

Business Model Tutorial

*Cecilie Hoffman
Business Analyst*

*“How can they possibly understand what we need when they
don't understand what we do?”*

-Manager of a credit approval department

*The audience for this discussion is software engineers who develop
software infrastructure and applications for business people.*

Table of Contents

CECILIE HOFFMAN BUSINESS ANALYST	1
BUSINESS MODELS	4
RECOGNIZING A MODEL	4
SELECTING A MODEL	4
WHAT HAPPENS WHEN THE MODEL IS WRONG	7
WHAT HAPPENS WHEN THE PROBLEM DEFINITION IS WRONG	8
MODELS ARE RE-USABLE	9
THE FALLACY OF GUT FEELINGS	9
THE BUSINESS MODEL IS THE REQUIREMENTS SET	10
CUSTOMER	11
SERVICE PROVIDER	11
GENERALIZING THE MODEL OF CUSTOMER AND PROVIDER	11
COMMON BUSINESS ACTIVITIES ACROSS INDUSTRIES	12
SUMMARY	12
THE FIRST END-TO-END THREAD	13
OVERVIEW	13
CASES	13
SUMMARY	15
DEFINING THE BUSINESS PROBLEM, OR, WHAT'S AN ARCHITECT GOOD FOR?	16
INTRODUCTION	16
CASES	16
WHAT IS A SCENARIO?	18
WHAT ARE THE USES OF A SCENARIO IN THE REAL WORLD?	18
WHAT IS THE PURPOSE OF A SCENARIO IN THE CONTEXT OF A SOFTWARE PROJECT?	18
WHY YOU WOULD WANT ONE?	19
WHAT IS "IN" AND "NOT IN" A SCENARIO?	20
WHAT DOES A SCENARIO LOOK LIKE?	22
SUMMARY	23
CONCLUSIONS	24
PHILOSOPHICAL VIEW	24

Overview

Software developers are a very intelligent, unique population that tends to favor technology based solutions to business problems. Because of our predisposition towards technology, we sometimes have trouble recognizing the entire business context of the problem that we have been asked to solve.

Software developers, like many doctors, are trained to “fix the problem”. Often, what everyone is focussed on is not the true problem; it is just a symptom. An approach that recognizes the *business context* of the problem is more likely to result in a successful project, rather than a cancelled project.

The big nightmare for a software development project is defining the business requirements. What are the requirements? How do we get them? How do we validate them? Often, software projects spend six months to over a year just gathering requirements.

Given a software development project for a business end-user community, a *business model* provides the primary leverage to identifying the correct solution. Furthermore, the business model can be easily validated, which provides some insurance to the business that they are going to get value for their investment.

What is a business model? We all have an innate understanding of models; but we need to learn how to match models to business situations. In the commercial world, models repeat themselves across industries; once we learn to recognize and use models, we become better software developers.

The process of determining a system architecture to support the business applications should be just common sense, but often appears to be black magic. Just as with other areas in life, common sense in business comes from years of experience. The “magic” is being able to unwind a highly compiled problem to expose the abstractions that provide problem-solving leverage. The “black” part of the magic is a combination of speed (which comes from experience) and intuition.

Here is an overview of the concepts discussed in this paper:

- ♦ Everybody is concerned about business requirements. Where do they come from, how do we get them in a timely manner, and how do we know they are right?
- ♦ Identify the right business model, and most of the business requirements are complete. Business models are the first leverage point.
- ♦ Validate the business model by running an end-to-end thread. Prove that we have a good understanding of the problem space.
- ♦ Separate the definition of the problem from the definition of the solution. The architect is the second leverage point.
- ♦ Build scenarios to clarify the expectations of the business community for how the software solution will fit into their day-to-day work habits. Scenarios are the third leverage point.

Business Models

Recognizing a Model

We all have models. Sometimes we think in terms of stereotypes, or we reason by analogy. As software designers and developers, we have strong discrimination abilities; we take pride in being able to categorize quickly and efficiently. The larger our mental library of stereotypes, the quicker we can determine an analogous situation, the greater potential we have to solve problems. Our experience allows us to continuously add to our library and create new associations to concepts that are familiar.

We are exposed to stereotypes early in life. Stereotypes become so ingrained that we can be unaware that we filtering our perceptions through a stereotype. Stereotypes often become cliché, such as in cowboy movies where the stranger rides into town on a dark horse, wearing black clothes and a black hat.

As we get older we gain more experience and perceive situations with greater clarity. We realize the limitations of stereotypes. We do not speak of refining a stereotype; we say, “break a stereotype.” However, we do *refine* a model. Based on our initial model we start making assumptions. Sometimes our assumptions are valid, sometimes not. When things turn out differently than we expect, we need to question our assumptions. That is, we need to question whether our model is wrong.

In the following discussion, we will walk through a detailed example of working with a simple model. Working with a model is not easy; but just providing an overview would misrepresent the painstaking process, so bear with us.

Selecting a Model

The usefulness of a model is in its predictive value. For example, in the paradigm for diagnosing faults in machinery, the “expert” has a model of normalcy. Expert diagnosticians use the symptoms to hypothesize about possible causes for the machine to be malfunctioning. In the diagnostics paradigm, we assess the situation and start with a hypothesis. Then we test the hypothesis, and, when the tests indicate a high-probability hypothesis, we determine a repair plan based on the hypothesis. When the situation is repaired, there may or may not be a follow-up plan.

