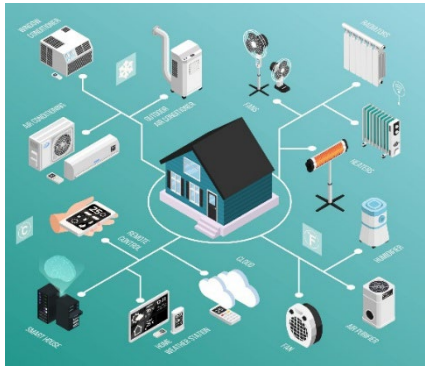
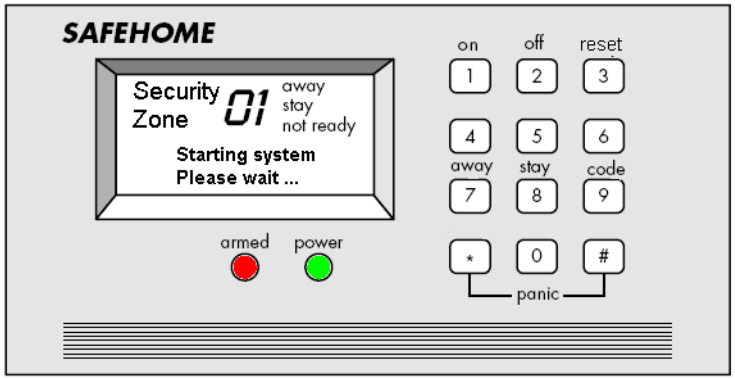
Quick Overview of SafeHome

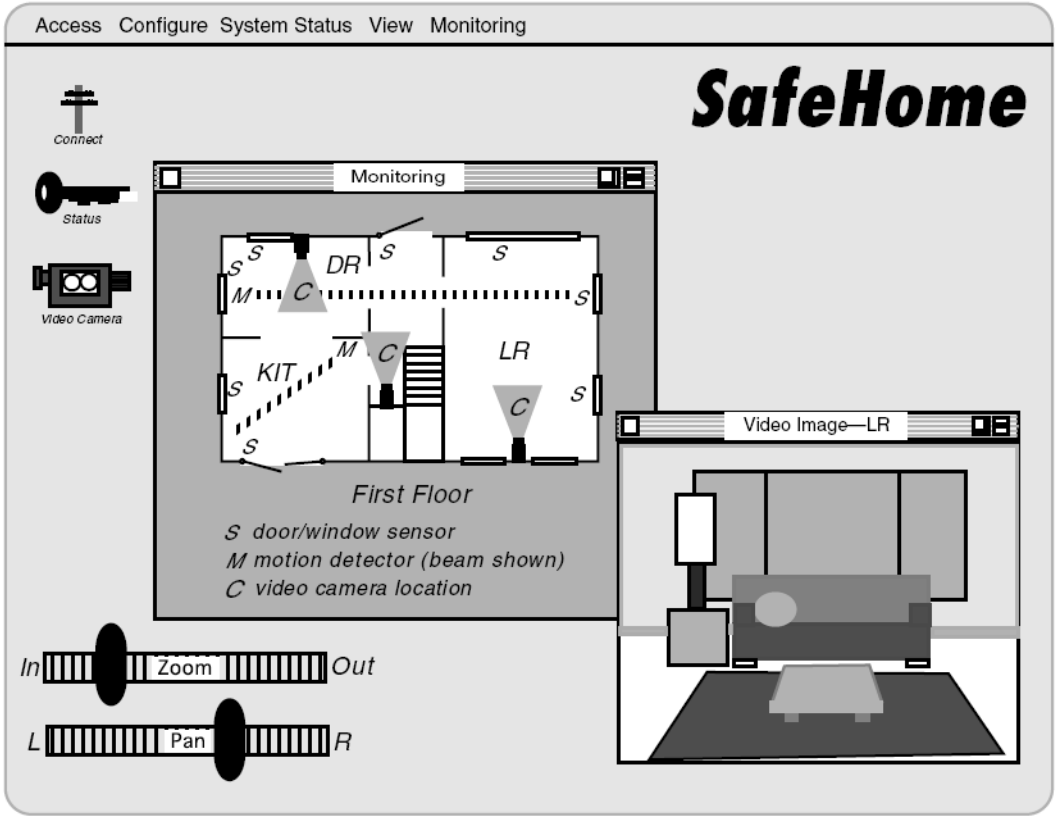


- The SafeHome company has developed an innovative HW box that implements wireless Internet (802.11n) connectivity in a very small form factor (the size of a matchbook).
- The idea is to use this technology to develop and market a comprehensive home automation product line.
  - This would provide
    - security functions
    - control over telephone answering machines
    - lights
    - heating
    - air conditioning
    - home entertainment devices.

- The first generation of the system will only focus on home security since that is a market the public readily understands.

SafeHome Product Prototype





How a Project Starts (ch1. pg 16)

- **The scene:**
  - Meeting room at CPI Corporation, a (fictional) company that makes consumer products for home and commercial use.
- **The players:**
  - Mal Golden, senior manager, product development;
  - Lisa Perez, marketing manager;
  - Lee Warren, engineering manager;
  - Joe Camalleri, executive VP, business development.

The conversation:

- **Joe:** Okay, Lee, what's this I hear about your folks developing a what? A generic universal wireless box?
- **Lee:** It's pretty cool, about the size of a small matchbook. We can attach it to sensors of all kinds, a digital camera, just about anything. Using the 802.11 b wireless protocol. It allows us to access the device's output without wires. We think it'll lead to a whole new generation of products.
- **Joe:** You agree, Mal?
- **Mal:** I do. In fact, with sales as flat as they've been this year, we need something new. Lisa and I have been doing a little market research, and we think we've got a line of products that could be big.
- **Joe:** How big... , bottom-line big?

- **Mal: (avoiding a direct commitment):** Tell him about our idea, Lisa.
- **Lisa:** It's a whole new generation of what we call "home management products." We call 'em *SafeHome.* They use the new wireless interface, provide homeowners or small business people with a system that's controlled by their PC--home security, home surveillance, appliance and device control. You know, turn down the home air conditioner while you're driving home, that sort of thing.
- **Lee: (jumping in)** Engineering's done a technical feasibility study of this idea, Joe. It's doable at low manufacturing cost. Most hardware is off the shelf. Software is an issue, but it's nothing that we can't do.
- **Joe:** Interesting. Now, I asked about the bottom line.
- **Mal:** PCs have penetrated 60 percent of all households in the USA. If we could price this thing right, it could be a killer-App. Nobody else has our wireless box--it's proprietary. We'll have a two-year jump on the competition. Revenue? Maybe as much as $30-40 million in the second year.
- **Joe (smiling):** Let's take this to the next level. I'm interested.

Selecting a Process Model, Part 1(ch2. pg 28)

- **The scene:**
  - Meeting room for the software engineering group at CPI Corporation, a (fictional) company that makes consumer products for home and commercial use.
- **The players:**
  - Lee Warren, engineering manager;
  - Doug Miller, software engineering manager;
  - Jamie Lazar, software team member;
  - Vinod Raman, software team member;
  - Ed Robbins, software team member.
- **The conversation:**
- **Lee:** So let's recapitulate. I've spent some time discussing the *SafeHome* product line as we see it at the moment. No doubt, we've got a lot of work to do to simply define the thing, but I'd like you guys to begin thinking about how you're going to approach the software part of this project.
- **Doug:** Seems like we've been pretty disorganized in our approach to software in the past.
- **Ed:** I don't know, Doug. We always got product out the door.
- **Doug:** True, but not without a lot of grief, and this project looks like it's bigger and more

complex than anything we've done in the past.

■**Jamie:** Doesn't look that hard, but I agree ... our ad hoc approach to past projects won't work here, particularly if we have a very tight timeline.

■**Doug (smiling):** I want to be a bit more professional in our approach. I went to a short course last week and learned a lot about software engineering ... good stuff. We need a process here.

■**Jamie (with a frown):** My job is to build computer programs, not push paper around.

■**Doug:** Give it a chance before you go negative on me. Here's what I mean. [Doug proceeds to describe the process framework described in Chapter 2 and the prescriptive process models presented to this point.

■**Doug:** So anyway, it seems to me that a linear model is not for us ... assumes we have all requirements up front and knowing this place, that's not likely.

■**Vinod:** Yeah, and that RAD model sounds way too IT-

oriented ... probably good for building an inventory control system or something, but it's just not right for *SafeHome.*

- **Doug:** I agree.
- **Ed:** That prototyping approach seems OK. A lot like what we do here anyway.
- **Vinod:** That's a problem. I'm worried that it doesn't provide us with enough structure.
- **Doug:** Not to worry. We've got plenty of other options, and I want you guys to pick what's best for the team and best for the project.

Selecting a Process Model, Part 2(ch2. pg 31)

- **The scene:**
  - Meeting room for the software engineering group at CPI Corporation, a company that makes consumer products for home and commercial use.
- **The players:**
  - Lee Warren, engineering manager;
  - Doug Miller, software engineering manager;
  - Ed and Vinod, members of the software engineering team.
- **The conversation:**
- (Doug describes evolutionary process options.)
- Ed: Now I see something I like. An incremental approach makes sense and I really like the flow of that spiral model thing. That's keepin' it real.
- Vinod: I agree. We deliver an increment, learn from customer feedback, re-plan, and then deliver another increment. It also fits into the nature of the product. We can have something on the market fast and then add functionality with each version, er, increment.
- Lee: Wait a minute, did you say that we regenerate the plan with each tour around the spiral, Doug? That's not so great, we

need one plan, one schedule, and we've got to stick to it.

- **Doug:** That's old school thinking, Lee. Like Ed said, we've got to keep it real. I submit that it's better to tweak the plan as we learn more and as changes are requested. It's way more realistic. What's the point of a plan if it doesn't reflect reality?
- **Lee (frowning):** I suppose so, but senior management's not going to like this ... they want a fixed plan.
- **Doug (smiling):** Then you'll have to reeducate them, buddy.

Considering Agile Software Development (ch3. pg 43)

- **The scene:**
  - Doug Miller's office.
- **The players:**
  - Doug Miller, software engineering manager;
  - Jamie Lazar, software team member;
  - Vinod Raman, software team member.
- **The conversation:**
- (A knock on the door)
- **Jamie:** Doug, you got a minute?
- **Doug:** Sure Jamie, what's up?
- **Jamie:** We've been thinking about our process discussion yesterday ... you know, what process we're going to choose for this new *SafeHome* project.
- **Doug:** And?
- **Vinod:** I was talking to a friend at another company, and he was telling me about Extreme Programming. It's an agile process model, heard of it?
- **Doug:** Yeah, some good, some bad.
- **Jamie:** Well, it sounds pretty good to us. Lets you develop software really fast, uses something called pair programming to do real-time quality checks ... it's pretty cool, I think.
- **Doug:** It does have a lot of really good ideas. I like the pair

programming concept, for instance, and the idea that stakeholders should be part of the team.

