Ch.17 WebApp Design (Cont.)

May 10, 2015

Interface Design Principles - III

- *Maintain work product integrity*—A work product (e.g., a form completed by the user, a user specified list) must be automatically saved so that it will not be lost if an error occurs.
- *Readability*—All information presented through the interface should be readable by young and old.
- *Track state*—When appropriate, the state of the user interaction should be tracked and stored so that a user can logoff and return later to pick up where she left off.
- *Visible navigation*—A well-designed WebApp interface provides "the illusion that users are in the same place, with the work brought to them."

- **Aesthetic Design**
  - Don’t be afraid of white space.
  - Emphasize content.
  - Organize layout elements from top-left to bottom right.
  - Group navigation, content, and function geographically within the page.
  - Don’t extend your real estate with the scrolling bar.
  - Consider resolution and browser window size when designing layout.

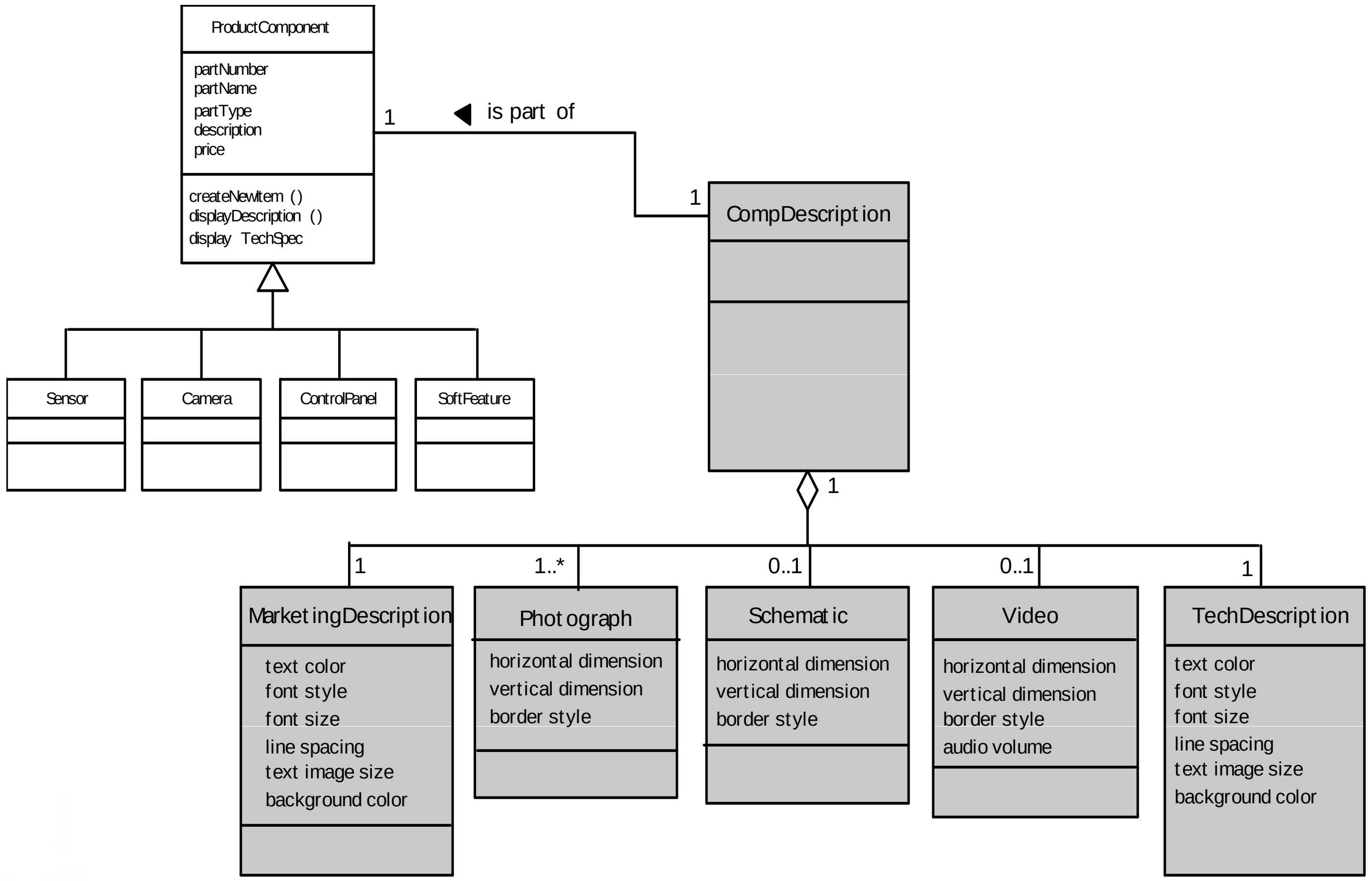
- **Content Design**

  - Develops a design representation for content objects
    - For WebApps, a content object is more closely aligned with a data object for conventional software

  - Represents the mechanisms required to instantiate(例示) their relationships to one another.

  - A content object has **attributes** that include **content-specific information** and **implementation-specific attributes** that are specified as part of **design**

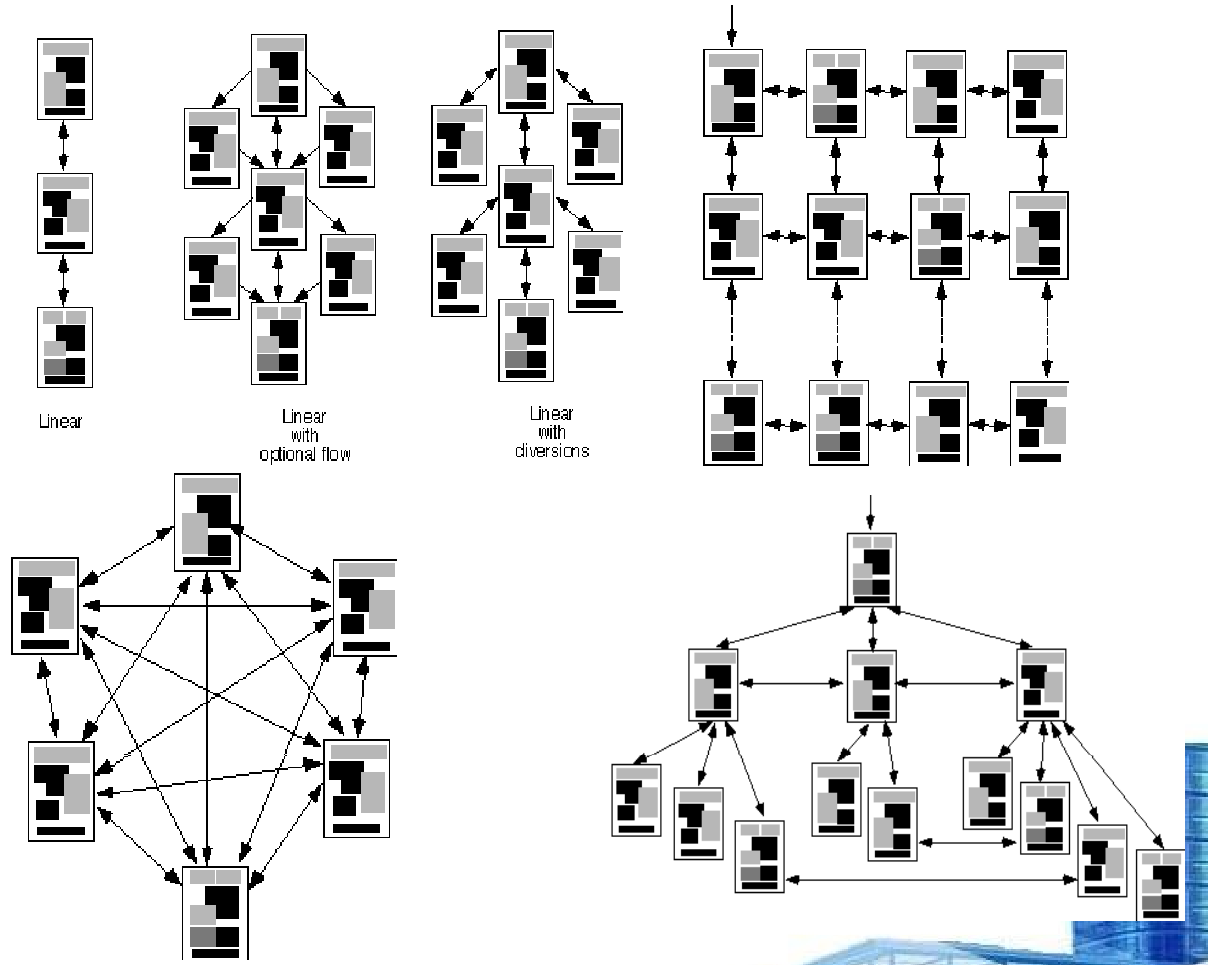
• Design of Content Objects



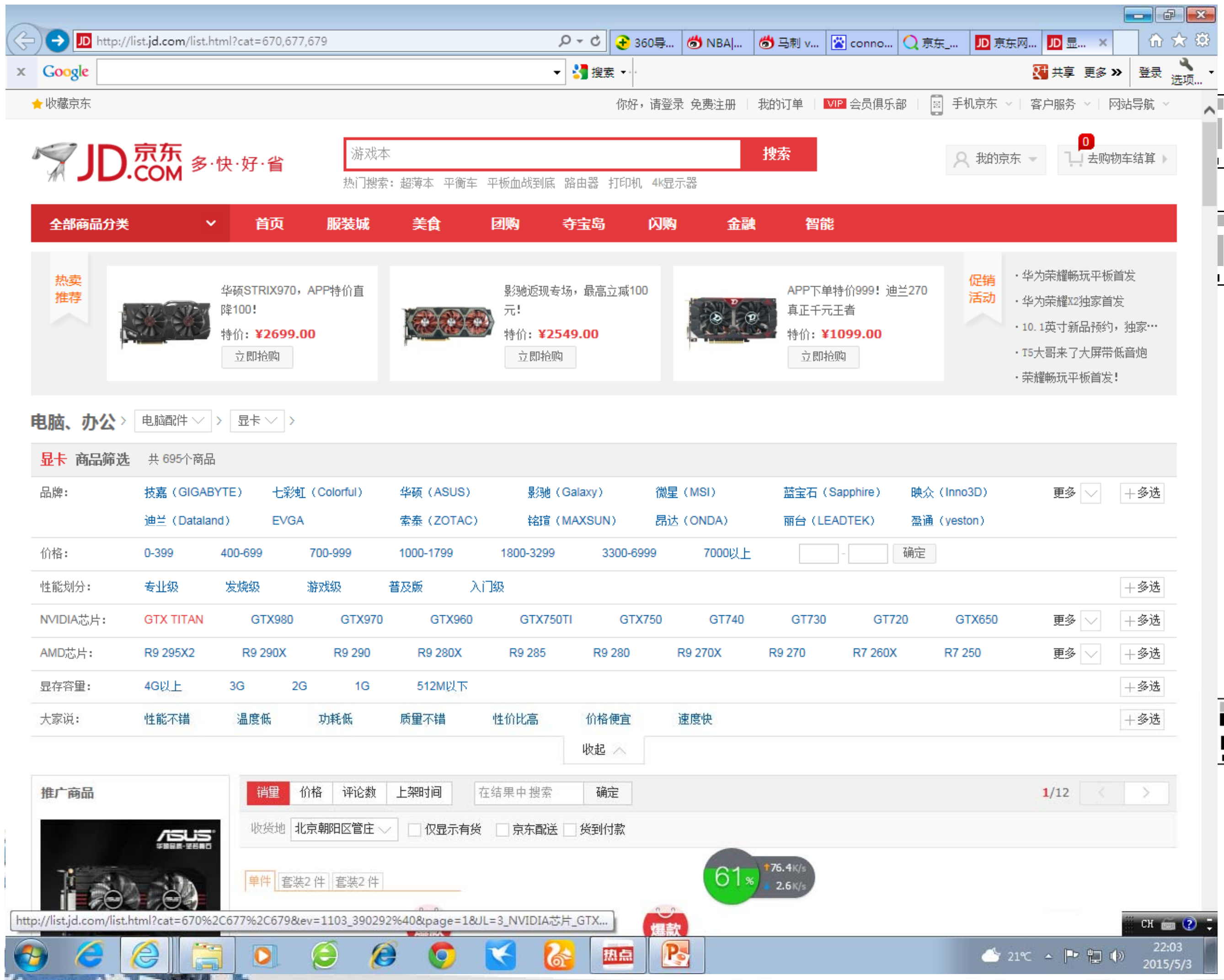
Architecture Design

- *Content architecture* focuses on the manner in which content objects (or composite objects such as Web pages) are structured for presentation and navigation.
  - The term information architecture is also used to connote (暗示) structures that lead to better organization, labeling, navigation, and searching of content objects.
- *WebApp architecture* addresses the manner in which the application is structured to manage user interaction, handle internal processing tasks, effect navigation, and present content.
- *Architecture design* is conducted in parallel with interface design, aesthetic design and content design.

