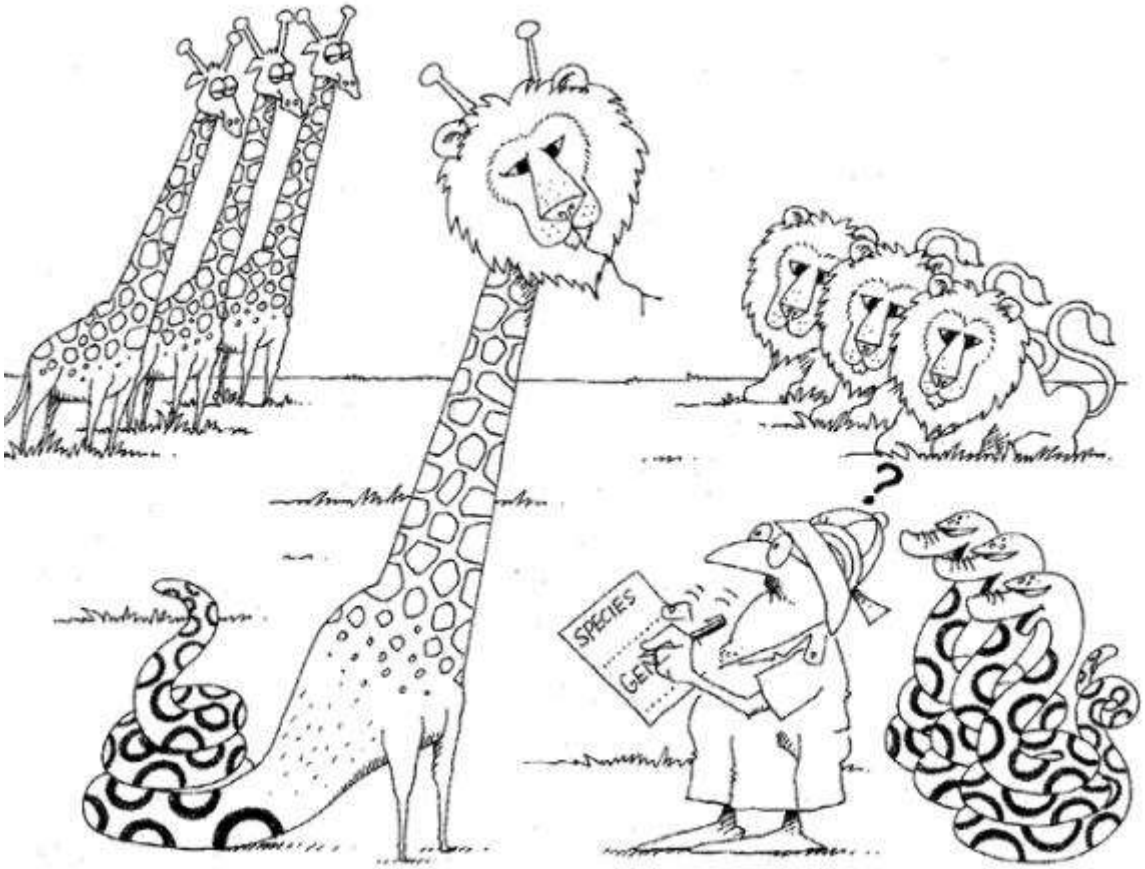
- سیستم انتقال قدرت خودرو
  - هر بخش (موتور، کلاچ، گیربکس، دیفرانسیل و...) به طور مستقل از سایر اجزا قابل طراحی و پیاده‌سازی است.
  - هر بخش می‌تواند از طریق اینترفیس تعریف شده با سایر بخش‌ها ارتباط برقرار کند اما ساختار داخلی سایر بخش‌ها برای او مشخص نیست (black-box).
  - در صورتی که بخواهیم موتور ماشین را عوض کنیم، با فرض عدم تفاوت در اینترفیس‌های موتور جدید، این کار به سادگی و بدون ایجاد تغییر در سایر اجزا قابل انجام است.
    - تغییر در یک بخش منجر به تغییر در سایل بخش‌ها نمی‌شود.