- **Jamie:** Huh? You mean that marketing will work on the project team with us?
- **Doug (nodding):** They're a stakeholder, aren't they?
- **Jamie:** Jeez ... they'll be requesting changes every five minutes.
- **Vinod:** Not necessarily. My friend said that there are ways to "embrace" changes during an XP project.
- **Doug:** So you guys think we should use XP?
- **Jamie:** It's definitely worth considering.
- **Doug:** I agree. And even if we choose an incremental model as our approach, there's no reason why we can't incorporate much of what XP has to offer.
- **Vinod:** Doug, before you said "some good, some bad." What was the "bad"?
- **Doug:** The thing I don't like is the way XP downplays analysis and design ... sort of says that writing code is where the action is.

- (The team members look at one another and smile.)
- **Doug:** So you agree with the XP approach?
- **Jamie (speaking for both):** Writing code is what we do, Boss!
- **Doug (laughing):** True, but I'd like to see you spend a little less time coding and then re-coding and a little more time analyzing what has to be done and designing a solution that works.
- **Vinod:** Maybe we can have it both ways, agility with a little discipline.
- **Doug:** I think we can, Vinod. In fact, I'm sure of it.

Team Structure (ch5. pg 79)

- **The scene:**
  - Doug Miller's office prior to the initiation of the *SafeHome* software project.
- **The players:**
  - Doug Miller (manager of the *SafeHome* software engineering team)
  - Vinod Raman, Jamie Lazar, other members of the product software engineering team.
- **The conversation:**
- **Doug:** Have you guys had a chance to look over the preliminary info on *SafeHome* that marketing's prepared?
- **Vinod (nodding and looking at his teammates):** Yes. But we have a bunch of questions.
- **Doug:** Let's hold on that for a moment. I'd like to talk about how we're going to structure the team, who's responsible for what. . . .
- **Jamie:** I'm really into the agile philosophy, Doug. I think we should be a self-organizing team.
- **Vinod:** I agree. Given the tight time line and some of the uncertainty, and that fact that we're all really competent [laughs], that seems like the right way to go.

- **Doug:** That's okay with me, but you guys know the drill.
- **Jamie (smiling and talking as if she were reciting something):** We make tactical decisions, about who does what and when, but it's our responsibility to get product out the door on time.
- **Vinod:** and with quality.
- **Doug:** Exactly. But remember there are constraints. Marketing defines the software increments to be produced--in consultation with us, of course.
- **Jamie:** And?

Communication Mistakes (ch 6. pg 90)

- **The scene:**
  - Software engineering team workspace.
- **The players:**
  - Jamie Lazar, software team member;
  - Vinod Raman, software team member;
  - Ed Robbins software team member.
- **The conversation:**
- **Ed:** What have you heard about this *SafeHome* project?
- **Vinod:** The kick-off meeting is scheduled for next week.
- **Jamie:** I've already done a little bit of investigation, but it didn't go well."
- **Ed:** What do you mean?
- **Jamie:** Well, I gave Lisa Perez a call. She's the marketing honcho on this thing."
- **Vinod:** And ... ?
- **Jamie:** I wanted her to tell me about *SafeHome* features and functions ... that sort of thing. Instead, she began asking me questions about security systems, surveillance systems ... I'm no expert.
- **Vinod:** What does that tell you?
- (**Jamie** shrugs.)
- **Vinod:** That marketing will need

us to act as consultants and that we'd better do some homework on this product area before our kick-off meeting. Doug said that he wanted us to "collaborate" with our customer, so we'd better learn how to do that.

- **Ed:** Probably would have been better to stop by her office. Phone calls just don't work as well for this sort of thing.
- **Jamie:** You're both right. We've got to get our act together or our early communications will be a struggle.
- **Vinod: I** saw Doug reading a book on "requirements engineering." I'll bet that lists some principles of good communication. I'm going to borrow it from him.
- **Jamie:** Good idea ... then you can teach us.
- **Vinod (smiling):** Yeah, right.

Conducting a Requirements Gathering Meeting (ch7. pg112)

- **The scene:**
  - A meeting room. The first requirements gathering meeting is in progress.
- **The players:**
  - Jamie Lazar, software team member;
  - Vinod Raman, software team member;
  - Ed Robbins, software team member;
  - Doug Miller, software engineering manager;
  - three members of marketing;
  - a product engineering representative;
  - a facilitator.

- **The conversation:**
- **Facilitator (pointing at white board):** So that's the current list of objects and services for the home security function.
- **Marketing person:** That about covers it from our point of view.
- **Vinod:** Didn't someone mention that they wanted all *SafeHome* functionality to be accessible via the Internet? That would include the home security function, no?
- **Marketing person:** Yes, that's right ... we'll have to add that functionality and the appropriate objects.

- **Facilitator:** Does that also add some constraints?
- **Jamie:** It does, both technical and legal.
- **Production rep:** Meaning?
- **Jamie:** We better make sure an outsider can't hack into the system, disarm it, and rob the place or worse. Heavy liability on our part.
- **Doug:** Very true.
- **Marketing:** But we still need Internet connectivity . just be sure to stop an outsider from getting in.
- **Ed:** That's easier said than done and....
- **Facilitator (interrupting):** I don't want to debate this issue now. Let's note it as an action item and proceed. (Doug, serving as the recorder for the meeting, makes an appropriate note.)
- **Facilitator:** I have a feeling there's still more to consider here.
- (The group spends the next 45 minutes refining and expanding the details of the home security function.)

Developing a Preliminary User Scenario (ch7. pg 113)

- **The scene:**
  - A meeting room, continuing the first requirements gathering meeting.
- **The players:**
  - Jamie Lazar, software team member;
  - Vinod Raman, software team member;
  - Ed Robbins, software team member;
  - Doug Miller, software engineering manager;
  - three members of marketing;
  - a product engineering representative;
  - a facilitator.

- **The conversation:**
- **Facilitator:** We've been talking about security for access to *SafeHome* functionality that will be accessible via the Internet. I'd like to try something.
- Let's develop a user scenario for access to the home security function.
- **Jamie:** How?
- **Facilitator:** We can do it a couple of different ways, but for now, I'd like to keep things really informal. Tell us (he points at a marketing person) how you envision accessing the system.

- **Marketing person:** Um. .. , well, this is the kind of thing I'd do if I was away from home and I had to let someone into the house, say a housekeeper or repair guy, who didn't have the security code.
- **Facilitator (smiling):** That's the *reason* you'd do it .. . tell me *how* you'd actually do this.
- **Marketing person:** Um . . . the first thing I'd need is a PC. I'd log on to a Web site we'd maintain for all users of *SafeHome*. I'd provide my user id and ...
- **Vinod (interrupting):** The Web page would have to be secure, encrypted, to guarantee that we're safe and....
- **Facilitator (interrupting):** That's good information, Vinod, but it's technical. Let's just focus on how the end-user will use this capability, OK?
- **Vinod:** No problem.
- **Marketing person:** So, as I was saying, I'd log on to a Web site and provide my user id and two levels of passwords.
- **Jamie:** What if I forget my password?

- **Facilitator (interrupting):** Good point, Jamie, but let's not address that now. We'll make a note of that and call it an "exception." I'm sure there'll be others.
- **Marketing person:** After I enter the passwords, a screen representing all *SafeHome* functions will appear. I'd select the home security function. The system might request that I verify who I am, say by asking for my address or phone number or something. It would then display a picture of the security system control panel along with a list of functions that I can perform--arm the system, disarm the system, disarm one or more sensors. I suppose it might also allow me to reconfigure security zones and other things like that, but I'm not sure.
- (As the marketing person continues talking, Doug takes copious notes. These form the basis for the first informal use-case scenario. Alternatively, the marketing person could have been asked to write the scenario, but this would be done outside the meeting.)

Developing a High-Level Use-Case Diagram (ch 7.pg 118)

- **The scene:**
  - A meeting room, continuing the requirements gathering meeting.
- **The players:**
  - Jamie Lazar, software team member;
  - Vinod Raman, software team member;
  - Ed Robbins, software team member;
  - Doug Miller, software engineering manager;
  - three members of marketing;
  - a product engineering representative;
  - a facilitator.

- **The conversation:**
- **Facilitator:** We've spent a fair amount of time talking about *SafeHome* home security functionality. During the break I sketched a use-case diagram to summarize the important scenarios that are part of this function. Take a look.
- (All attendees look at Figure 7.3.)
- **Jamie:** I'm just beginning to learn UML notation. So the home security function is represented by the big box with the ovals inside it? And the

ovals represent use-cases that we've written in text?

- **Facilitator:** Yep. And the stick figures represent actors--the people or things that interact with the system
- as described by the use-case ... oh, I use the labeled square to represent an actor that's not a person, in this case, sensors.
- **Doug:** Is that legal in UML?
- **Facilitator:** Legality isn't the issue. The point is to communicate information. I view the use of a human-like stick figure for representing a device to be misleading. So I've adapted things a bit. I don't think it creates a problem.
- **Vinod:** Okay, so we have use-case narratives for each of the ovals. Do we need to develop the more detailed template-based narratives I've read about?
- **Facilitator:** Probably, but that can wait until we've considered other *SafeHome* functions.
- **Marketing person:** Wait, I've been looking at this diagram, and all of a sudden I realize we missed something.
- **Facilitator:** Oh really. Tell me what we've missed. (The meeting continues.)

Preliminary Behavioral Modeling (ch7. pg 121)

- **The scene:**
  - A meeting room, continuing the requirements meeting.
- **The players:**
  - Jamie Lazar, software team member;
  - Vinod Raman, software team member;
  - Ed Robbins, software team member;
  - Doug Miller, software engineering manager;
  - three members of marketing;
  - a product engineering representative;
  - a facilitator.

- **The conversation:**
- **Facilitator:** We've just about finished talking about *SafeHome* home security functionality. But before we do, I want to discuss the behavior of the function.
- **Marketing person:** I don't understand what you mean by behavior.
- **Ed (laughing):** That's when you give the product a "timeout" if it misbehaves.
- **Facilitator:** Not exactly. Let me explain.
- (The facilitator explains the

basics of behavioral modeling to the requirements gathering team.)

- **Marketing person:** This seems a little technical. I'm not sure I can help here.
- **Facilitator:** Sure you can. What behavior do you observe from the user's point of view?
- **Marketing person:** **Uh... ,** well the system will be *monitoring* the sensors. It'll be *reading commands* from the homeowner. It'll be *displaying* its status.
- **Facilitator:** See, you can do it.
- **Jamie:** It'll also be *polling* the PC to determine if there is any input from it, for example Internet-based access or configuration information.
- **Vinod:** Yeah, in fact, *configuring the system* is a state in its own right.
- **Doug:** You guys are rolling. Let's give this a bit more thought . . . Is there a way to diagram this stuff?
- **Facilitator:** There is, but let's postpone that until after the meeting.

The Start of a Negotiation (ch7. pg 122)

- **The scene:**
  - Lisa Perez's office, after the first requirements gathering meeting.
- **The players:**
  - Doug Miller, software engineering manager
  - Lisa Perez, marketing manager.
- **The conversation:**
- **Lisa:** So, I hear the first meeting went really well.
- **Doug:** Actually, it did. You sent some good people to the meeting ... they really contributed.
- **Lisa (smiling):** Yeah, they actually told me they got into it, and it wasn't a propeller head activity.
- **Doug (laughing):** I'll be sure to take off my techie beanie the next time I visit ... Look, Lisa, I think we may have a problem with getting all of the functionality for the home security function out by the dates your management is talking about. It's early, I know, but I've already been doing a little back of the envelope planning and....