• Content Architecture



- **Content Architecture**



http://list.jd.com/list.html?cat=670,677,679

360导... NBA|... 马刺 v... conno... 京东_... 京东网... 显...

Google 搜索 共享 更多 » 登录 选项...

收藏京东 你好，请登录 免费注册 我的订单 VIP 会员俱乐部 手机京东 客户服务 网站导航

JD.COM 京东 多·快·好·省

游戏本 搜索

热门搜索：超薄本 平衡车 平板血战到底 路由器 打印机 4k显示器

我的京东 0 去购物车结算

全部商品分类 首页 服装城 美食 团购 夺宝岛 闪购 金融 智能

热卖推荐

华硕STRIX970，APP特价直降100！ 特价：¥2699.00 立即抢购

影驰返现专场，最高立减100元！ 特价：¥2549.00 立即抢购

APP下单特价999！迪兰270真正千元王者 特价：¥1099.00 立即抢购

促销活动

- 华为荣耀畅玩平板首发
- 华为荣耀X2独家首发
- 10. 1英寸新品预约，独家…
- T5大哥来了大屏带低音炮
- 荣耀畅玩平板首发！

电脑、办公 > 电脑配件 > 显卡 >

显卡 商品筛选 共 695个商品

品牌：	技嘉（GIGABYTE）	七彩虹（Colorful）	华硕（ASUS）	影驰（Galaxy）	微星（MSI）	蓝宝石（Sapphire）	映众（Inno3D）			更多	+多选
	迪兰（Dataland）	EVGA	索泰（ZOTAC）	铭瑄（MAXSUN）	昂达（ONDA）	丽台（LEADTEK）	盈通（yeston）				
价格：	0-399	400-699	700-999	1000-1799	1800-3299	3300-6999	7000以上	- 确定			
性能划分：	专业级	发烧级	游戏级	普及版	入门级						+多选
NVIDIA芯片：	GTX TITAN	GTX980	GTX970	GTX960	GTX750TI	GTX750	GT740	GT730	GT720	GTX650 更多	+多选
AMD芯片：	R9 295X2	R9 290X	R9 290	R9 280X	R9 285	R9 280	R9 270X	R9 270	R7 260X	R7 250 更多	+多选
显存容量：	4G以上	3G	2G	1G	512M以下						+多选
大家说：	性能不错	温度低	功耗低	质量不错	性价比高	价格便宜	速度快				+多选

收起

推广商品

销量 价格 评论数 上架时间 在结果中搜索 确定 1/12

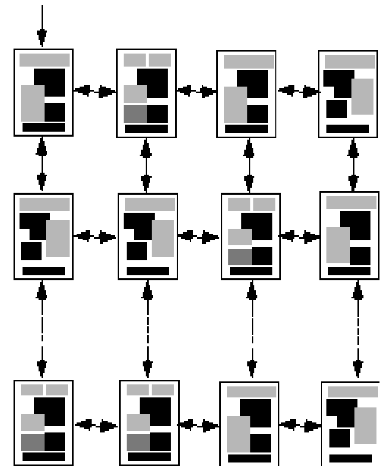
收货地 北京朝阳区管庄 仅显示有货 京东配送 货到付款

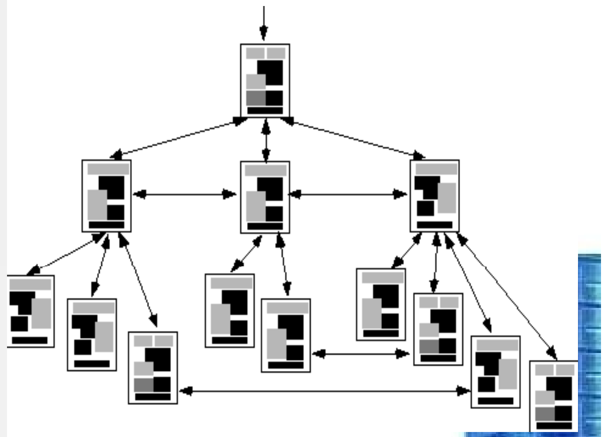
单件 套装2 件 套装2 件

61% 76.4K/s 2.6K/s

http://list.jd.com/list.html?cat=670%2C677%2C679&ev=1103_390292%40&page=1&JL=3_NVIDIA芯片_GTX...

爆款 热点 CH 21℃ 22:03 2015/5/3

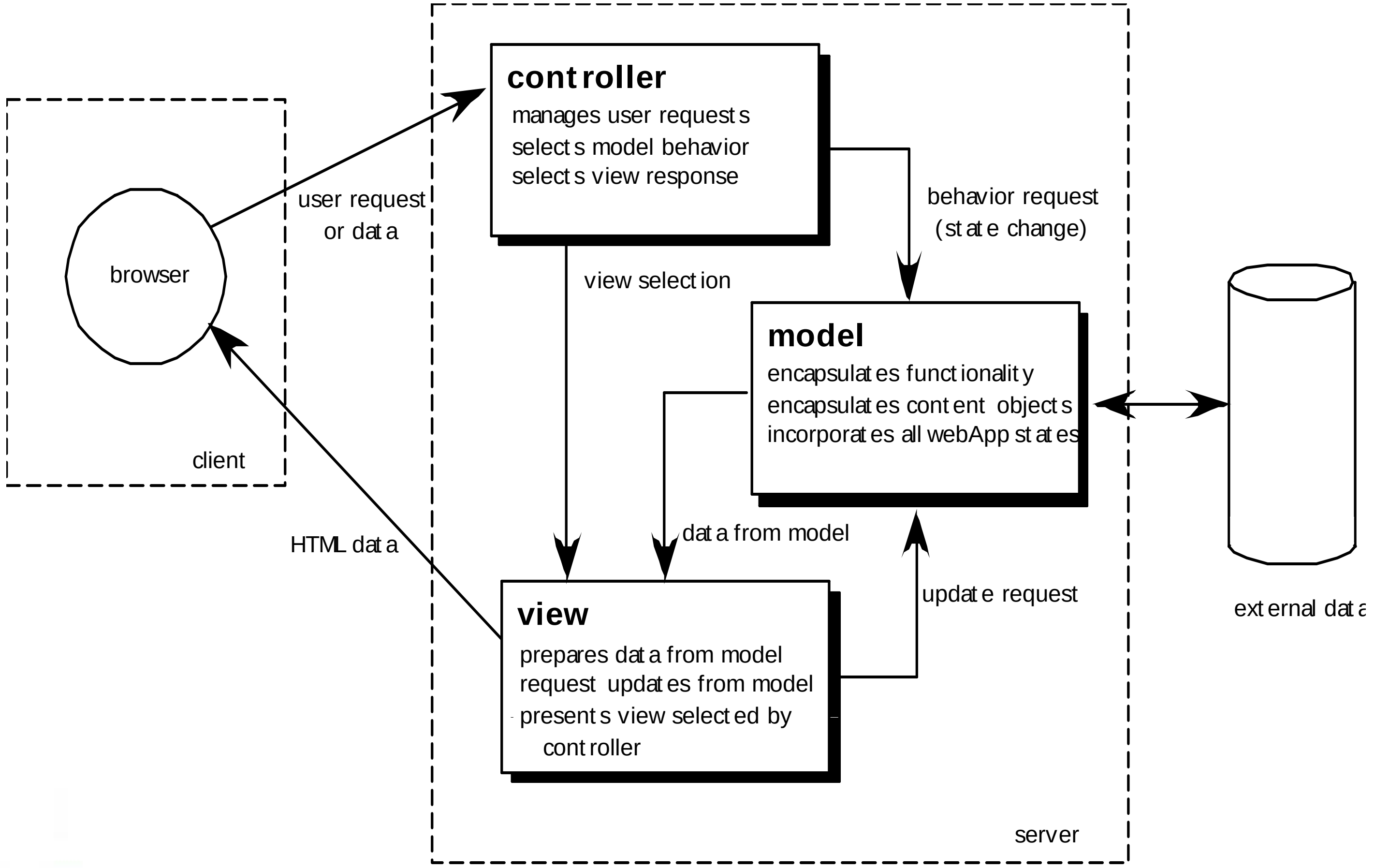




• MVC Architecture

- The *model* contains all application specific content and processing logic, including
  - all content objects
  - access to external data/information sources,
  - all processing functionality that are application specific
- The *view* contains all interface specific functions and enables
  - the presentation of content and processing logic
  - access to external data/information sources,
  - all processing functionality required by the end-user.
- The *controller* manages access to the model and the view and coordinates the flow of data between them.