Bring Order to Chaos



Bring Order to Chaos

- Classification



چرا طراحی شی‌گرا؟

- در اغلب سیستم‌های پیچیده، پیچیدگی به صورت سلسله مراتب (Hierarchy) ظاهر می‌شود.
  - سلسله مراتب‌های Is-A یا Has-A
  - در سیستمی که از چند زیرسیستم تشکیل می‌گردد، ارتباط بین اجزای درونی هر زیر سیستم (Intra-component Linkage) قوی‌تر از ارتباط بین زیرسیستم‌ها (Inter-components Linkage) است.
- سیستم‌های سلسله مراتبی معمولا از تعداد کمی از زیر سیستم‌های مشخص و متفاوت تشکیل می‌شوند که این زیرسیستم‌ها به صورت‌های گوناگون و ترتیب‌های مختلف ظاهر می‌شوند.
- معمولا سیستم‌های پیچیده که به صورت محکم و استوار عمل می‌کنند حاصل تکامل سیستم‌های ساده‌ای هستند که به درستی عمل می‌کردند.

چرا طراحی شی‌گرا؟

- ماهیت شی‌گرایی به نحوی است که می‌تواند با توجه به ویژگی‌های سیستم‌های پیچیده، آنرا کنترل کند.
  - این ماهیت دلیل اصلی انتخاب رویکرد شی‌گرا است.
  - برخی به اشتباه فکر می‌کنند چون انسان با اشیا سروکار دارد و طراحی با این مفاهیم آشنا برایش ساده‌تر است پارادایم شی‌گرا شکل گرفته است! **اما این همه‌ی داستان نیست!**
    - سلسله مراتب‌هایی که در سیستم‌های نرم افزاری با آن مواجه می‌شویم و می‌توانیم از آن استفاده کنیم قابل مقایسه با آنچه در اشیا می‌بینیم نیست!

چرا طراحی شی‌گرا؟

- طراحی شی‌گرا، طراحی خوب را تشویق می‌کند.
  - اگر تمام اصول و قواعد رعایت شود و فقط با طراحی کلاس خود را فریب ندهیم، طراحی مناسبی خواهیم داشت.
- طراحی شی‌گرا تنها راه رسیدن به یک طراحی خوب نیست.
  - اما کماکان از روش‌های شناخته شده‌ی دیگر بهتر است.

چرا طراحی شی‌گرا؟

- اشیا انتزاعی از موجودیت‌های واقعی سیستم هستند.
- اشیا مستقل از هم هستند.
  - می‌توانند به صورت مجرد طراحی و پیاده‌سازی شوند.
  - به این شکل می‌توان به شکاندن پیچیدگی مسئله کمک کرد.
- اطلاعات اشیا محصور شده است.
  - تغییرات از بیرون را نمی‌پذیرد.
  - با دریافت پیغام‌ها خودش وضعیت خودش را تغییر می‌دهد.

چرا طراحی شی‌گرا؟

- به دلیل ویژگی‌های ذکر شده در اسلاید قبل، در صورت طراحی درست، تغییر در یک بخش از نرم‌افزار منجر به تغییر در سایر بخش‌ها نخواهد شد.
  - نگهداری ساده‌تر
  - تغییر کم هزینه‌تر
  - بالا رفتن طول عمر نرم‌افزار

چرا طراحی شی‌گرا؟

- قابلیت استفاده مجدد از کد

  - همانطور که ذکر شد، سیستم‌های پیچیده از زیر سیستم‌های مشخص و متفاوت تشکیل شده که این زیر سیستم‌ها به صورت‌های گوناگون در نقاط مختلف ظاهر می‌شوند.
    - در بحث برنامه‌نویسی شی‌گرا تحت عنوان چند ریختی آنرا خواهیم دید.
  - ضمن آنکه ایده‌آل ما این است که بتوانیم همانند سخت افزار با مونتاژ کردن قطعه کدهای مختلف کنار هم (اینجا اشیا) برنامه‌های جدید بسازیم

چرا طراحی شی گرا؟

*OO is about **not** writing code*

The more you can reuse code,
the less you have to write

چرا طراحی شی‌گرا؟

- کدام کدها می‌تواند مورد استفاده مجدد قرار گیرد؟
  - کتابخانه کلاس‌ها: مثلا جاوا دارای هزاران از آن است.
  - کدهایی که توسط دیگران نوشته شده است.
  - کدهای خودتان

برقراری موازنه در طراحی

- در طراحی نرم‌افزار بین چند هدف باید توازن برقرار ساخت. می‌خواهیم سیستمی طراحی کنیم که:
  - ساده باشد
  - پیاده سازی آن آسان باشد
  - نگهداری و توسعه آن آسان باشد
  - اجرای آن سریع باشد
  - به حافظه کمی نیاز داشته باشد
  - درک آن برای سایر برنامه‌نویسان ساده باشد
  - استفاده از آن ساده باشد
  - به درستی کار کند
  - قابل اجرا روی پلت فرم‌های دیگر باشد
  - تمام امکانات مورد نظر همه کاربران را داشته باشد
  - ...