- Lisa: We've got to have it by that date, Doug. What functionality are you talking about?
- **Doug:** I figure we can get full home security functionality out by the drop-dead date, but we'll have to delay Internet access till the second release.
- Lisa: Doug, it's the Internet access that gives *SafeHome* "gee whiz" appeal. We're going to build our entire marketing campaign around it. We've gotta have it!
- **Doug:** I understand your situation, I really do. The problem is that in order to give you Internet access, we'll need a fully secure Web site up and running. That takes time and people. We'll also have to build a lot of additional functionality into the first release . . . I don't think we can do it with the resources we've got.
- Lisa **(frowning):** I see, but you've got to figure out a way to get it done. It's pivotal to home security functions and to other functions as well ... the other

- functions can wait until the next releases . . . I'll agree to that.
- Lisa and Doug appear to be at an impasse, and yet they must negotiate a solution to this problem. Can they both "win" here? Playing the role of a mediator, what would you suggest?

Developing Another Preliminary User Scenario (ch 8, pg 132)

- **The scene:**
  - A meeting room, during the second requirements gathering meeting.
- **The players:**
  - Jamie Lazar, software team member;
  - Ed Robbins, software team member;
  - Doug Miller, software engineering manager;
  - three members of marketing;
  - a product engineering representative;
  - a facilitator.
- **The conversation:**
- **Facilitator:** It's time that we begin talking about the
- *SafeHome* surveillance function. Let's develop a user scenario for access to the home security function.
- **Jamie:** Who plays the role of the actor on this?
- **Facilitator:** I think Meredith (a marketing person) has been working on that functionality. Why don't you play the role.
- **Meredith:** You want to do it the same way we did it last time, right?
- **Facilitator:** Right ... same way.
- **Meredith:** Well, obviously the reason for surveillance is to

allow the homeowner to check out the house while he or she is away, to record and play back video that is captured ... that sort of thing.

- Ed: Will the video be digital, and will it be stored on disk?
- Facilitator: Good question, but let's postpone implementation issues for now. Meredith?
- **Meredith:** Okay, so basically there are two parts to the surveillance function ... the first configures the system including laying out a floor plan--we need tools to help the homeowner do this--and the second part is the actual surveillance function itself. Since the layout is part of the configuration activity, I'll focus on the surveillance function.
- Facilitator **(smiling):** Took the words right out of my mouth.
- **Meredith:** Um ... I want to gain access to the surveillance function either via the PC or via the Internet. My feeling is that the Internet access would be more frequently used. Anyway, I want to be able to display camera views on a PC and

control pan and zoom for a specific camera. I specify the camera by selecting it from the house floor plan. I want to selectively record camera output and replay camera output. I also want to be able to block access to one or more cameras with a specific password. And I want the option of seeing small windows that show views from all cameras and then be able to pick the one I want enlarged.

- **Jamie:** Those are called thumbnail views.
- **Meredith:** Okay, then I want thumbnail views from all the cameras. I also want the interface to the surveillance function to have the same look and feel as all other *SafeHome* interfaces. I want it to be intuitive, meaning I don't want to have to read a manual to use it.
- **Facilitator:** Good job, now, let's go into this function in a bit more detail....

Use-Case Template for Surveillance (ch 8, pg 136)

- Use-case:
  - Access camera surveillance--display camera views (ACS-DCV).
- Iteration:
  - 2, last modification: Jan 14 by V.Raman
- Primary actor:
  - Homeowner.
- Goal in context:
  - To view output of camera placed throughout the house from any remote location via the Internet.
- Preconditions:
  - System must be fully configured; appropriate user ID and passwords must be obtained.
- Trigger:
  - The homeowner decides to take a look inside the house while away.
- Scenario:
  1. The homeowner logs onto the *SafeHome Products* Web site.
  2. The homeowner enters his or her user ID.
  3. The homeowner enters two passwords (each at least eight characters in length).
  4. The system displays all major function buttons.
  5. The homeowner selects "surveillance" from the major function buttons.
  6. The homeowner selects "pick a camera."
  7. The system displays the floor plan of the house.
  8. The homeowner selects a camera icon from the floor plan.

9. The homeowner selects the "view" button.
10. The system displays a viewing window that is identified by the camera ID.
11. The system displays video output within the viewing window at one frame per second.

- Exceptions:
  1. ID or passwords are incorrect or not recognized—see use-case: "validate ID and passwords."
  2. Surveillance function not configured for this system--system displays appropriate error message; see use-case: "configure surveillance function."
  3. Homeowner selects "view thumbnail snapshots for all cameras"--see use-case: "view thumbnail snapshots for all cameras."
  4. A floor plan is not available or has not been configured--display appropriate error message and see use-case: "configure floor plan."
  5. An alarm condition is encountered--see use-case: "alarm condition encountered."
- Priority:

  Moderate priority, to be implemented after basic functions.
- When available: Third increment.
- Frequency of use: Infrequent.

- Channel to actor:

  Via PC-based browser and Internet connection to *SafeHome* Web site.

- Secondary actors:

  System administrator, cameras.

- Channels to secondary actors:

  1. System administrator: PC-based system
  2. Cameras: wireless connectivity

- Open issues:

  1. What mechanisms protect unauthorized use of this capability by employees of the company?
  2. Is security sufficient? Hacking into this feature would represent a major invasion of privacy.
  3. Will system response via the Internet be acceptable given the bandwidth required for camera views?
  4. Will we develop a capability to provide video at a higher frames-per-second rate when high bandwidth connections are available?

Class Models (ch 8, pg 142)

- **The scene:**
  - Ed's cubicle, as analysis modeling begins.
- **The players:**
  - Jamie, Vinod, Ed

    all members of the *SafeHome* software engineering team.
- **The conversation:**
- (Ed has been working to extract classes from the use-case template for **Access camera surveillance--display camera views**" [presented in an earlier sidebar in this chapter] and is presenting the classes he has extracted to his colleagues.)
- **Ed:** So when the homeowner wants to pick a camera, he or she has to pick it from a floor plan. I've defined a **FloorPlan** class. Here's the diagram.
- (They look at Figure 8.14.)
- **Jamie:** So **FloorPlan** is a class that is put together with walls that are composed of wall segments, doors and windows, and also cameras; that's what those labeled lines mean, right?
- **Ed:** Yeah, they're called "associations." One class is associated with another according to the associations

I've shown. [Associations are discussed in Section 8.7.5.]

- **Vinod:** So the actual floor plan is made up of walls and contains cameras and sensors that are placed within those walls. How does the floor plan know where to put those objects?
- **Ed:** It doesn't, but the other classes do. See the attributes under, say, **WallSegment,** which is used to build a wall. The wall segment has start and stop coordinates and the *draw ()* operation does the rest.
- **Jamie:** And the same goes for windows and doors. Looks like camera has a few extra attributes.
- **Ed:** Yeah, I need them to provide pan and zoom info.
- **Vinod:** I have a question. Why does the camera have an ID but the others don't?
- **Ed:** We'll need to identify each camera for display purposes.
- **Jamie:** Makes sense to me, but I do have a few more questions.

- (**Jamie** asks questions which result in minor modifications.)
- **Vinod:** Do you have CRC cards for each of the classes? If so, we ought to role play through them, just make sure nothing has been omitted.
- **Ed:**" I'm not quite sure how to do them.
- **Vinod:** It's not hard, and they really pay off. I'll show you.

CRC models (ch 8, pg 145)

- **The scene:**
  - Ed's cubicle, as analysis modeling continues.
- **The players:**
  - Vinod, Ed

    members of the *SafeHome* software engineering team.
- **The conversation:**
- (Vinod has decided to show Ed how to develop CRC cards by showing him an example.)
- **Vinod:** While you've been working on surveillance and Jamie has been tied up with security, I've been working on the home management function.
- **Ed:** What's the status of that? Marketing kept changing its mind.
- **Vinod:** Here's the first cut use-case for the whole function ... we've refined it a bit, but it should give you an overall view.
- **Use-case:** *SafeHome* home management function.
- **Narrative:** We want to use the home management interface on a **PC** or an Internet connection to control electronic devices that have wireless interface controllers. The system should allow me to turn specific lights

on and off, to control appliances that are connected to a wireless interface, to set my heating and air conditioning system to temperatures that I define. To do this, I want to select the devices from a floor plan of the house. Each device must be identified on the floor plan. As an optional feature, I want to control all audio-visual devices--audio, television, DVD, digital recorders, and so forth. With a single selection, I want to be able to set the entire house for various situations.

One is *home,* another is *away,* a third is *overnight travel,* and a fourth is *extended travel.* All of these situations will have settings that will be applied to all devices. In the *overnight travel* and *extended travel* states, the system should turn lights on and off at random intervals (to make it look like someone is home) and control the heating and air conditioning system. I should be able to override these settings via the Internet with appropriate password protection.

- **Ed:** The hardware guys have got all the wireless interfacing figured out?
- **Vinod (smiling):** They're working on it, say it's no biggy. Anyway, I extracted a bunch of classes for home management, and we can use one as an example. Let's use the **HomeManagementInterface** class.
- **Ed:** Okay . . . so the responsibilities are ... the attributes and operations for the class, and the collaborations are the classes that the responsibilities point to.

- **Vinod:** I thought you didn't understand CRC.
- **Ed:** Maybe a little, but go ahead.
- **Vinod:** So here's my class definition for **HomeManagementInterface.**
- **Attributes:**
- optionsPanel--provides info on buttons that enable user to select functionality
- situationPanel--provides info on buttons that enable user to select situation
- FloorPlan--same as surveillance object but this one displays devices

- devicelcons--info on icons representing lights, appliances, HVAC, etc.
- devicePanels--simulation of appliance or device control panel; allows control
- **Operations:**
  - *displayControl(), selectControl(), displaySituation(), selectSituation(), accessFloorplan(), selectDevicelcon(), displayDevicePanel(), accessDevicePanel(), . . .*
- **Class:**
  - HomeManagementInterface

- **Responsibility** **Collaborator**
  - displayControl **OptionsPanel** (class)
  - selectControl **OptionsPanel** (class)
  - displaySituation **SituationPanel** (class)
  - selectSituation **SituationPanel** (class)
  - accessFloorplan **FloorPlan** (class) ...
  - •
  - •
  - •

- **Ed:** So when the operation *accessFloorplan()* is invoked, it collaborates with the **FloorPlan** object just like the one we developed for surveillance. Wait, I have a description of it here. (They look at Figure 8.14.)
- **Vinod:** Exactly. And if we wanted to review the entire class model, we could start with this index card, then go to the collaborator's index card, and from there to one of the collaborator's collaborators, and so on.
- **Ed:** Good way to find omissions or errors.
- **Vinod:** Yep.

Design versus Coding (ch 9, pg 159)

- **The scene:**
  - Jamie's cubicle, as the team prepares to translate requirements into design.
- **The players:**
  - Vinod, Jamie, Ed

    all members of the *SafeHome* software engineering team.
- **The conversation:**
- **Jamie:** You know, Doug [the team manager] is obsessed with design. I gotta be honest, what I really love doing is coding. Give me C++ or Java, and I'm happy.
- **Ed:** Nah . . . you like to design.
- **Jamie:** You're not listening; coding is where it's at.
- **Vinod:** I think what Ed means is you don't really like coding; you like to design and express it in code. Code is the language you use to represent the design.
- **Jamie:** And what's wrong with that?
- **Vinod:** Level of abstraction.
- **Jamie:** Huh?
- **Ed:** A programming language is good for representing details like data structures and algorithms, but it's not so good for representing architecture or component-to-component collaboration . . . stuff like that.