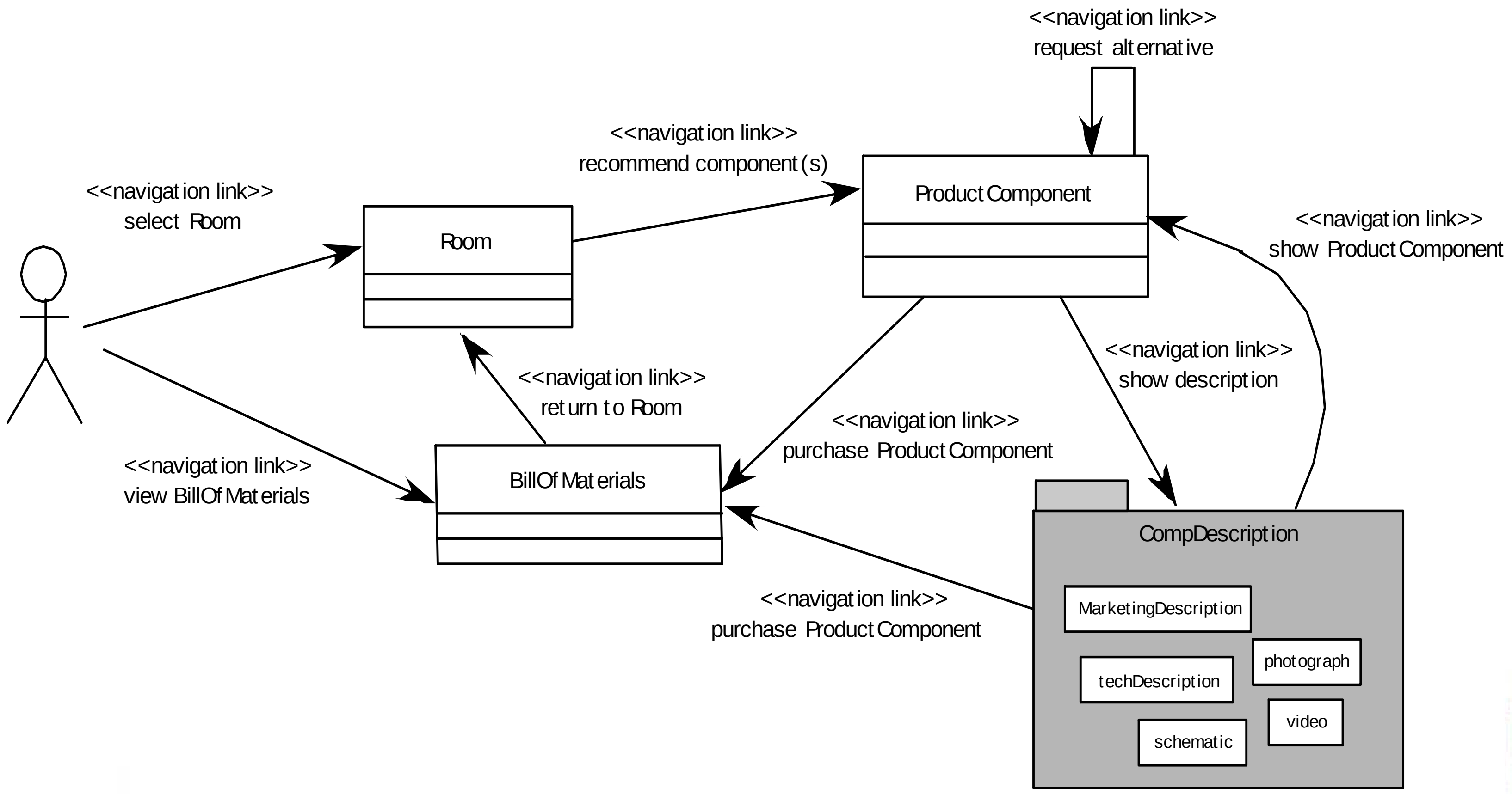
• MVC Architecture



Navigation Design

- Begins with a consideration of the user hierarchy and related use-cases
  - Each actor may use the WebApp somewhat differently and therefore have different navigation requirements
- As each user interacts with the WebApp, she encounters a series of *navigation semantic units* (NSUs)
  - NSU—"a set of information and related navigation structures that collaborate in the fulfillment of a subset of related user requirements"

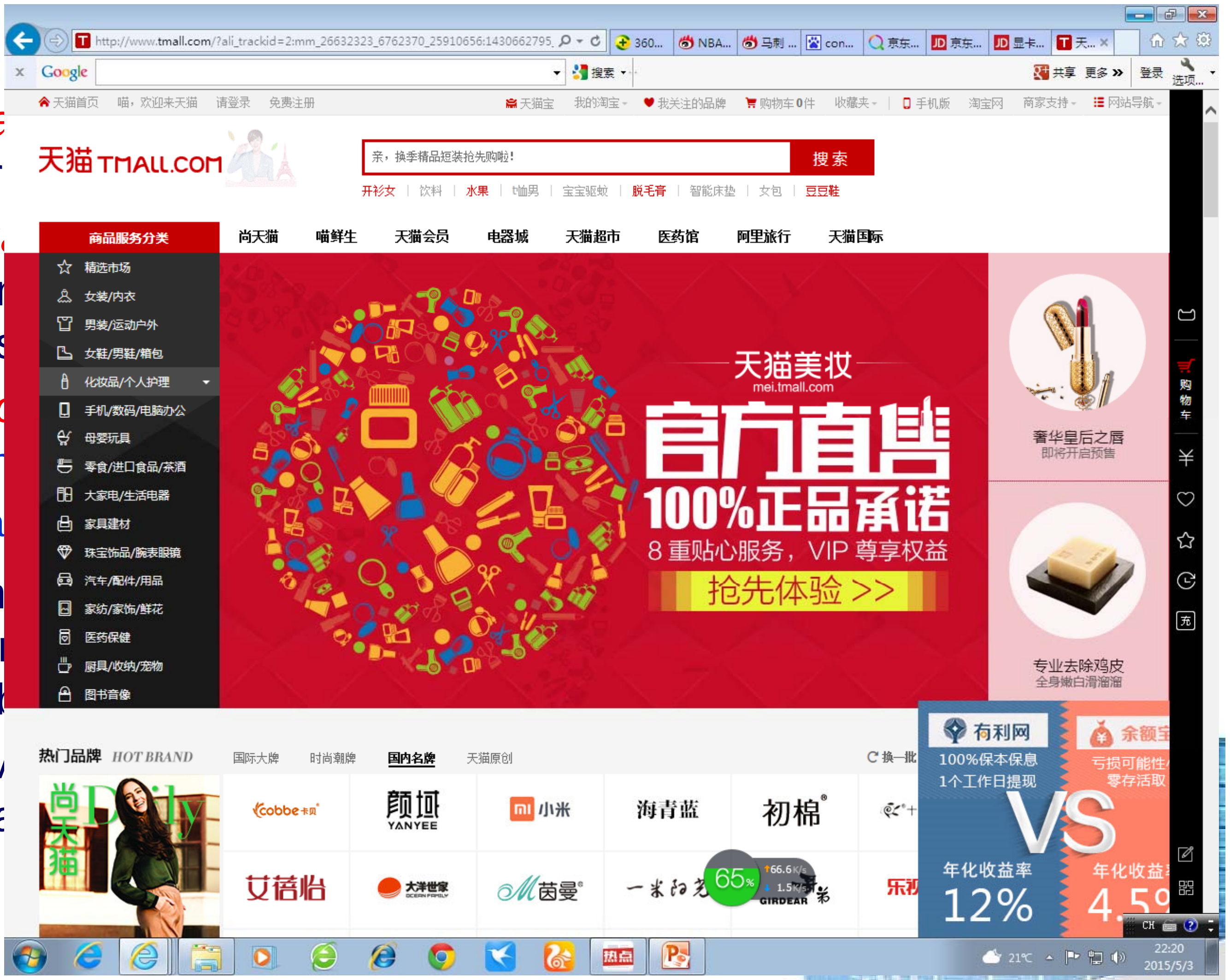
- **Creating an NSU**



Navigation Syntax

- *Individual navigation link*—text-based links, icons, buttons and switches, and graphical metaphors(隐喻)..
- *Horizontal navigation bar*—lists major content or functional categories in a bar containing appropriate links. In general, between 4 and 7 categories are listed.
- *Vertical navigation column*
  - lists major content or functional categories
  - lists virtually all major content objects within the WebApp.
- *Tabs*—a metaphor that is nothing more than a variation of the navigation bar or column, representing content or functional categories as tab sheets that are selected when a link is required.
- *Site maps*—provide an all-inclusive tab of contents for navigation to all content objects and functionality contained within the WebApp.

- **Navigation Syntax**
  - *Individual naviga* switches, and gr
  - *Horizontal navig* in a bar containi categories are lis
  - *Vertical navigatio*
    - lists major con
    - lists virtually a
  - *Tabs*—a metaph navigation bar o categories as tab
  - *Site maps*—prov content objects a



14

Component-Level Design

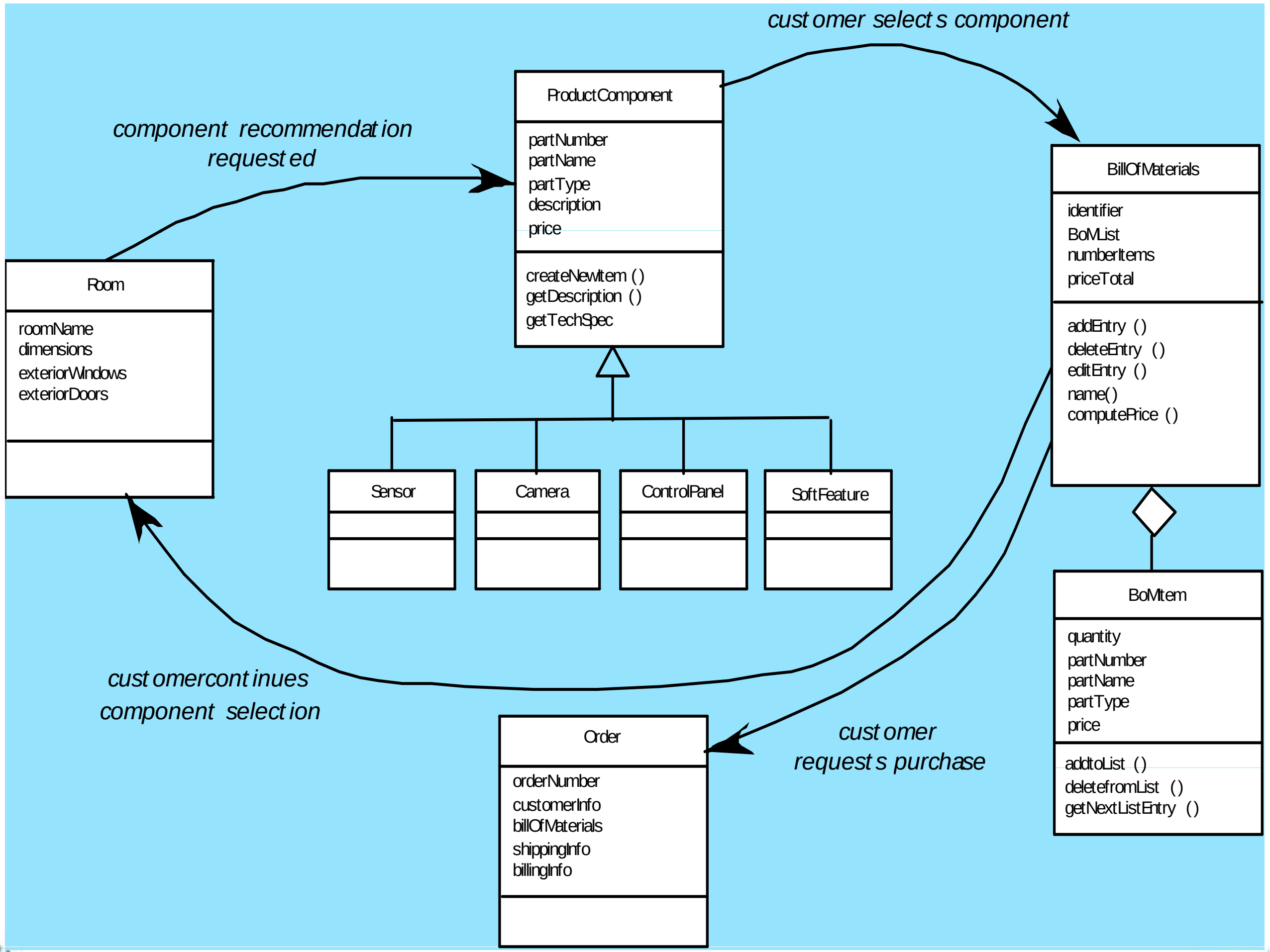
- WebApp components implement the following functionality
  - perform localized processing to generate content and navigation capability in a dynamic fashion
  - provide computation or data processing capability that are appropriate for the WebApp's business domain
  - provide sophisticated database query and access
  - establish data interfaces with external corporate systems.