- **Vinod:** And a screwed-up architecture can ruin even the best code.
- **Jamie (thinking for a minute):** So, you're saying that I can't represent architecture in code . . . that's not true.
- **Vinod:** You can certainly imply architecture in code, but in most programming languages, it's pretty difficult to get a quick, big-picture read on architecture by examining the code.
- **Ed:** And that's what we want before we begin coding.
- **Jamie:** Okay, maybe design and coding are different, but I still like coding better.

Design Concepts (ch 9, pg 168)

- **The scene:**
  - Vinod's cubicle, as design modeling begins.
- **The players:**
  - Vinod, Jamie, Ed

    members of the *SafeHome* software engineering team . Also, Shakira, a new member of the team.
- **The conversation:**
- (All four team members have just returned from a morning seminar, entitled "Applying Basic Design Concepts," offered by a local computer science professor.)

- **Vinod:** Did you get anything out of the seminar?
- **Ed:** Knew most of the stuff, but it's not a bad idea to hear it again, I suppose.
- **Jamie:** When I was an undergrad CS major, I never really understood why information hiding was as important as they say it is.
- **Vinod:** Because ... bottom line ... it's a technique for reducing error propagation in a program. Actually, functional independence also accomplishes the same thing.

- **Shakira:** I wasn't a CS grad, so a lot of the stuff the instructor mentioned is new to me. I can generate good code and fast. I don't see why this stuff is so important.
- **Jamie:** I've seen your work, Shak, and you know what, you do a lot of this stuff naturally ... that's why your designs and code work.
- **Shakira (smiling):** Well, I always do try to partition the code, keep it focused on one thing, keep interfaces simple and constrained, reuse code whenever I can that sort of thing.
- **Ed:** Modularity, functional independence, hiding, patterns ... see.
- **Jamie:** I still remember the very first programming course I took ... they taught us to refine the code iteratively.
- **Vinod:** Same thing can be applied to design, you know.
- **Ed:** The only concept I hadn't heard of before was "refactoring."
- **Shakira:** That's used in Extreme Programming, I think she said.

- **Ed:** Yep. It's not a whole lot different than refinement, only you do it after the design or code is completed. Kind of an optimization pass through the software, if you ask me.
- **Jamie:** Let's get back to *SafeHome* design. I think we should put these concepts on our review checklist as we develop the design model for *SafeHome.*
- **Vinod:** I agree. But as important, let's all commit to think about them as we develop the design.

Refining an Analysis Class into a Design Class (ch 9, pg 171)

- **The scene:**
  - Ed's cubicle, as design modeling continues.
- **The players:**
  - Vinod, Ed
    members of the *SafeHome* software engineering team.
- **The conversation:**
- (Ed is working on the **FloorPlan** class [see sidebar discussion in Section 8.7.4 and Figure 8.14] and has refined it for the design model.)
- Ed**:** So you remember the **FloorPlan** class, right? It's used as part of the surveillance and home management functions.
- Vinod **(nodding):** Yeah, I seem to recall that we used it as part of our CRC discussions for home management.
- Ed**:** We did. Anyway, I'm refining it for design. Want to show how we'll actually implement the **FloorPlan** class. My idea is to implement it as a set of linked lists [a specific data structure]. So ... I had to refine the analysis class **FloorPlan** (Figure 8.14) and, actually, sort of simplify it.

- **Vinod:** The analysis class showed only things in the problem domain, well, actually on the computer screen, that were visible to the end-user, right?
- **Ed:** Yep, but for the **FloorPlan** design class, I've got to add some things that are implementation specific. I needed to show that **FloorPlan** is an aggregation of segments--hence the **Segment** class--and that the **Segment** class is composed of lists for wall segments, windows, doors, and so on. The class **Camera** collaborates with **FloorPlan,** and obviously, there can be many cameras in the floor plan.
- **Vinod:** Phew, let's see a picture of this new **FloorPlan** design class.
- (Ed shows Vinod the drawing shown in Figure 9.3.)
- **Vinod:** Okay, I see what you're trying to do. This allows you to modify the floor plan easily because new items can be added or deleted to the list--the aggregation--without any problems.
- **Ed (nodding):** Yeah, I think it'll work. **Vinod:** So do I.

Choosing an Architectural Style (ch 10, pg 192)

- **The scene:**
  - Jamie's cubicle, as design modeling continues.
- **The players:**
  - Jamie, Ed

    members of the *SafeHome* software engineering team.
- **The conversation:**
- Ed **(frowning):** We've been modeling the security function using UML. . . you know classes, relationships,
- that sort of stuff. So I guess the object-oriented architecture' is the right way to go.
- Jamie**:** But . . . ?
- Ed**:** But . . . I have trouble visualizing what an object-oriented architecture is. I get the call and return architecture, sort of a conventional process hierarchy, but 00 .. I don't know. It seems sort of amorphous.
- Jamie **(smiling):** Amorphous, huh?
- Ed**:** Yeah . . . what I mean is I can't visualize a real structure, just design classes floating in space.
- Jamie**:** Well, that's not true. There are class hierarchies . . .

think of the hierarchy (aggregation) we did for the **FloorPlan** object [Figure 9.3]. An 00 architecture is a combination of that structure and the interconnections—you know, collaborations--between the classes. We can show it by fully describing the attributes and operations, the messaging that goes on, and the structure of the classes.

- **Ed:** I'm going to spend an hour mapping out a call and return architecture, then I'll go back and consider an 00 architecture.
- **Jamie:** Doug'll have no problem with that. He said that we should consider architectural alternatives. By the way, there's absolutely no reason why both of these architectures couldn't be used in combination with one another.
- **Ed:** Good. I'm on it.

Evaluating Architectural Decisions(ch 10, pg 194)

- **The scene:**
  - Jamie's cubicle, as design modeling continues.
- **The players:**
  - Jamie, Ed

    members of the *SafeHome* software engineering team.
- **The conversation:**
- **Ed:** I finished my call-return architectural model of the security function.
- **Jamie:** Great! Do you think it meets our needs?
- **Ed:** It doesn't introduce any unneeded features, so it seems to be economic.
- **Jamie:** How about visibility?
- **Ed:** Well, I understand the model and there's no problem implementing the security requirements needed for this product.
- **Jamie:** I get that you understand the architecture, but you may not be the programmer for this part of the project. I'm a little worried about spacing. This design may not be as modular as an object-oriented design.
- **Ed:** Maybe, but that may limit our ability to reuse some of our code when we have to create the web-based version of this *SafeHome*.

- **Jamie**: What about symmetry?
- **Ed**: Well, that's harder for me to assess. It seems to me the only place for symmetry in the security function is adding and deleting PIN information.
- **Jamie**: That will get more complicated when we add remote security features to the web-based product.
- **Ed**: That's true, I guess.
- [They both pause for a moment, pondering the architectural issues.]
- **Jamie**: *SafeHome* is a real-time system, so state transition and sequencing of events will be tough to predict.
- **Ed**: Yeah, but the emergent behavior of this system can be handled with a finite state model.
- **Jamie**: How?
- **Ed**: The mode can be implemented based on the call-return architecture. Interrupts can be handled easily in many programming languages.
- **Jamie**: Do you think we need to do the same kind of analysis for the object-oriented architecture we were initially considering?

- **Ed**: I suppose it might be a good idea, since architecture is hard to change once implementation starts.
- **Jamie**: It's also important for us to map the nonfunctional requirements besides security on top of these architectures to be sure they have been considered thoroughly.
- **Ed**: Also, true.

Architecture Assessment (ch 10, pg 202)

- **The scene:**
  - Doug Miller's office as architectural design modeling proceeds.
- **The players:**
  - Vinod, Jamie, Shakira, Ed
    members of the *SafeHome* software engineering team.
  - Doug Miller
    manager of the software engineering group.
- **The conversation:**
- **Doug:** I know you guys are deriving a couple of different architectures for the *SafeHome* product, and that's a good thing. I guess my question is, how are we going to choose the one that's best?
- **Ed:** I'm working on a call and return style, and then either Jamie or I are going to derive an 00 architecture.
- **Doug:** Okay, and how do we choose?
- **Shakira:** I took a course in design in my senior year, and I remember that there are a number of ways to do it.
- **Vinod:** There are, but they're a bit academic. Look, I think we

can do our assessment and choose the right one using use-cases and scenarios.

- **Doug**: Isn't that the same thing?
- **Vinod**: Not when you're talking about architectural assessment. We already have a complete set of use-cases. So we apply each to both architectures and see how the system reacts--how components and connectors work in the use-case context.
- **Ed**: That's a good idea. Makes sure we didn't leave anything out.
- **Vinod**: True, but it also tells us whether the architectural design is convoluted, whether the system has to twist itself into a pretzel to get the job done.
- **Jamie**: Scenarios aren't just another name for use-cases?
- **Vinod**: No, in this case a scenario implies something different.
- **Doug**: You're talking about a quality scenario or a change scenario, right?
- **Vinod**: Yes. What we do is go back to the stakeholders and

ask them how *SafeHome* is likely to change over the next, say, three years. You know, new versions, features, that sort of thing. We build a set of change scenarios. We also develop a set of quality scenarios that define the attributes we'd like to see in the software architecture.

- **Jamie**: And we apply them to the alternatives.
- **Vinod**: Exactly. The style that handles the use-cases and scenarios best is the one we choose.

The OCP in Action (ch 11, pg 213)

- **The scene:**
  - Vinod's cubicle.
- **The players:**
  - Vinod, Shakira

    members of the *SafeHome* software engineering team.
- **The conversation:**
- **Vinod:** I just got a call from Doug [the team manager]. He says marketing wants to add a new sensor.
- **Shakira (smirking):** Not again, jeez!
- **Vinod:** Yeah ... and you're not going to believe what these guys have come up with.
- **Shakira:** Amaze me.
- **Vinod (laughing):** They call it a doggie angst sensor.
- **Shakira:** Say what?
- **Vinod:** It's for people who leave their pets home in apartments or condos or houses that are close to one another. The dog starts to bark. The neighbor gets angry and complains. With this sensor, if the dog barks for more than, say, a minute, the sensor sets a special alarm mode that calls the owner on his or her cell phone.
- **Shakira:** You're kidding me, right?

- **Vinod:** Nope. Doug wants to know how much time it's going to take to add it to the security function.
- **Shakira (thinking a moment):** Not much ... look. [She shows Vinod Figure 11.4] We've isolated the actual sensor classes behind the **sensor** interface. As long as we have specs for the doggie sensor, adding it should be a piece of cake. Only thing I'll have to do is create an
- appropriate component ... uh, class, for it. No change to the **Detector** component at all.
- **Vinod:** So I'll tell Doug it's no big deal.
- **Shakira:** Knowing Doug, he'll keep us focused and not deliver the doggie thing until the next release.
- **Vinod:** That's not a bad thing, but can you implement now if he wants you to?
- **Shakira:** Yeah, the way we designed the interface lets me do it with no hassle.
- **Vinod (thinking a moment):** Have you ever heard of the "Open-Closed Principle"?
- **Shakira (shrugging):** Never heard of it.
- **Vinod (smiling):** Not a problem.