OOHDM

- Object-Oriented Hypermedia Design Method (OOHDM)

	conceptual design	navigational design	abstract interface design	implementation
work products	Classes, sub-systems, relationships, attributes	Nodes, links, access structures, navigational contexts, navigational transformations	Abstract interface objects, responses to external events, transformations	executable WebApp
design mechanisms	Classification, composition, aggregation, generalization specialization	Mapping between conceptual and navigation objects	Mapping between navigation and perceptible objects	Resource provided by target environment
design concerns	Modeling semantics of the application domain	Takes into account user profile and task. Emphasis on cognitive aspects.	Modeling perceptible objects, implementing chosen metaphors. Describe interface for navigational objects	Correctness; Application performance; completeness

- **Conceptual Schema(概要， 图表)**



Ch.18 MobileApp Design

May 11, 2015

18.1 Mobile Development Considerations – 1

- Multiple hardware and software platforms
- Many development frameworks and programming languages.
- Many app stores with differing acceptance rules and tool requirements
- Short development cycles
- User interface limitations

18.1 Mobile Development Considerations – 2

- Complex camera/sensor interaction
- Effective use of context
- Power management
- Security and privacy models/policies
- Device limitations (computation and storage)
- Integration of external services
- Texting complexities

18.2 MobileApp Development Process Model

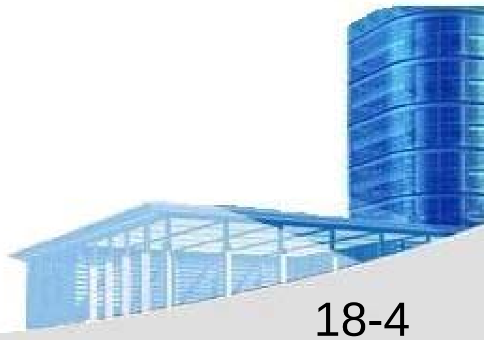
- Formulation
- Planning
- Analysis
- Engineering
- Implementation and testing
- User evaluation

18.3 MobileApp Quality Checklist - 1

- Can content and/or function and/or navigation options be tailored (使适应)to the user's preferences?（Ex. 春雨医生）
- Can content and/or functionality be customized to the bandwidth at which the user communicates? Does the app account for weak or lost signal in an acceptable manner?
- Can content and/or function and/or navigation options be made context aware according to the user's preferences?
- Has adequate consideration been given to the power availability on the target device(s)?
- Have graphics, media (audio, video), and other web or cloud services been used appropriately? (Ex. 3D Printing)

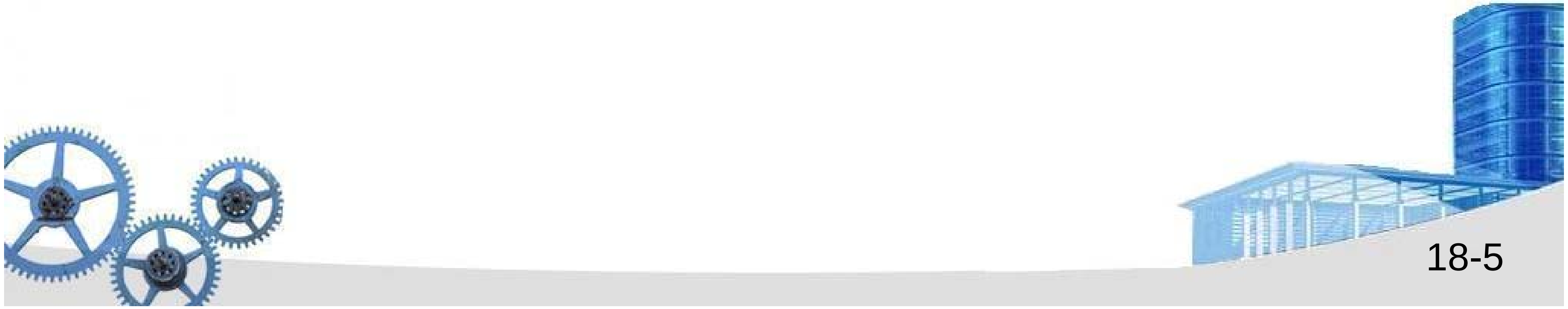
18.3 MobileApp Quality Checklist - 2

- Is the overall page design easy to read and navigate?
- Does the app take screen size differences into account?
- Does the user interface conform to the display and interaction standards adopted for the targeted mobile device(s)?
- Does the app conform to the reliability, security, and privacy expectations of its users?
- What provisions (规定) have been made to ensure app remains current?
- Has the MobileApp been tested in all targeted user environments and for all targeted devices?



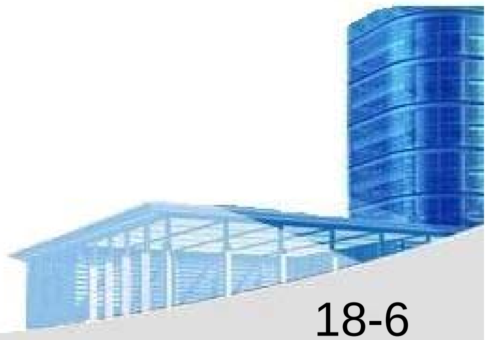
18.4 MobileApp User Interface Design Considerations

- Define user interface brand signatures
- Focus the portfolio of products
- Identify core user stories
- Optimize UI flows and elements
- Define scaling rules
- Create user performance dashboard (仪表盘)
- Rely on dedicated champion with user interface engineering skills



18.5 MobileApp Design Mistakes

- Kitchen sink
- Inconsistency
- Overdesigning
- Lack of speed
- Verbiage (冗词)
- Non-standard interaction
- Help-and –FAQ (Frequently Asked Questions，常见问题解答)-it is

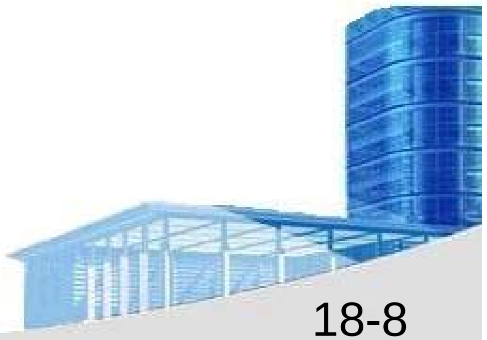


18.6 MobileApp Design Best Practices

- Identify the audience
- Design for context of use
- Recognize line between simplicity is not laziness
- Use the platform to its advantage
- Allow for discoverability of advanced functionality
- Use clear and consistent labels
- Cleaver icons should never be developed at the expense of user understanding
- Long scrolling forms trump (胜过) multiple screens

18.7 Assessing Mobile Interactive Development Environments

- General productivity features
- Third-party SDK integration
- Post-compilation(编辑) tools
- Over the air development support
- End-to-end mobile application development
- Documentation and tutorials
- Graphical User Interface (GUI) builders



18.8 MobileApp Middleware (中间件)

- Facilitates communication and coordination of distributed components
- Allows developers to rely on abstractions and hide mobile environment details
- Helps MobileApps to achieve context awareness as required

Case:智慧城管市民互动应用

智慧城管市民互动应用

让城市生活更便捷

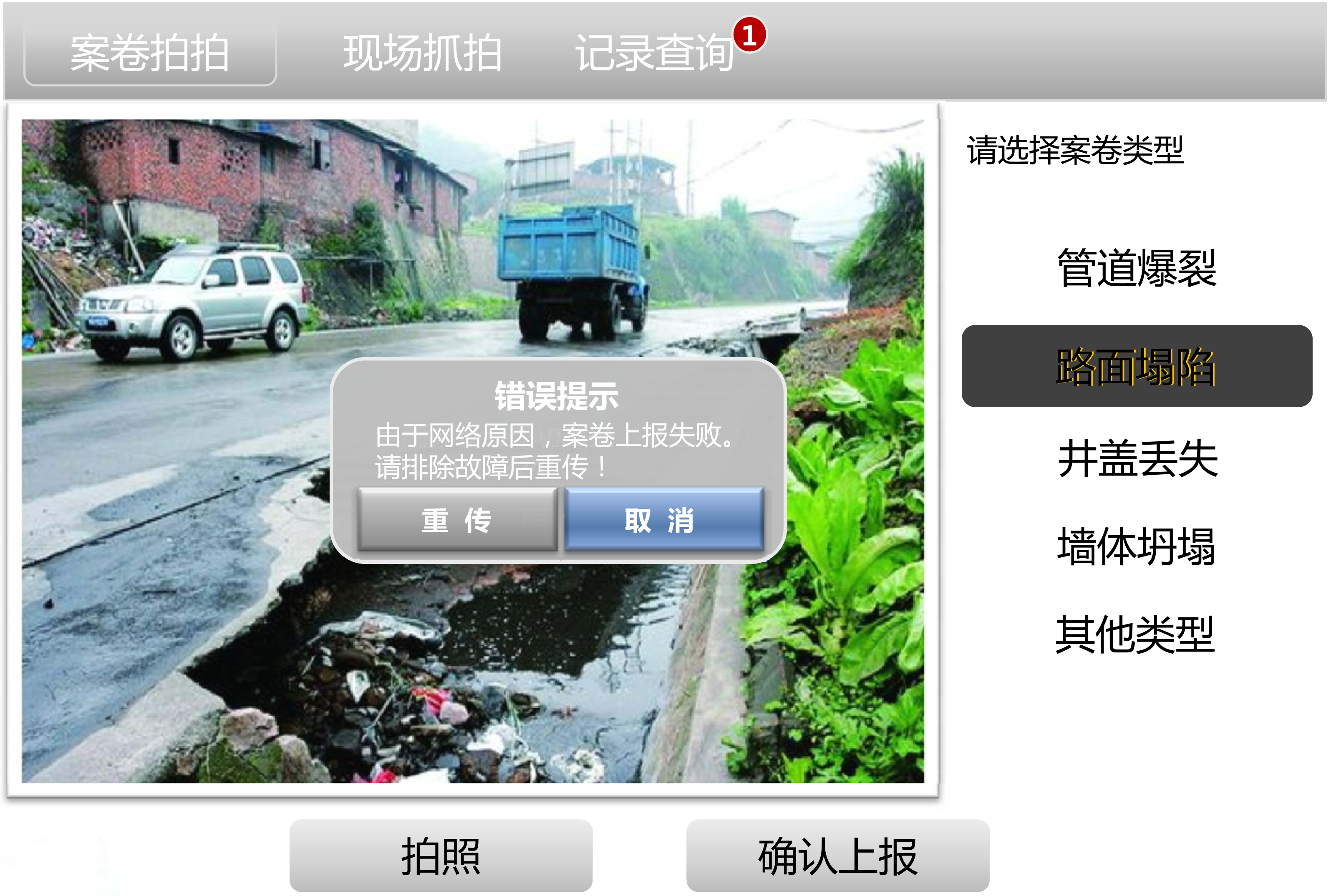
- 智慧城市建设示范试点项目
- 提供城市管理的多项公共服务
- 提供市民快速参与城市管理的渠道
- **iOS&Android**