Cohesion in Action (ch 11, pg 217)

- **The scene:**
  - Jamie's cubicle.
- **The players:**
  - Jamie, Ed

    members of the *SafeHome* software engineering team who are working on the surveillance function.
- **The conversation:**
- **Ed:** I have a first-cut design of the camera component.
- **Jamie:** Wanna do a quick review?
- **Ed:** I guess ... but really, I'd like your input on something.
- (Jamie gestures for him to continue.)
- **Ed:** We originally defined five operations for **camera.** Look ... [shows Jamie the list]
  - *determineType()* tells me the type of camera.
  - *translateLocation()* allows me to move the camera around the floor plan.
  - *displayID()* gets the camera ID and displays it near the camera icon.
  - *displayView()* shows me the field of view of the camera graphically.
  - *displayZoom()* shows me the magnification of the camera graphically.
- **Ed:** I've designed each separately, and they're pretty simple operations. So I thought

it might be a good idea to combine all of the display operations into just one that's called displayCamera()--it'll show the ID, the view, and the zoom. Whaddaya think?

- Jamie **(grimacing):** Not sure that's such a good idea.
- Ed **(frowning):** Why? All of these little ops can cause headaches.
- Jamie**:** The problem with combining them is we lose cohesion. You know, the *displayCamera()* op won't be single-minded.
- Ed **(mildly exasperated):** So what? The whole thing will be less than 100 source lines, max. It'll be easier to implement, I think.
- Jamie**:** And what if marketing decides to change the way that we represent the view field?
- Ed**:** I'll just jump into the *displayCamera()* op and make the mod.
- Jamie**:** What about side effects?
- Ed**:** Whaddaya mean?
- Jamie**:** Well, say you make the change but inadvertently create a problem with the ID display.

- **Ed:** I wouldn't be that sloppy.
- **Jamie:** Maybe not, but what if some support person two years from now has to make the mod. He might not understand the op as well as you do and, who knows, he might be sloppy.
- **Ed:** So you're against it?
- **Jamie:** You're the designer . . . it's your decision . . . just be sure you understand the consequences of low cohesion.
- **Ed (thinking a moment):** Maybe we'll go with separate display ops.
- **Jamie:** Good decision.

Coupling in Action (ch 11, pg 219)

- **The scene:**
  - Shakira's cubicle.
- **The players:**
  - Vinod, Shakira

    members of the *SafeHome* software engineering team who are working on the security function.
- **The conversation:**
- **Shakira:** I had what I thought was a great idea ... then I thought about it a little, and it seemed like a not-sogreat idea. I finally rejected it, but I just thought I'd run it by you.
- **Vinod:** Sure, what's the idea?
- **Shakira:** Well, each of the sensors recognizes an alarm condition of some kind, right?
- **Vinod (smiling):** That's why we call them sensors, Shakira.
- **Shakira (exasperated):** Sarcasm, Vinod. You've got to work on your interpersonal skills.
- **Vinod:** You were saying?
- **Shakira:** Okay, anyway, I figured ... why not create an operation within each sensor object called *makeCall()* that would collaborate directly with the **OutgoingCall** component, well, with an interface to the **OutgoingCall** component.

- **Vinod (pensive):** You mean rather than having that collaboration occur out of a component like **ControlPanel** or something?
- **Shakira:** Yeah ... but then I said to myself, that means that every sensor object will be connected to the **OutgoingCall** component, and that means that it's indirectly coupled to the outside world and . . . well, I just thought it made things complicated.
- **Vinod:** I agree. In this case, it's a better idea to let the sensor interface pass info to the **ControlPanel** and let it initiate the outgoing call. Besides, different sensors might result in different phone numbers. You don't want the sensor to store that information because if it changes.
- **Shakira:** It just didn't feel right.
- **Vinod:** Design heuristics for coupling tell us it's not right.
- **Shakira:** Whatever . . .

Graphic Design (ch12, pg 237)

- **The scene:**
  - Doug Miller's office after the first web interface prototype review.
- **The players:**
  - Doug Miller
    SafeHome software engineering project manager;
  - Vinod Raman
    member of the *SafeHome* software engineering team.
- **The conversation:**
- **Doug:** What's your impression of new Web page design?
- **Vinod:** I like it, but more importantly, our customers like it.
- **Doug:** How mush help did you get from the graphic designer we borrowed from marketing?
- **Vinod:** A lot, actually. She has a great eye for page layout and suggested an awesome graphic theme for the pages. Much better than what we came up with on our own.
- **Doug:** That's good. Any issues?
- **Vinod:** We still have to create alternate pages to take accessibility issues into account for some of our visually impaired users. But we would have had to do that for any Web page design we had.
- **Doug:** Do we need graphic design help on the alternative pages as well?

- **Vinod**: Sure. The designer has a good understanding of usability and accessibility issues.
- **Doug**: OK, I'll ask marketing if we can borrow her a little longer.

Violating a UI "Golden Rule" (ch 12, pg 240)

- **The scene:**
  - Vinod's cubicle, as user interface design begins.
- **The players:**
  - Vinod, Jamie
    members of the *SafeHome* software engineering team.
- **The conversation:**
- **Jamie:** I've been thinking about the surveillance function interface.
- **Vinod (smiling):** Thinking is good.
- **Jamie:** I think maybe we can simplify matters some.
- **Vinod:** Meaning?
- **Jamie:** Well, what if we eliminate the floor plan entirely? It's flashy, but it's going to take serious development effort. Instead we just ask the user to specify the camera he wants to see and then display the video in a video window.
- **Vinod:** How does the homeowner remember how many cameras are set up and where they are?
- **Jamie (mildly irritated):** He's the homeowner, he should know.
- **Vinod:** But what if he doesn't?
- **Jamie:** He should.

- **Vinod:** That's not the point ... what if he forgets?
- **Jamie:** Uh, we could provide a list of operational cameras and their locations.
- **Vinod:** That's possible, but why should he have to ask for a list?
- **Jamie:** Okay, we provide the list whether he asks or not.
- **Vinod:** Better. At least he doesn't have to remember stuff that we can give him.
- **Jamie (thinking for a moment):** But you like the floor plan, don't you?
- **Vinod:** Uh huh.
- **Jamie:** Which one will marketing like, do you think?
- **Vinod:** You're kidding, right?
- **Jamie:** No.
- **Vinod:** Duh ... the one with the flash ... they love sexy product features ... they're not interested in which is easier to build.
- **Jamie (sighing):** Okay, maybe I'll prototype both.
- **Vinod:** Good idea ... then we let the customer decide.

Use-Cases for UI Design (ch 12, pg 247)

- **The scene:**
  - Vinod's cubicle, as user interface design continues.
- **The players:**
  - Vinod, Jamie

    members of the SafeHome software engineering team.
- **The conversation:**
- **Jamie:** I pinned down our marketing contact and had her write a use-case for the surveillance interface.
- **Vinod:** From who's point of view?
- **Jamie:** The home owner's, who else is there?
- **Vinod:** There's also the system administrator role. Even if it's the homeowner playing the role, it's a different point of view. The "administrator" sets the system up, configures stuff, lays out the floor plan, places the cameras ...
- **Jamie:** All I had marketing do was play the role of a homeowner who wants to see video.
- **Vinod:** That's okay. It's one of the major behaviors of the surveillance function interface. But we're going to have to

examine the system administration behavior as well.

- **Jamie (irritated):** You're right.
- (**Jamie** leaves to find the marketing person. She returns a few hours later.)
- **Jamie:** I was lucky. I found our marketing contact and we worked through the administrator use-case together. Basically, we're going to define "administration" as one function that's applicable to all other *SafeHome* functions. Here's what we came up with.
- (**Jamie** shows the informal use-case to **Vinod**.)
- **Informal use-case:** I want to be able to set or edit the system layout at any time. When I set up the system, I select an administration function. It asks me whether I want to do a new set-up, or whether I want to edit an existing set-up. If I select a new set-up, the system displays a drawing screen that will enable me to draw the floor plan onto a grid. There will be icons for walls, windows, and doors so that drawing is easy. I just

stretch the icons to their appropriate lengths. The system will display the lengths in feet or meters (I can select the measurement system). I can select from a library of sensors and cameras and place them on the floor plan. I get to label each, or the system will do automatic labeling. I can establish settings for sensors and cameras from appropriate menus. If I select edit, I can move sensors or cameras, add new ones or delete existing ones, edit the floor plan, and edit the setting for cameras and sensors. In every case, I expect the system to do consistency checking and to help me avoid mistakes.

- Vinod **(after reading the scenario):** Okay, there are probably some useful design patterns or reusable components for GUIs for drawing programs. I'll betcha 50 bucks we can implement some or most of the administrator interface using them.
- Jamie**:** Agreed. I'll check it out.

Interface Design Review (ch 12, pg 254)

- **The scene:**
  - Doug Miller's office.
- **The players:**
  - Doug Miller
    software engineering manager;
  - Vinod
    a member of the *SafeHome* software engineering team.
- **The conversation:**
- **Doug:** Vinod, have you and the team had a chance to review the **SafeHomeAssured.com** e-commerce interface prototype?
- **Vinod:** Yeah . . . we all went through it from a technical point of view, and I have a bunch of notes. I e-mailed 'em to Sharon [manager of the WebApp team for the outsourcing vendor for the *SafeHome* e-commerce website] yesterday.
- **Doug:** You and Sharon can get together and discuss the small stuff . . . give me a summary of the important issues.
- **Vinod:** Overall, they've done a good job, nothing ground breaking, but it's a typical e-commerce interface, decent aesthetics, reasonable layout, they've hit all the important functions . . .
- **Doug (smiling ruefully):** But?
- **Vinod:** Well, there are a few things. . . .

- **Doug:** Such as . . . ?
- **Vinod (showing Doug a sequence of story-boards for the interface prototype):** Here's the major functions menu that's displayed on the home page:

- **Learn about SafeHome.**
- **Describe your home.**
- **Get SafeHome component recommendations.**
- **Purchase a SafeHome system.**
- **Get technical support.**

The problem isn't with these functions. They're all okay, but the level of abstraction isn't right.

- **Doug:** They're all major functions, aren't they?
- **Vinod:** They are, but here's the thing . . . you can purchase a system by inputting a list of components . . . no real need to describe the house if you don't want to. I'd suggest only four menu options on the home page:

- **Learn about SafeHome.**
- **Specify the SafeHome system you need.**
- **Purchase a SafeHome system.**
- **Get technical support.**

When you select **Specify the SafeHome system you need**, you'll then have the following options:

- **Select SafeHome components.**
- **Get SafeHome component recommendations.**

If you're a knowledgeable user, you'll select components from a set of categorized pull-down menus for sensors, cameras, control panels, etc. If you need help, you'll ask for a recommendation and that will require that you describe your house. I think it's a bit more logical.

- **Doug:** I agree. Have you talked with Sharon about this?
- **Vinod:** No, I want to discuss this with marketing first; then I'll give her a call.