智慧城管应用

城管指南	便民驿站	搜找公厕	政策法规	公交慢行
防汛抗冻	河道水质	紧急开挖	便捷热线	关联
我的积分	我的任务	我的爱车	我的宠物	我的店铺

智慧城管应用

城管拍拍	城管动态	便民公告	政策法规	每日一题
泊车达人	挪车求助	路况预告	云呼热线	天天骑车
我的积分	我的任务	我的爱车	我的宠物	我的店铺

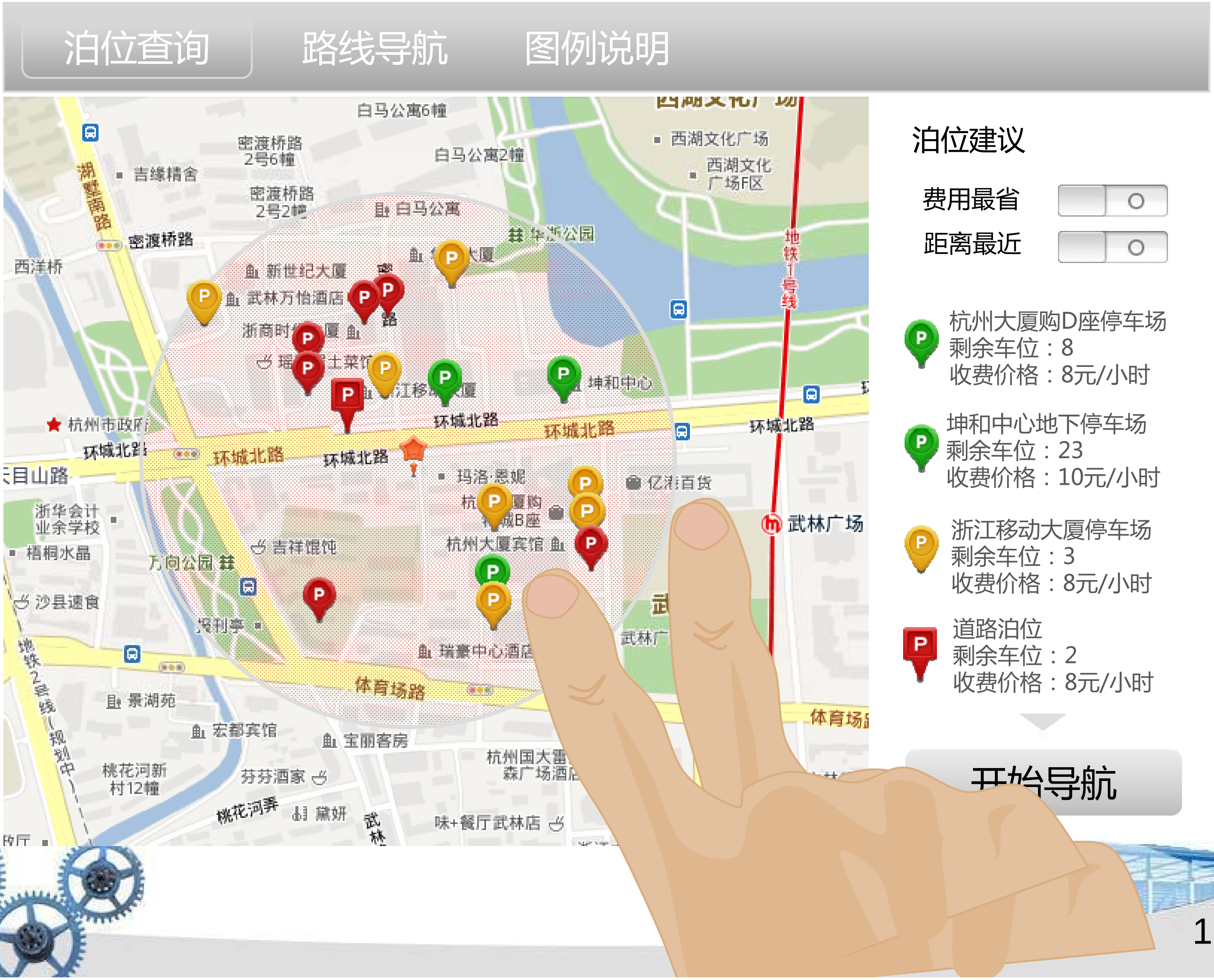


错误提示

由于网络原因，案卷上报失败。
请排除故障后重传！

智慧城管应用

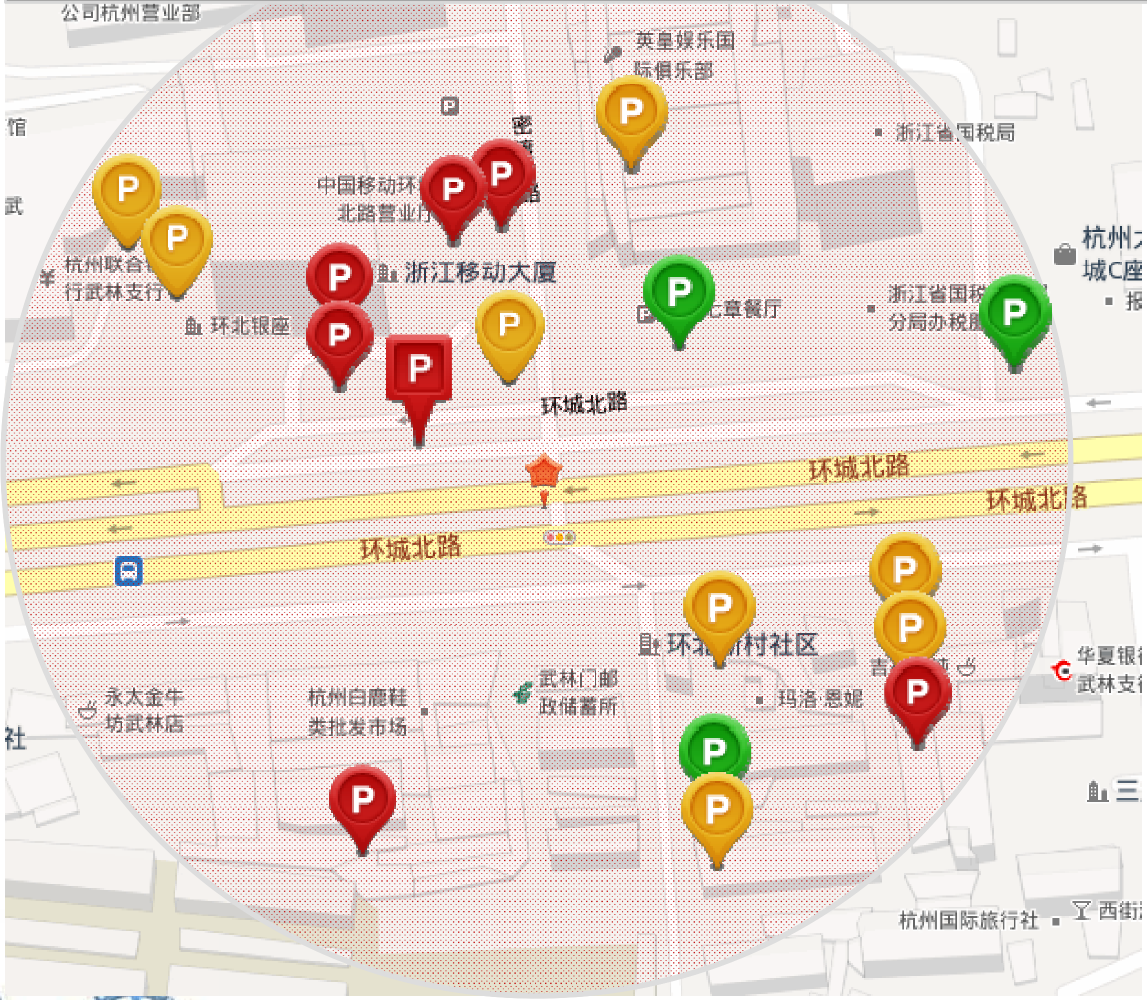
城管拍拍	城管动态	便民公告	政策法规	每日一题
泊车达人	挪车求助	路况预告	云呼热线	天天骑车
我的积分	我的任务	我的爱车	我的宠物	我的店铺



泊位建议

🟢	杭州大厦购D座停车场 剩余车位：8 收费价格：8元/小时
🟢	坤和中心地下停车场 剩余车位：23 收费价格：10元/小时
🟡	浙江移动大厦停车场 剩余车位：3 收费价格：8元/小时
🟥	道路泊位 剩余车位：2 收费价格：8元/小时

泊位查询　　路线导航　　图例说明



泊位建议

费用最省 ○

距离最近 ○

🟩 杭州大厦购D座停车场
剩余车位：8
收费价格：8元/小时

🟩 坤和中心地下停车场
剩余车位：23
收费价格：10元/小时

🟨 浙江移动大厦停车场
剩余车位：3
收费价格：8元/小时

🟥 道路泊位
剩余车位：2
收费价格：8元/小时

开始导航

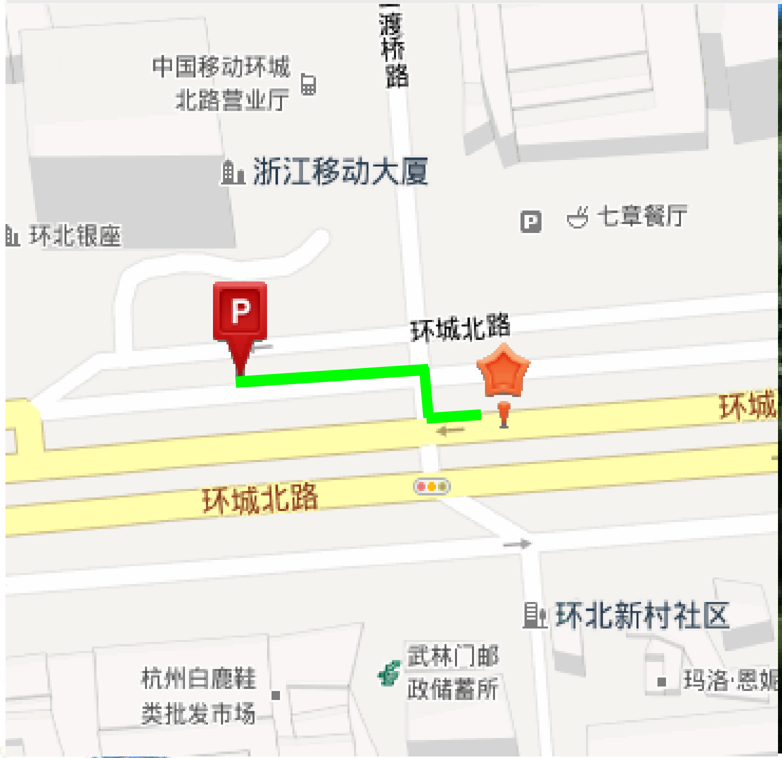
泊位查询　路线导航　图例说明

5m

环城北路

100m

前方靠右进入辅道，直行100m后使用泊位支付。



泊位查询　　路线导航　　图例说明

★	您当前所在的位置		
🟥	道路停车泊位紧张	🔴	小区企业内部停车泊位紧张
🟨	道路停车泊位较少	🟡	小区企业内部停车泊位较少
🟩	道路停车泊位充足	🟢	小区企业内部停车泊位充足
🔴	经营性停车泊位紧张		
🟡	经营性停车泊位较少		
🟢	经营性停车泊位充足		

Case: Features

- 应用界面简洁美观，易于上手
- 满足iOS、Android多平台使用需求
- 应用功能贴近用户实际需求
- 案件上报成功或错误均有明确提示
- 泊位查找选项充分考虑到用户喜好，可以进行个性化定制
- 注意点：后台数据库需要定时更新，保证数据准确性

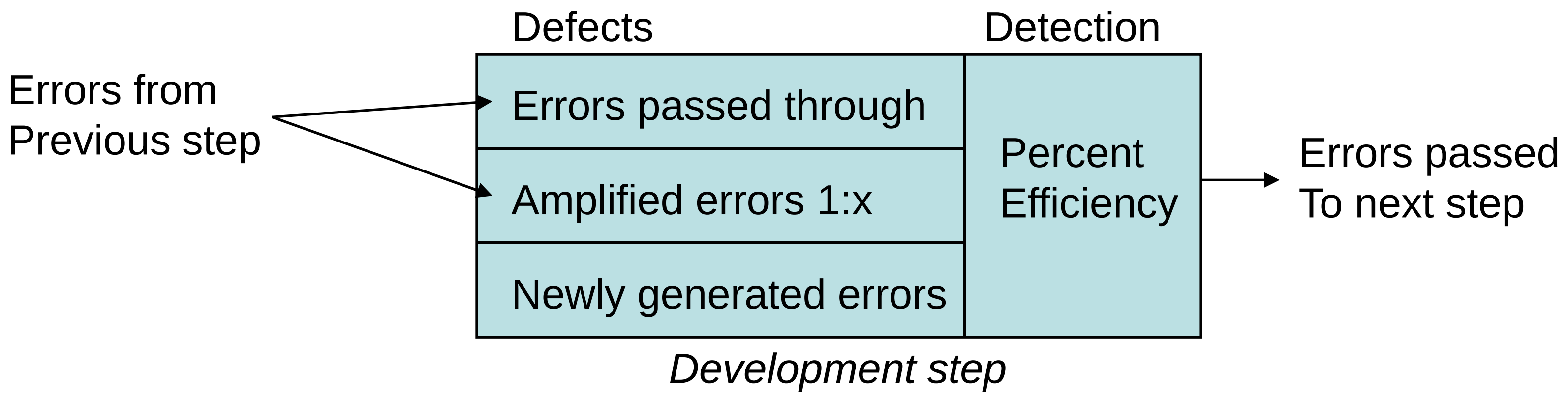
Ch.20 Review Techniques

20.1 Overview

- What Are Reviews?
  - a meeting conducted by technical people for technical people
  - a technical assessment of a work product created during the software engineering process
  - a software quality assurance mechanism
  - a training ground
- Errors and defects
  - Error—a quality problem found before the software is released to end users
  - Defect—a quality problem found only after the software has been released to end-users
- However, the temporal distinction made between errors and defects in this book is not mainstream thinking

20.2 Defect Amplification and Removal

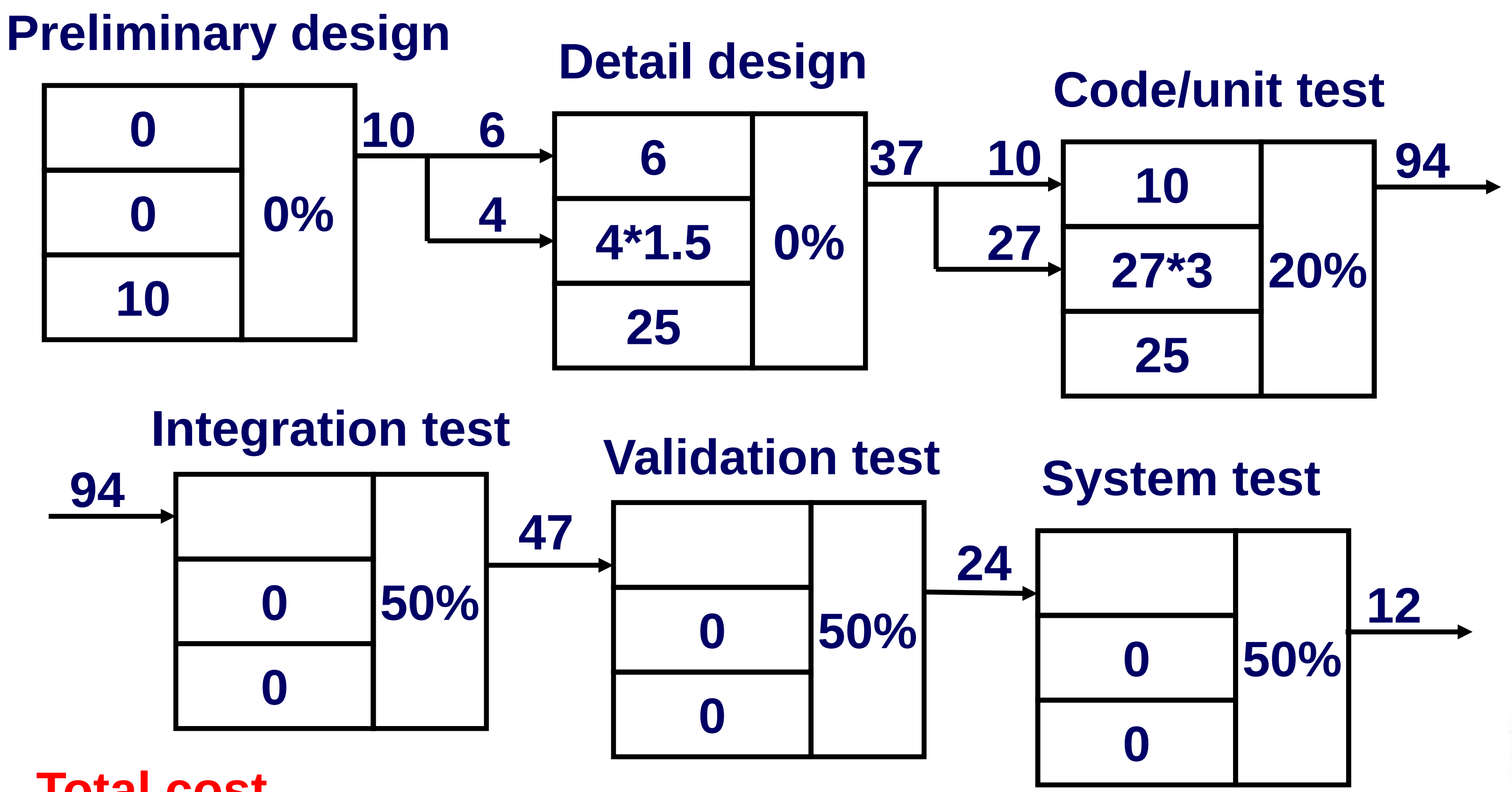
- **Defect Amplification Model**



- Assume that an error uncovered during **design** will cost **1.5** monetary unit to correct. Relative to this cost, the same error uncovered just **before testing** commences will cost **6.5** units; **during testing**, **15** units; and **after release**, between **67** and **100** units.
- A number of studies indicate that **design activities** introduce between **50% - 65%** of all errors during the software process. However, **formal review technique** have been shown to be up to **75%** effective in uncovering design flaws.

20.2 Defect Amplification and Removal

- **Example: Defect Amplification No Reviews**

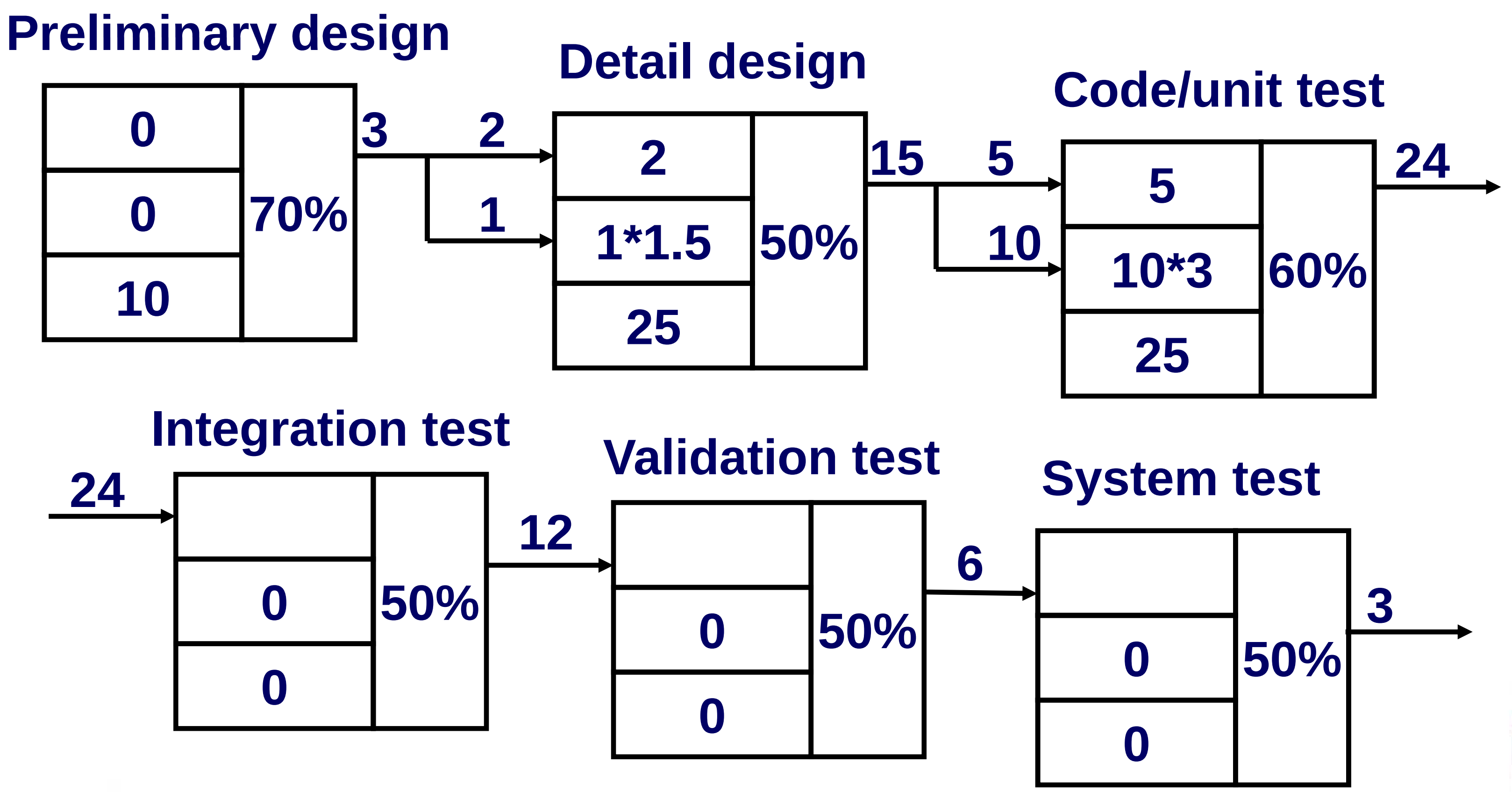


Total cost

= (10+27*3+25)*20%*6.5 + (94+47+24)*50%*15 + 12*67 = 2177

20.2 Defect Amplification and Removal

- **Example: Defect Amplification With Reviews**



Total cost = (10*70%+28.5*50%)*1.0 + (5+10*3+25)*60%*6.5 + (24+12+6)*50%*15 + 3*67 = **771**

20.3 Review Metrics and Their Use

- The total review effort and the total number of errors discovered are defined as:
  - $E_{review} = E_p + E_a + E_r$
  - $Err_{tot} = Err_{minor} + Err_{major}$
- *Defect density* represents the errors found per unit of work product reviewed.
  - Defect density = Err_{tot} / WPS