Formulating MobileApp Requirements (ch 13,pg 269)

- **The scene:**
  - A meeting room. The first meeting to identify requirements for a mobile version of the *SafeHome* WebApp.
- **The players:**
  - Jamie Lazar, software team member;
  - Vinod Raman, software team member;
  - Ed Robbins, software team member;
  - Doug Miller, software engineering manager;
  - three members of marketing;
  - a product engineering representative;
  - a facilitator.
- **The conversation:**
  - **Facilitator (pointing at whiteboard):** So that's the current list of objects and services for the home security function present in the WebApp.
  - **Vinod (interrupting):** My understanding is that people want *SafeHome* functionality to be accessible from mobile devices as well . . . including the home security function?
  - **Marketing person:** Yes, that's right . . . We'll have to add that functionality and try to make it context aware to help personalize the user experience.

- **Facilitator**: Context aware in what sense?
- **Marketing person**: People might want to use a smartphone instead of the control panel and avoid logging on to a website when they are in the driveway at home. Or they might not want all family members to have access to the master control dashboard for the system from their phones.
- **Facilitator**: Do you have specific mobile devices in mind?
- **Marketing person**: Well, all smartphones would be nice. We will have a Web version done, so won't the MobileApp run on all of them?
- **Jamie**: Not quite. If we took a mobile phone browser approach we might be able to reuse a lot of our WebApps. But remember, smartphone screen sizes vary and they may or may not all have the same touch capabilities. So at the very least we would have to create a mobile website that takes the device features into account.
- **Ed**: Perhaps we should build the mobile version of the website first.

- **Marketing person:** OK, but a mobile website solution wasn't what we had in mind.
- **Vinod:** Each mobile platform seems to have its own unique development environment too.
- **Production rep:** can we restrict MobileApp development to only one or two types of smartphones?
- **Marketing person:** I think that might work. Unless I'm mistaken, the smartphone market is dominated by two smartphone platforms right now.
- **Jamie:** There's also security to worry about. We better make sure on outsider can't hack into the system, disarm it, and rob the place or worse. Also a phone could get lost or stolen more easily than a laptop.
- **Doug:** Very true.
- **Marketing:** But we still need the same level of security . . . Just also be sure to stop an outsider from getting in with a stolen phone.
- **Ed:** That's easier said than done and . . .
- **Facilitator (interrupting):** Let's not worry about those details yet.

- (Doug, serving as the recorder for the meeting, makes an appropriate note.)
- **Facilitator:** As a starting point, can we identify which elements of WebApp security function are needed in the MobileApp and which will need to be newly created? Then we can decide how many mobile platforms we can support and when we can move forward on this project.
- (The group spends the next 20 minutes refining and expanding the details of the home security function.)

Applying Patterns (ch 14, pg 301)

- **The scene:**
  - Informal discussion during the design of a software increment that implements sensor control via the Internet for **SafeHomeAssured.com**
- **The players:**
  - Vinod responsible for design;
  - Jamie **SafeHomeAssured.com** chief system architect.
- **The conversation:**
- **Vinod:** So how is the design of the camera control interface coming along?
- **Jamie:** Not too bad. I've designed most of the capability to connect to the actual sensors without too many problems. I've also started thinking about the interface for the users to actually move, pan, and zoom the cameras from a remote Web page, but I'm not sure I've got it right yet.
- **Vinod:** What have you come up with?
- **Jamie:** Well, the requirements are that the camera control needs to be highly interactive—as the user moves the control, the camera should move as soon as possible. So, I was

thinking of having a set of buttons laid out like a normal camera, but when the user clicks them, it controls the camera.

- **Vinod:** Hmmm. Yeah, that would work, but I'm not sure it's right—for each click of a control you need to wait for the whole client-server communication to occur, and so you won't get a good sense of quick feedback.
- **Jamie:** That's what I thought—and why I wasn't very happy with the approach, but I'm not sure how else I might do it.
- **Vinod:** Well, why not just use the **InteractiveDeviceControl** pattern!
- **Jamie:** Uhmmm—what's that? I haven't heard of it?
- **Vinod:** It's basically a pattern for exactly the problem you are describing. The solution it proposes is basically to create a control connection to the server with the device, through which control commands can be sent. That way you don't need to send normal HTTP requests. And the pattern even shows how you can implement this using some simple AJAX techniques. You have some

simple client-side JavaScript that communicates directly with the server and sends the commands as soon as the user does anything.

- **Jamie**: Cool! That's just what I needed to solve this thing. Where do I find it?
- **Vinod**: It's available in an online repository. Here's the URL.
- **Jamie**: I'll go check it out.
- **Vinod**: Yep—but remember to check the consequences field for the pattern. I seem to remember that there was something in there about needing to be careful about issues of security. I think it might be because you are creating a separate control channel and so bypassing the normal Web security mechanisms.
- **Jamie**: Good point. I probably wouldn't have thought of that! Thanks.

Quality Issues (ch 15, pg 319)

- **The scene:**
  - Doug Miller's office as the *SafeHome* software project begins.
- **The players:**
  - Doug Miller
    manager of the *SafeHome* software engineering team;
  - Other members of the product software engineering team.
- **The conversation:**
- **Doug:** I was looking at an industry report on the costs of repairing software defects. They are pretty sobering.
- **Jamie:** We are already working on developing test cases for each functional requirement.
- **Doug:** That's good, but I was noticing that it costs eight times as much to repair a defect that is discovered in testing that it does if the defect is caught and reapired during coding.
- **Vinod:** We are using pairs programming so we should be able to catch most of the defects during coding.
- **Doug:** I think you are missing the point. Quality is more than simply removing coding errors. We need to look at the project quality goals and ensure that the evolving software products are meeting them.

- **Jamie**: Do you mean things like usability, security, and reliability?
- **Doug**: Yes, I do. We need to build checks into the software process to monitor our progress toward meeting our quality goals.
- **Vinod**: Can't we finish the first prototype and then check it for quality?
- **Doug**: I am afraid not. We must establish a culture of quality early in the project.
- **Vinod**: What do you want us to do Doug?
- **Doug**: I think we will need to find a technique that will allow us to monitor the quality of the *SafeHome* products. Let's think about this and revisit this again tomorrow.

Quality Issues (ch 16, pg 335)

- **The scene:**
  - Doug Miller's office as the *SafeHome* software project begins.
- **The players:**
  - Doug Miller
    manager of the *SafeHome* software engineering team;
  - Other members of the product software engineering team.
- **The conversation:**
- **Doug:** I know we didn't spend time developing a quality plan for this project, but we're already into it and we have to consider quality ... right?
- **Jamie:** Sure. We've already decided that as we develop the requirements model [Chapters 6 and 7], Ed has committed to develop a testing procedure for each requirement.
- **Doug:** That's really good, but we're not going to wait until testing to evaluate quality, are we?
- **Vinod:** No! Of course not. We've got reviews scheduled into the project plan for this software increment. We'll begin quality control with the reviews.
- **Jamie:** I'm a bit concerned that we won't have enough time to conduct all the reviews. In fact, I know we won't.
- **Doug:** Hmmm. So what do you propose?

- **Jamie**: I say we select those elements of the requirements and design model that are most critical to *SafeHome* and review them.
- **Vinod**: But what if we miss something in a part of the model we don't review?
- **Shakira**: I read something about a sampling technique [Section 15.6.4] that might help us target candidates for review. (Shakira explains the approach.)
- **Jamie**: Maybe ... but I'm not sure we even have time to sample every element of the models.
- **Vinod**: What do you want us to do, Doug?
- **Doug**: Let's steal something from Extreme Programming [Chapter 3]. We'll develop the elements of each model in pairs—two people—and conduct an informal review of each as we go. We'll then target "critical" elements for a more formal team review, but keep those reviews to a minimum. That way, everything gets looked at by more than one set of eyes, but we still maintain our delivery dates.
- **Jamie**: That means we're going to have to revise the schedule.
- **Doug**: So be it. Quality trumps schedule on this project.

Software Quality Assurance (ch 17, pg 345)

- **The scene:**
  - Doug Miller's office as the *SafeHome* software project begins.
- **The players:**
  - Doug Miller
    manager of the *SafeHome* software engineering team;
  - Other members of the product software engineering team.
- **The conversation:**
- **Doug:** How are things going with the informal reviews?
- **Jamie:** We're conducting informal reviews of the critical project elements in pairs as we code but before testing. It's going faster than I thought.
- **Doug:** That's good, but I want to have Bridget Thorton's SQA group conduct audits of our work products to ensure that we're following our processes and meeting our quality goals.
- **Vinod:** Aren't they already doing the bulk of the testing?
- **Doug:** Yet, they are. But QA is more than testing. We need to be sure that our documents are evolving along with out code and that we're making sure we don't introduce errors as we integrate new components.
- **Jamie:** I really don't want to be evaluated based on their findings.

- **Doug:** No worries. The audits are focuses on conformance of our work products to the requirements and process our activities. We'll only be using audit results to try to improve our processes as well as our software products.
- **Vinod:** I have to believe it's going to take more of our time.
- **Doug:** In the long run it will save us time when we find defects earlier. It also costs less to fix defects if they're caught early.
- **Jamie:** That sounds like a good thing then.
- **Doug:** It's also important to identify the activities where defects were introduced and add review tasks to catch them in the future.
- **Vinod:** That'll help us determine if we're sampling carefully enough with our review activities.
- **Doug:** I think SQA activities will make us a better team in the long run.

Preparing for Testing (ch 19, pg 377)

- **The scene:**
  - Doug Miller's office, as component-level design continues and construction of certain components begins.
- **The players:**
  - Doug Miller
    software engineering manager;
  - Vinod, Jamie, Ed, Shakira
    members of the *SafeHome* software engineering team.
- **The conversation:**
- **Doug:** It seems to me that we haven't spent enough time talking about testing.
- **Vinod:** True, but we've all been just a little busy. And besides, we have been thinking about it ... in fact, more than thinking.
- **Doug (smiling):** I know ... we're all overloaded, but we've still got to think down the line.
- **Shakira:** I like the idea of designing unit tests before I begin coding any of my components, so that's what I've been trying to do. I have a pretty big file of tests to run once code for my components is complete.

- **Doug:** That's an Extreme Programming [an agile software development process, see Chapter 4] concept, no?
- **Ed:** It is. Even though we're not using Extreme Programming per se, we decided that it would be a good idea to design unit tests before we build the component—the design gives us all of the information we need.
- **Jamie:** I've been doing the same thing.
- **Vinod:** And I've taken on the role of the integrator, so every time one of the guys passes a component to me, I'll integrate it and run a series of regression tests on the partially integrated program. I've been working to design a set of appropriate tests for each function in the system.
- **Doug (to Vinod):** How often will you run the tests?
- **Vinod:** Every day ... until the system is integrated ... well, I mean until the software increment we plan to deliver is integrated.
- **Doug:** You guys are way ahead of me!
- **Vinod (laughing):** Anticipation is everything in the software biz, Boss.

Designing Unique Tests (ch 19, pg 382)

- **The scene:**
  - Vinod's cubical.
- **The players:**
  - Vinod, Ed

    members of the *SafeHome* software engineering team.
- **The conversation:**
- **Vinod:** So these are the test cases you intend to run for the *password Validation* operation.
- **Ed:** Yeah, they should cover pretty much all possibilities for the kinds of passwords a user might enter.
- **Vinod:** So let's see ... you note that the correct password will be 8080, right?
- **Ed:** Uh huh.
- **Vinod:** And you specify passwords 1234 and 6789 to test for errors in recognizing invalid passwords?
- **Ed:** Right, and I also test passwords that are close to the correct password, see ... 8081 and 8180.
- **Vinod:** Those are okay, but I don't see much point in running both the 1234 and 6789 inputs. They're redundant . . . test the same thing, don't they?

- **Ed**: Well, they're different values.
- **Vinod**: That's true, but if 1234 doesn't uncover an error ... in other words ... the *password Validation* operation notes that it's an invalid password, it is not likely that 6789 will show us anything new.
- **Ed**: I see what you mean.
- **Vinod**: I'm not trying to be picky here ... it's just that we have limited time to do testing, so it's a good idea to run tests that have a high likelihood of finding new errors.
- **Ed**: Not a problem ... I'll give this a bit more thought.

Using Cyclomatic Complexity (ch 19, pg 386)

- **The scene:**
  - Shakira's cubicle.
- **The players:**
  - Vinod, Shakira

    members of the *SafeHome* software engineering team who are working on test planning for the security function.
- **The conversation:**
- **Shakira:** Look ... I know that we should unit test al! the components for the security function, but there are a lot of 'em and if you consider the number of operations that have to be exercised, I don't know ... maybe we should forget white-box testing, integrate everything, and start running black-box tests.
- **Vinod:** You figure we don't have enough time to do component tests, exercise the operations, and then integrate?
- **Shakira:** The deadline for the first increment is getting closer than I'd like ... yeah, I'm concerned.
- **Vinod:** Why don't you at least run white-box tests on the operations that are likely to be the most error prone?

- **Shakira (exasperated):** And exactly how do I know which are likely to be the most error prone?
- **Vinod:** V of *G*.
- **Shakira:** Huh?
- **Vinod:** Cyclomatic complexity--V of *G*. Just compute *V(G)* for each of the operations within each of the components and see which have the highest values for V(G). They're the ones that are most likely to be error prone.
- **Shakira:** And how do I compute *V* of *G*?
- **Vinod:** It's really easy. Here's a book that describes how to do it.
- **Shakira (leafing through the pages):** Okay, it doesn't look hard. I'll give it a try. The ops with the highest *V(G)* will be the candidates for white-box tests.
- **Vinod:** Just remember that there are no guarantees. A component with a low *V(G)* can still be error prone.
- **Shakira:** Alright. But at least this'll help me to narrow down the number of components that have to undergo white-box testing.

Class Testing (ch 19, pg 391)

- **The scene:**
  - Shakira's cubicle.
- **The players:**
  - Jamie, Shakira

    members of the SafeHome software engineering team who are working on test case design for the security function.
- **The conversation:**
- **Shakira:** I've developed some tests for the **Detector** class [Figure 11.4]--you know, the one that allows access to all of the **Sensor** objects for the security function. You familiar with it?
- **Jamie (laughing):** Sure, it's the one that allowed you to add the "doggie angst" sensor.
- **Shakira:** The one and only. Anyway, it has an interface with four ops: *read()*, *enable()*, *disable()*, and *test °*, Before a sensor can be read, it must be enabled. Once it's enabled, it can be read and tested. It can be disabled at any time, except if an alarm condition is being processed. So I defined a simple test sequence that will exercise its behavioral life history.

- (Shows Jamie the following sequence.)
  1. enable-test-read-disable
- **Jamie:** That'll work, but you've got to do more testing than that!
- **Shakira:** I know, I know. Here are some other sequences I've come up with.
- (She shows Jamie the following sequences.)
  2. enable-test-[read]*-test-disable
  3. [read]*
  4. enable-disable-[test | read]
- **Jamie:** So let me see if I understand the intent of these. #1 goes through a normal life history, sort of a conventional usage. #2 repeats the read operation n times, and that's a likely scenario. #3 tries to read the sensor before it's been enabled ... that should produce an error message of some kind, right? #4 enables and disables the sensor and then tries to read it. Isn't that the same as test #3?
- **Shakira:** Actually no. In #4, the sensor has been enabled. What #4 really tests is whether the disable op works as it should. A read() or test() after disable()

should generate the error message. If it doesn't, then we have an error in the disable op.

- **Jamie**: Cool. Just remember that the four tests have to be applied for every sensor type since all the ops may be subtly different depending on the type of sensor.
- **Shakira**: Not to worry. That's the plan.

Preparing for Validation (ch 20, pg 408)

- **The scene:**
  - Doug Miller's office, as component-level design continues and construction of certain components begins.
- **The players:**
  - Doug Miller
    software engineering manager,
  - Vinod, Jamie, Ed, Shakira
    members of the *SafeHome* software engineering team.
- **The conversation:**
- **Doug:** The first increment will be ready for validation in what ... about three weeks?
- **Vinod:** That's about right. Integration is going well. We're smoke testing daily, finding some bugs but nothing we can't handle. So far, so good.
- **Doug:** Talk to me about validation.
- **Shakira:** Well, we'll use all of the use-cases as the basis for our test design. I haven't started yet, but I'll be developing tests for all of the use-cases that I've been responsible for.
- **Ed:** Same here.
- **Jamie:** Me too, but we've got to get our act together for

acceptance testing and also for alpha and beta testing, no?

- **Doug:** Yes, In fact I've been thinking that we could bring in an outside contractor to help us with validation. I have the money in the budget ... and it would give us a new point of view.
- **Vinod:** I think we've got it under control.
- **Doug:** I'm sure you do, but an ITG gives us an independent look at the software.
- **Jamie:** We're tight on time here, Doug. I, for one, don't have the time to baby-sit anybody you bring in to do the job.
- **Doug:** I know, I know. But if an ITG works from requirements and use-cases, not too much baby sitting will be required.
- **Vinod:** I still think we've got it under control.
- **Doug:** I hear you, Vinod, but I'm going to overrule on this one. Let's plan to meet with the ITG rep later this week. Get 'em started and see what they come up with.
- **Vinod:** Okay, maybe it'll lighten the load a bit.

WebApp Testing (ch 21, pg 419)

- **The scene:**
  - Doug Miller's office.
- **The players:**
  - Doug Miller
    (manager of the *SafeHome* software engineering group)
  - Vinod Raman
    a member of the product software engineering team.
- **The conversation:**
- **Doug:** What do you think of the SafeHomeAssured.com e-commerce WebApp V0.0?
- **Vinod:** The outsourcing vendor's done a good job. Sharon [development manager for the vendor] tells me they're testing as we speak.
- **Doug:** I'd like you and the rest of the team to do a little informal testing on the e-commerce site.
- **Vinod (grimacing):** I thought we were going to hire a third-party testing company to validate the WebApp. We're still killing ourselves trying to get the product software out the door.
- **Doug:** We're going to hire a testing vendor for performance and security testing, and our outsourcing vendor is already testing. Just thought another

point of view would be helpful, and besides, we'd like to keep costs in line, so .. .

- **Vinod (sighs):** What are you looking for?
- **Doug:** I want to be sure that the interface and all navigation are solid.
- **Vinod**: I suppose we can start with the use-cases for each of the major interface functions:
  - **Learn about *SafeHome***
  - **Specify the *SafeHome* system you need Purchase a *SafeHome* system**
  - **Get technical support**
- **Doug:** Good. But take the navigation paths all the way to their conclusion.
- **Vinod (looking through a notebook of use-cases):** Yeah, when you select **Specify the *SafeHome* system you need,** that'll take you to:
  - **Select *SafeHome* components**
  - **Get *SafeHome* component recommendations**

  We can exercise the semantics of each path.
- **Doug:** While you're there, check out the content that appears at each navigation node.

- **Vinod:** Of course . . . and the functional elements as well. Who's testing usability?
- **Doug:** Oh… the testing vendor will coordinate usability testing. We've hired a market research firm to line up 20 typical users for the usability study, but if you guys uncover any usability issues ..
- **Vinod:** I know, pass them along.
- **Doug:** Thanks, Vinod.

SCM Issues (ch 22, pg 451)

- **The scene:**
  - Doug Miller's office as the *SafeHome* software project begins.
- **The players:**
  - Doug Miller (manager of the *SafeHome* software engineering team) ;
  - Vinod Raman, Jamie Lazar, other members of the product software engineering team.
- **The conversation:**
- **Doug:** I know it's early, but we've got to talk about change management.
- **Vinod (laughing):** Hardly. Marketing called this morning with a few "second thoughts." Nothing major, but it's just the beginning.
- **Jamie:** We've been pretty informal about change management on past projects.
- **Doug:** I know, but this is bigger and more visible, and as I recall ...
- **Vinod (nodding):** We got killed by uncontrolled changes on the home lighting control project ... remember the delays that ...
- **Doug (frowning):** A nightmare that I'd prefer not to relive.
- **Jamie:** So what do we do.

- **Doug:** As I see it, three things. First we have to develop--or borrow--a change control process.
- **Jamie:** You mean how people request changes?
- **Vinod:** Yeah, but also how we evaluate the change, decide when to do it (if that's what we decide), and how we keep records of what's affected by the change.
- **Doug:** Second, we've got to get a really good SCM tool for change and version control.
- **Jamie:** We can build a database for all of our work products.
- **Vinod:** They're called SCIs in this context, and most good tools provide some support for that.
- **Doug:** That's a good start, now we have to ...
- **Jamie:** Uh, Doug, you said there were three things
- **Doug (smiling):** Third--we've all got to commit to follow the change management process and use the tools--no matter what, okay?

Debating Product Metrics (ch 23, pg 464)

- **The scene:**
  - Vinod's cubicle.
- **The players:**
  - Vinod, Jamie, Ed

    members of the *SafeHome* software engineering team, who are continuing work on component-level design and test case design.
- **The conversation:**
- **Vinod:** Doug [Doug Miller, software engineering manager] told me that we should all use product metrics, but he was kind of vague. He also said that he wouldn't push the matter ... using them was up to us.
- **Jamie:** That's good, 'cause there's no way I have time to start measuring stuff. We're fighting to maintain the schedule as it is.
- **Ed:** I agree with Jamie. We're up against it, here ... no time.
- **Vinod:** Yeah, I know, but there's probably some merit to using them.
- **Jamie:** I'm not arguing that, Vinod. It's a time thing ... and I for one don't have any to spare.
- **Vinod:** But what if measuring saves you time?

- **Ed**: Wrong, it takes time and like Jamie said ...
- **Vinod**: No, wait ... what if it saves us time?
- **Jamie**: How?
- **Vinod**: Rework ... that's how. If a metric we use helps us avoid one major or even moderate problem, and that saves us from having to rework a part of the system, we save time. No?
- **Ed**: It's possible, I suppose, but can you guarantee that some product metric will help us find a problem?
- **Vinod**: Can you guarantee that it won't?
- **Jamie**: So what are you proposing?
- **Vinod**: I think we should select a few design metrics, probably class-oriented, and use them as part of our review process for every component we develop.
- **Ed**: I'm not real familiar with class-oriented metrics.
- **Vinod**: I'll spend some time checking them out and make a recommendation ... okay with you guys?
- (Ed and Jamie nod without much enthusiasm.)