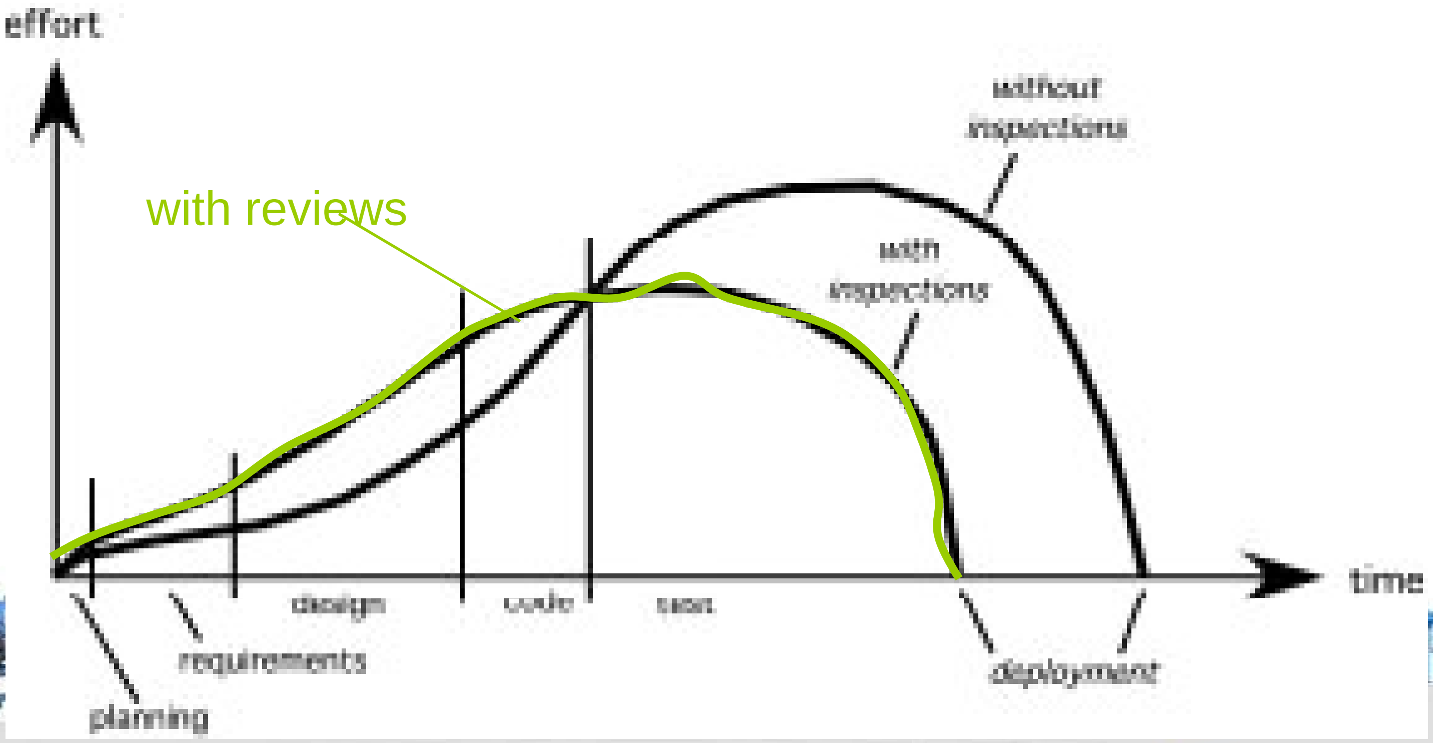
- *Preparation effort, E_p* — the effort (in person-hours) required to review a work product prior to the actual review meeting
- *Assessment effort, E_a* — the effort that is expending during the actual review
- *Rework effort, E_r* — the effort that is dedicated to the correction of those errors uncovered during the review
- *Work product size, WPS* — a measure of the size of the work product that has been reviewed (e.g., the number of UML models, or the number of document pages)
- *Minor errors found, Err_{minor}* — the number of errors found that can be categorized as minor (requiring less than some pre-specified effort to correct)
- *Major errors found, Err_{major}* — the number of errors found that can be categorized as major (requiring more than some pre-specified effort to correct)

Evaluate Saving: An Example—I

- The effort required to correct a minor model error (immediately after the review) was found to require 4 person-hours.
- The effort required for a major requirement error was found to be 18 person-hours.
- Examining the review data collected, you find that minor errors occur about 6 times more frequently than major errors. Therefore, you can estimate that the average effort to find and correct a requirements error during review is about 6 person-hours.
- Requirements related errors uncovered during testing require an average of 45 person-hours to find and correct. Using the averages noted, we get:
- Effort saved per error = $E_{testing} – E_{reviews}$
- 45 – 6 = 39 person-hours/error
- Since 22 errors were found during the review of the requirements model, a saving of about 660 person-hours of testing effort would be achieved. And that’s just for requirements-related errors.

20.3 Review Metrics

- Effort expended with and without reviews
  - The effort expended when reviews are used does increase earl, but this early investment for reviews pays dividends because testing and corrective effort is reduced.
  - The development date with reviews is sooner than the development date without reviews. Reviews don't take time, they save it.

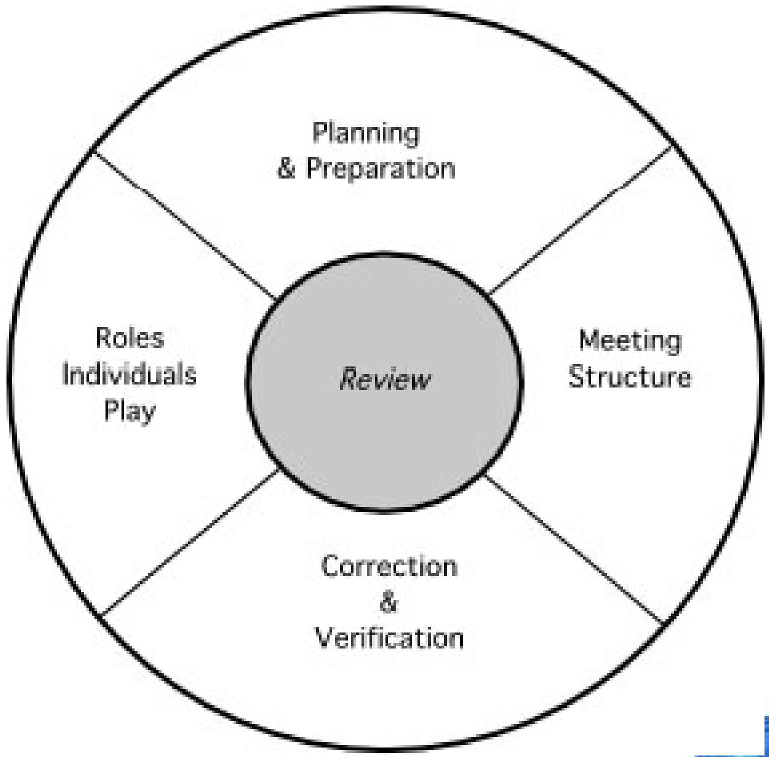


Prediction Work Performance: An Example—II

- If past history indicates that
  - the average defect density for a requirements model is 0.6 errors per page, and a new requirement model is 32 pages long,
  - a rough estimate suggests that your software team will find about 19 or 20 errors during the review of the document.
  - If you find only 6 errors, you've done an extremely good job in developing the requirements model *or* your review approach was not thorough enough.

20.4 Reference Model

- The formality of a review increases when:
  - Distinct roles are explicitly defined for the reviewers.
  - There is a sufficient amount of planning and preparation for the review.
  - A distinct structure for the review is defined.
  - Follow-up by the reviewers occurs for any corrections that are made.



20.5 Informal Reviews

- Informal reviews include:
  - a simple desk check of a software engineering work product with a colleague
  - a casual meeting (involving more than 2 people) for the purpose of reviewing a work product, or
  - the review-oriented aspects of pair programming
- *pair programming* encourages continuous review as a work product (design or code) is created.
  - The benefit is immediate discovery of errors and better work product quality as a consequence.

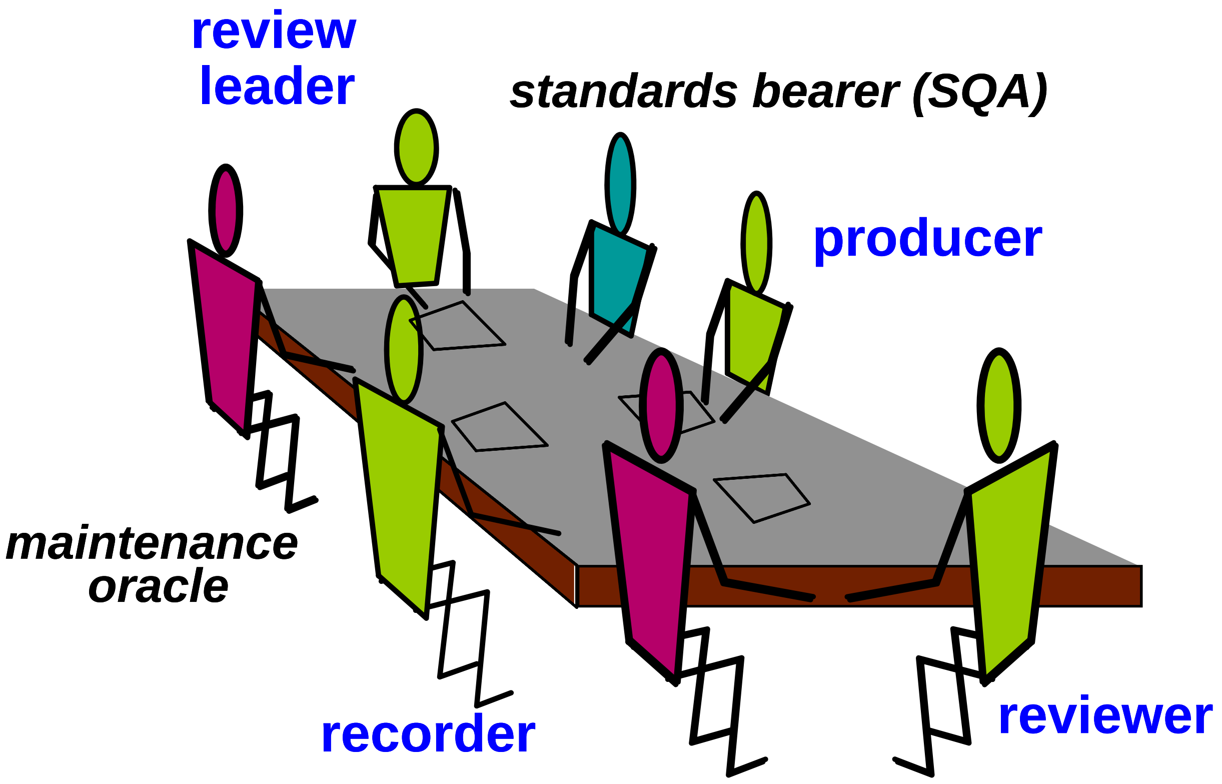
20.6 Formal Technical Reviews （FTR）

- The objectives of an FTR are:
  - to uncover errors in function, logic, or implementation for any representation of the software
  - to verify that the software under review meets its requirements
  - to ensure that the software has been represented according to predefined standards
  - to achieve software that is developed in a uniform manner
  - to make projects more manageable
- The FTR is actually a class of reviews that includes *walkthroughs* and *inspections*.