Applying CK Metrics (ch 23, pg 471)

- **The scene:**
  - Vinod's cubicle.
- **The players:**
  - Vinod, Jamie, Shakira, Ed

    members of the *SafeHome* software engineering team, who are continuing work on component-level design and test case design.
- **The conversation:**
- **Vinod:** Did you guys get a chance to read the description of the CK metrics suite I sent you on Wednesday and make those measurements?
- **Shakira:** Wasn't too complicated. I went back to my UML class and sequence diagrams, like you suggested, and got rough counts for DIT, RFC, and LCOM. I couldn't find the CRC model, so I didn't count CBO.
- **Jamie (smiling):** You couldn't find the CRC model because I had it.
- **Shakira:** That's what I love about this team, superb communication.
- **Vinod:** I did my counts . . . did you guys develop numbers for the CK metrics?

- (Jamie and Ed nod in the affirmative.)
- **Jamie:** Since I had the CRC cards, I took a look at CBO, and it looked pretty uniform across most of the classes. There was one exception, which I noted.
- **Ed:** There are a few classes where RFC is pretty high, compared with the averages . . . maybe we should take a look at simplifying them.
- **Jamie:** Maybe yes, maybe no. I'm still concerned about time, and I don't want to fix stuff that isn't really broken.
- **Vinod:** I agree with that. Maybe we should look for classes that have bad numbers in at least two or more of the CK metrics. Kind of two strikes and you're modified.
- **Shakira (looking over Ed's list of classes with high RFC):** Look, see this class? It's got a high LCOM m well as a high RFC. Two strikes?
- **Vinod:** Yeah I think so . . . it'll be difficult to implement because of complexity and difficult to test for the same reason. Probably worth designing two separate classes to achieve the same behavior.
- **Jamie:** You think modifying it'll save us time?
- **Vinod:** Over the long haul, yes.

Establishing a Metrics Approach (ch 23, pg 479)

- **The scene:**
  - Doug Miller's office as the *SafeHome* software project is about to begin.
- **The players:**
  - Doug Miller (manager of the *SafeHome* software engineering team)
  - Vinod Raman, Jamie Lazar members of the product software engineering team.
- **The conversation:**
- **Doug:** Before we start work on this project, I'd like you guys to define and collect a set of simple metrics. To start, you'll have to define your goals.
- **Vinod (frowning):** We've never done that before, and ...
- **Jamie (interrupting):** And based on the timeline management has been talking about, we'll never have the time. What good are metrics anyway?
- **Doug (raising his hand to stop the onslaught):** Slow down and take a breath, guys. The fact that we've never done it before is all the more reason to start now, and the metrics work I'm talking about shouldn't take much time at all ... in fact, it just might save us time.
- **Vinod:** How?

- **Doug:** Look, we're going to be doing a lot more in-house software engineering as our products get more intelligent, become Web enabled, all that ... and we need to understand the process we use to build software ... and improve it so we can build software better. The only way to do that is to measure.
- **Jamie:** But we're under time pressure, Doug. I'm not in favor of more paper pushing ... we need the time to do our work, not collect data.
- **Doug (calmly):** Jamie, an engineer's work involves collecting data, evaluating it, and using the results to improve the product and the process. Am I wrong?
- **Jamie:** No, but ...
- **Doug:** What if we hold the number of measures we collect to no more than five or six and focus on quality?
- **Vinod:** No one can argue against high quality ...
- **Jamie:** True ... but, I don't know, I still think this isn't necessary.

- **Doug:** I'm going to ask you to humor me on this one. How much do you guys know about software metrics?
- **Jamie (looking at Vinod):** Not much.
- **Doug:** Here are some Web refs ... spend a few hours getting up to speed.
- **Jamie (smiling):** I thought you said this wouldn't take any time.
- **Doug:** Time you spend learning is never wasted ... go do it and then we'll establish some goals, ask a few questions, and define the metrics we need to collect.

Team Structure (ch 24, pg 496)

- **The scene:**
  - Doug Miller's office prior to the initiation of the *SafeHome* software project.
- **The players:**
  - Doug Miller (manager of the *SafeHome* software engineering team)
  - Vinod Raman, Jamie Lazar, other members of the product software engineering team.
- **The conversation:**
- **Doug:** Have you guys had a chance to look over the preliminary info on *SafeHome* that marketing's prepared?
- **Vinod (nodding and looking at his teammates):** Yes. But we have a bunch of questions.
- **Doug:** Let's hold on that for a moment. I'd like to talk about how we're going to structure the team, who's responsible for what. . . .
- **Jamie:** I'm really into the agile philosophy, Doug. I think we should be a self-organizing team.
- **Vinod:** I agree. Given the tight time line and some of the uncertainty, and that fact that we're all really competent [laughs], that seems like the right way to go.

- **Doug:** That's okay with me, but you guys know the drill.
- **Jamie (smiling and talking as if she were reciting something):** We make tactical decisions, about who does what and when, but it's our responsibility to get product out the door on time.
- **Vinod:** and with quality.
- **Doug:** Exactly. But remember there are constraints. Marketing defines the software increments to be produced--in consultation with us, of course.
- **Jamie:** And?
- **Doug:** And, we're going to use UML as our modeling approach.
- **Vinod:** But keep extraneous documentation to an absolute minimum.
- **Doug:** Who is the liaison with me?
- **Jamie:** We decided that Vinod will be the tech lead—he's got the most experience, so Vinod is your liaison, but feel free to talk to any of us.
- **Doug (laughing):** Don't worry, I will.

Estimating (ch 25,pg 513)

- **The scene:**
  - Doug Miller's office as project planning begins.
- **The players:**
  - Doug Miller (manager of the *SafeHome* software engineering team) ;
  - Vinod Raman, Jamie Lazar, other members of the product software engineering team.
- **The conversation:**
- **Doug:** We need to develop an effort estimate for the project, and then we've got to define a micro-schedule for the first increment and a macro schedule for the remaining increments.
- **Vinod (nodding):** Okay, but we haven't defined any increments yet.
- **Doug:** True, but that's why we need to estimate.
- **Jamie (frowning):** You want to know how long it's going to take us?
- **Doug:** Here's what I need. First, we need to functionally decompose the *SafeHome* software ... at a high level ... then we've got to estimate the number of lines of code that each function will take ... then ....

- **Jamie**: Whoa! How are we supposed to do that?
- **Vinod**: I've done it on past projects. You use use-cases, determine the functionality required to implement each, guesstimate the LOC count for each piece of the function. The best approach is to have everyone do it independently and then compare results.
- **Doug**: Or you can do a functional decomposition for the entire project.
- **Jamie**: But that'll take forever, and we've got to get started.
- **Vinod**: No ... it can be done in a few hours ... this morning, in fact.
- **Doug**: I agree ... we can't expect exactitude, just a ball-park idea of what the size of *SafeHome* will be.
- **Jamie**: I think we should just estimate effort ... that's all.
- **Doug**: We'll do that too. Then use both estimates as a cross check.
- **Vinod**: Let's go do it....

Tracking the Schedule (ch 25, pg 529)

- **The scene:**
  - Doug Miller's office, prior to the initiation of the *SafeHome* software project.
- **The players:**
  - Doug Miller (manager of the *SafeHome* software engineering team) ;
  - Vinod Raman, Jamie Lazar, other members of the product software engineering team.
- **The conversation:**
- **Doug (glancing at a Powerpoint slide):** The schedule for the first *SafeHome* increment seems reasonable, but we're going to have trouble tracking progress.
- **Vinod (a concerned look on his face):** Why? We have tasks scheduled on a daily basis, plenty of work products, and we've been sure that we're not over-allocating resources.
- **Doug:** All good, but how do we know when the analysis model for the first increment is complete?
- **Jamie:** Things are iterative, so that's difficult.
- **Doug:** I understand that, but ... well, for instance, take *analysis classes defined*. You indicated that as a milestone.

- **Vinod:** We have.
- **Doug:** Who makes that determination?
- **Jamie (aggravated):** They're done when they're done.
- **Doug:** That's not good enough, Jamie. We have to schedule FTRs [formal technical reviews, Chapter 26], and you haven't done that. The successful completion of a review on the analysis model, for instance, is a reasonable milestone. Understand?
- **Jamie (frowning):** Okay, back to the drawing board.
- **Doug:** It shouldn't take more than an hour to make the corrections ... everyone else can get started now.

Risk Analysis (ch 26, pg 541)

- **The scene:**
  - Doug Miller's office, prior to the initiation of the *SafeHome* software project.
- **The players:**
  - Doug Miller (manager of the *SafeHome* software engineering team) ;
  - Vinod Raman, Jamie Lazar, other members of the product software engineering team.
- **The conversation:**
- **Doug:** I'd like to spend some time brainstorming risks for the *SafeHome* project.
- **Jamie:** As in what can go wrong?
- **Doug:** Yep. Here are a few categories where things can go wrong. [He shows everyone the categories noted in the introduction to Section 25.3.]
- **Vinod:** Umm ... do you want us to just call them out,
- **Doug:** No here's what I thought we'd do. Everyone make a list of risks ... right now ...
- (Ten minutes pass; everyone is writing.)
- **Doug:** Okay, stop.
- **Jamie:** But I'm not done!

- **Doug:** That's okay. We'll revisit the list again. Now, for each item on your list, assign a percent likelihood that the risk will occur. Then, assign an impact to the project on a scale of 1 (minor) to 5 (catastrophic).
- **Vinod:** So if I think that the risk is a coin flip, I specify a 50 percent likelihood, and if I think it'll have a moderate project impact, I specify a 3, right?
- **Doug:** Exactly.
- (Five minutes pass; everyone is writing.)
- **Doug:** Okay, stop. Now we'll make a group list on the white board. I'll do the writing, we'll call out one entry from your list in round robin format.
- (Fifteen minutes pass; the list is created.)
- **Jamie (pointing at the board and laughing):** Vinod, that risk (pointing toward an entry on the board) is ridiculous. There's a higher likelihood that we'll all get hit by lightning. We should remove it.

- **Doug:** No, let's leave it for now. We consider all risks, no matter how weird. Later we'll winnow the list.
- **Jamie:** But we already have over 40 risks ... how on earth can we manage them all?
- **Doug:** We can't. That's why we'll define a cut-off after we sort these guys. I'll do that off-line, and we'll meet again tomorrow. For now, get back to work ... and in your spare time, think about any risks that we've missed.

Conclusion? (ch 30, pg 604)

- **The scene:**
  - Doug Miller's office.
- **The players:**
  - Doug Miller
    (manager of the *SafeHome* software engineering group) ;
  - Vinod Raman,
    a member of the product software engineering team.
- **The conversation:**
- **Doug:** I'm really pleased that we got it done without too much drama.
- **Vinod (sighing and learning back in his chair):** Yeah, but the project grew, didn't it.
- **Doug:** And you're surprised? When we started *SafeHome*, marketing thought a desktop app would do the trick and then . . .
- **Vinod (smiling):** And then, the Web and mobility took over.
- **Doug:** But we all learned a lot.
- **Vinod:** We did. The tech stuff was interesting, but the software engineering stuff is probably what allowed us to get it done close to schedule.
- **Doug:** Yeah, that and hard work by all of you guys. What are you seeing from customer support? How's quality in the field?

- **Vinod:** There are a few issues, but nothing really serious. We're on it. In fact, I gotta meet with Jamie on one of them in five minutes.
- **Doug:** Before you go . . .
- **Vinod (on his way out the door):** I know, more work, right?
- **Doug:** Engineering has developed a new sensor . . . Very high tech . . . We'll need to integrate it in *SafeHome* II.
- **Vinod:** *SafeHome* II?
- **Doug:** Yeah, *SafeHome* II. We'll begin planning next week.