20.6.1 The Review Meeting

- Between three and five people (typically) should be involved in the review.
- Advance preparation should occur but should require no more than two hours of work for each person.
- The duration of the review meeting should be less than two hours.
- Focus is on a work product (e.g., a portion of a requirements model, a detailed component design, source code for a component)。

The Players


review leader
standards bearer (SQA)
producer
maintenance oracle
recorder
reviewer

Process

- ***Role of players***
  - *Producer*—the individual who has developed the work product
    - informs the project leader that the work product is complete and that a review is required
  - *Review leader*—evaluates the product for readiness, generates copies of product materials, and distributes them to two or three *reviewers* for advance preparation.
  - *Reviewer(s)*—expected to spend between one and two hours reviewing the product, making notes, and otherwise becoming familiar with the work.
  - *Recorder*—reviewer who records (in writing) all important issues raised during the review.

⑩ ***Process***

- *Preparation phase: producer->review leader->reviewers->problem list*
- *Perform phase: producer introduction -> reviewers raise issue->recorder*
- *Track phase:* conclusion、*SQA Report；*

20.6.3 Review Guidelines

- Review the product, not the producer.
- Set an agenda（议事日程） and maintain it.
- Limit debate and rebuttal（ 反驳 ）.
- Enunciate （ 确切说明 ） problem areas, but don't attempt to solve every problem noted.
- Take written notes.
- Limit the number of participants and insist upon advance preparation.
- Develop a checklist for each product that is likely to be reviewed.
- Allocate resources and schedule time for FTRs.
- Conduct meaningful training for all reviewers.
- Review your early reviews.

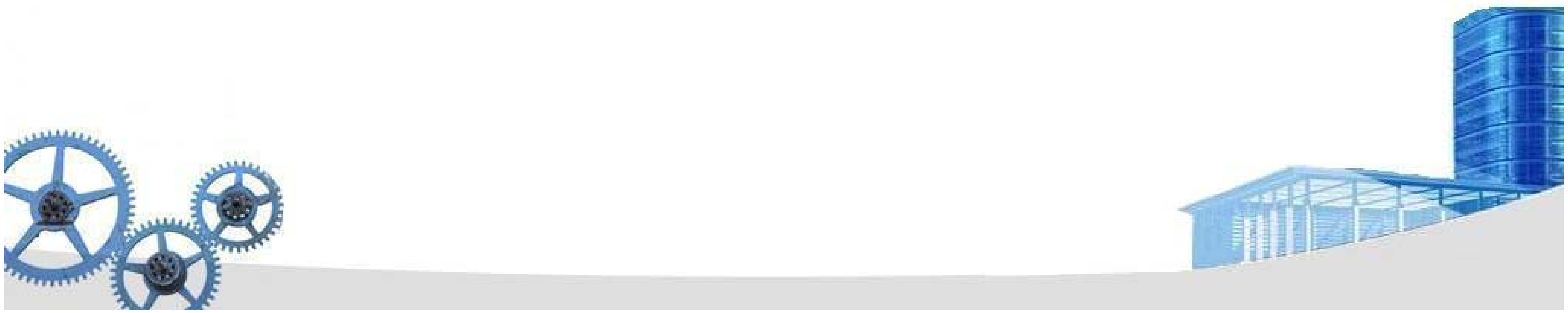
20.6.4 Sample-Driven Reviews (SDRs)

- SDRs attempt to quantify those work products that are primary targets for full FTRs.

To accomplish this …

- Inspect a fraction a_i of each software work product, *i*. Record the number of faults, f_i found within a_i.
- Develop a gross estimate of the number of faults within work product *i* by multiplying f_i by $1/a_i$.
- Sort the work products in descending order according to the gross estimate of the number of faults in each.
- Focus available review resources on those work products that have the highest estimated number of faults.

Ch.21 Software Quality Assurance



21.1 Comment on Quality

- Phil Crosby once said:
  - The problem of quality management is not what people don't know about it. The problem is what they think they do know . . . In this regard, quality has much in common with sex.
  - *Everybody is for it.* (Under certain conditions, of course.)
  - *Everyone feels they understand it.* (Even though they wouldn't want to explain it.)
  - *Everyone thinks execution is only a matter of following natural inclinations*（倾向）*.* (After all, we do get along somehow.)
  - *And, of course, most people feel that problems in these areas are caused by other people.* (If only they would take the time to do things right.)

21.2 Elements of SQA

- Standards
- Reviews and Audits
- Testing
- Error/defect collection and analysis
- Change management
- Education
- Vendor（ 供应商 ） management
- Security management
- Safety
- Risk management

21.4.1 SQA Tasks I

- **Prepares an SQA plan for a project.**
  - The plan identifies
    - evaluations to be performed
    - audits and reviews to be performed
    - standards that are applicable to the project
    - procedures for error reporting and tracking
    - documents to be produced by the SQA group
    - amount of feedback provided to the software project team
- **Participates in the development of the project’s software process description.**
  - The SQA group reviews the process description for compliance with organizational policy, internal software standards, externally imposed standards (e.g., ISO-9001), and other parts of the software project plan.

21.4.1 SQA Tasks II

- **Reviews software engineering activities to verify compliance with the defined software process.**
  - identifies, documents, and tracks deviations from the process and verifies that corrections have been made.
- **Audits designated software work products to verify compliance with those defined as part of the software process.**
  - reviews selected work products; identifies, documents, and tracks deviations; verifies that corrections have been made
  - periodically reports the results of its work to the project manager.
- **Ensures that deviations in software work and work products are documented and handled according to a documented procedure.**
- **Records any noncompliance and reports to senior management.**
  - Noncompliance items are tracked until they are resolved.

21.4.2 SQA Goals

- **Requirements quality.** The correctness, completeness, and consistency of the requirements model will have a strong influence on the quality of all work products that follow.

- **Design quality.** Every element of the design model should be assessed by the software team to ensure that it exhibits high quality and that the design itself conforms to requirements.

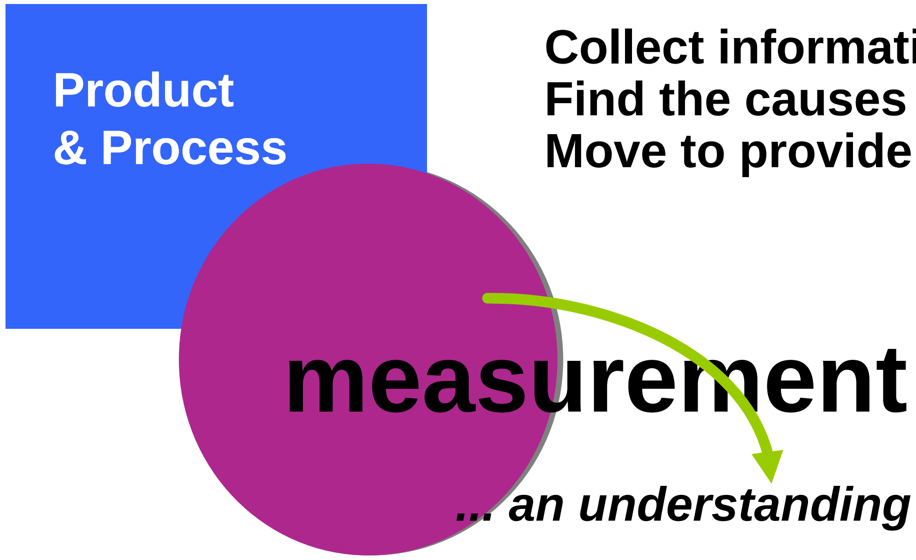
- **Code quality.** Source code and related work products (e.g., other descriptive information) must conform to local coding standards and exhibit characteristics that will facilitate maintainability.

- **Quality control effectiveness.** A software team should apply limited resources in a way that has the highest likelihood of achieving a high quality result. (see Figure 21.1)

21.5 Formal SQA

- Assumes that a rigorous syntax and semantics can be defined for every programming language
- Allows the use of a rigorous approach to the specification of software requirements
- Applies mathematical proof of correctness techniques to demonstrate that a program conforms to its specification

21.6 Statistical SQA



Collect information on all defects
Find the causes of the defects
Move to provide fixes for the process

… an understanding of how to improve quality…

- Information about software errors and defects is collected and categorized.
- An attempt is made to trace each error and defect to its underlying(潜在的) cause.

- Using the Pareto principle (80 percent of the defects can be traced to 20 percent of all possible causes), isolate the 20 percent (the *vital few*).
- Once the vital few causes have been identified, move to correct the problems that have caused the errors and defects.

21.6.2 Six-Sigma for Software Engineering

- The term “six sigma” is derived from six standard deviations—3.4 instances (defects) per million occurrences—implying an extremely high quality standard.
- The Six Sigma methodology defines three core steps:
  - *Define* customer requirements and deliverables and project goals via well-defined methods of customer communication
  - *Measure* the existing process and its output to determine current quality performance (collect defect metrics)
  - *Analyze* defect metrics and determine the vital few causes.
  - *Improve* the process by eliminating the root causes of defects.
  - *Control* the process to ensure that future work does not reintroduce the causes of defects.

	# of defects / million occurrences
6 Sigma	3.4 (0.00034%)
5 Sigma	230 (0.023%)
4 Sigma	6210 (0.621%)
3 Sigma	66,800 (6.68%)
2 Sigma	308,000 (30.8%)
1 Sigma	690,000 (69%)

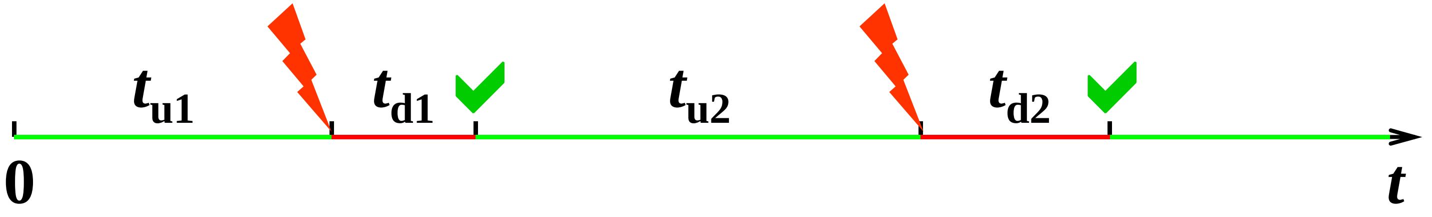
21.7.1 Software Reliability

Reliability = MTBF(mean-time-between-failure)
= MTTF + MTTR

Availability = $\frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}$ (Shooman, 1983)

where **MTTF** = Mean Time To **F**ailure = $\frac{1}{n}\sum_{i=1}^{n} t_{ui}$

MTTR = Mean Time To **R**epair = $\frac{1}{n}\sum_{i=1}^{n} t_{di}$



21.7.2 Software Safety

- *Software safety* is a software quality assurance activity that focuses on the identification and assessment of potential hazards that may affect software negatively and cause an entire system to fail.

- If hazards can be identified early in the software process, software design features can be specified that will either eliminate or control potential hazards.

21.8 ISO 9001:2008 Standard

- ISO 9001:2008 is the quality assurance standard that applies to software engineering.
- The standard contains 20 requirements that must be present for an effective quality assurance system.
- The requirements delineated by ISO 9001:2008 address topics such as
  - management responsibility, quality system, contract review, design control, document and data control, product identification and traceability, process control, inspection and testing, corrective and preventive action, control of quality records, internal quality audits, training, servicing, and statistical techniques.

➢ **ISO9000贯标文件**

Related Doc

- 质量保证过程
- 质量保证计划
- QA检查汇总及记分表
- SQA阶段工作表
- 软件过程审计报告

Tasks

- **Review** Ch.18,20,21
- **Finish** “Problems and points to ponder” in Ch. 18, 20,21
- **Preview** Ch. 14,15,16
- **Submit System design Specification due May 13!**