We have all had the experience of walking into a dark room, flicking on the wall light switch and being surprised (annoyed) when the table-lamp light doesn't go on.

This type of problem solving is largely the ability to match the patterns of indicators, symptoms, and test results; to a list of candidate causes and repairs. This matching process is based upon a model of “normal function”. Above we listed the symptom, “the light did not go on when we turned on the wall switch”.

In order to diagnose the problem, we need a model of how a lamp normally functions:

- ♦ The bulb needs to in good working order, and the bulb needs to be properly seated in the lamp socket.

- ♦ The lamp switch needs to be in the proper position, and the lamp cord needs to be properly plugged in.
- ♦ The wall switch has to be in the correct position.
- ♦ There has to be electricity available from the outlet to the lamp.

We also need a set of tests and their associated repairs. This particular problem appears to be simple, and certainly does not require a planned approach to resolve. However, it can demonstrate the usefulness of models and the basic differences in different problem solving approaches. Consider the following problem solving scenario:

Step-1: The 99% case tells us that the light bulb is bad. Our first activity is to replace the light bulb.

While this appears to be a good first step, we are not done. We need to determine how long the light bulb has been operating. If it is burning out every week then we are seeing the symptom (burned out bulb) and not the problem.

Step-2: We inserted a new bulb and the light still did not go on. Now we actually have to begin to think about the problem. Now we need the model. Now we need the problem-solving checklist. Now we need a plan.

A review of the model allows us to divide our tests into chunks or layers. We may have more than one problem. We use the model to test our current environment. We resist speculating about the problem. We may be thinking, “what could be wrong?”, but we stay focussed and perform tests to see if our environment matches the model.

Test-1: We check to see that the lamp switch is in the “on” position.

Test-2: We check to see that the lamp cord is plugged into the correct wall socket.

Test-3: We check to see that the bulb is working. In our initial solution (step-1), we replaced the bulb with a new bulb, but the new bulb has never been tested. Therefore, we need to retrieve a bulb that is currently installed in another lamp, and working, and install it in our non-working lamp.

The Analytical Approach

Up to this point, our tests have not required any special tools or knowledge. According to our model, we need to verify that we have power to the lamp socket. There are two approaches to verifying power; (1) using an electrical tester to verify the volts, amps and resistance at the lamp socket, and (2) plug a working lamp into this wall socket.

Approach-1: Using an electrical testing tool we assume: (1) if there is the power to the lamp socket then we must have crossed our hands with the bulbs and never really place a good bulb in the lamp socket or, the bulb was never seated correctly or, (2) if there is no power to the lamp socket and there is power to the wall socket then the lamp must be faulty, or (3) if there is no power to the wall socket then we must continue our diagnoses.

Approach-2: Using a substitute for a calibrated tool we assume: (1) if the new lamp works then our original lamp **may** be faulty, or (2) if the new lamp does not work then there is no power to the wall socket.

Approach-1 is a pure test and will provide non-ambiguous results. Approach-2 is a “substitution model” test and by its nature has ambiguity. Approach-1 is preferred, but in some cases, the necessary tools may not be available. When “substitution” tests are used, we must extend our tests to eliminate the ambiguities. For example, notice that with the substitution test (approach-2, test-1) we concluded that the original lamp **might** be faulty. This is the same result we obtained with approach-1, test-1, where the actual problem was a faulty test with the light bulb replacement, not a faulty lamp. Continuing on we will assume we are now at the state where there is no power to the wall socket.

At this point, we can state clearly that the lamp will not work without power. A correlating test would be to plug our original lamp into the wall socket where our substitute lamp worked. This test will verify that our lamp and bulb are working correctly.

The problem of no power to the wall socket further escalates the issues of tools and knowledge. To further diagnose the problem we now need a different model. Our original model assumed power from the wall socket, the “lamp” model. We now need a model of the electrical wiring in the house:

- ♦ We must have power to the main circuit box for the house.
- ♦ The circuit breakers must all be on.
- ♦ The wall sockets and wall switches must be wired correctly.
- ♦ The electrical draw from any given circuit must not exceed the breakers maximum.

Since we have power to the other sockets in the house (see previous tests), we know that there is power to the main circuit box.

Test-1: Check the circuit breaker for the wall socket in question. We find that the breakers are not all labeled, but we check each and every breaker by turning it off and then on again. After completing this test, we must re-test the power at the wall socket; and it is still absent power.

Test-2: We remove the wall socket plate and check the wiring. It appears to be okay.

Test-3: We remove the wall switch plate and check the wiring. We find that the hot-wire from the switch to the wall socket is disconnected, never allowing the circuit to complete no matter what the position of the light switch.

We correct the light switch wiring problem and then reconnect the original lamp to the wall socket and finally solve our problem. Later we find that the painter's had removed the wall switch plate the previous day in order to replace the wall switch with a new one matching the new wall color; they forgot to reconnect the new switch.

We have described a scenario using a very analytical approach to problem solving. Contrast the analytical approach with the *just fix-it* approach.

The “Just Fix It” Approach

Thinking that we are good at problem solving, we make a hypothesis by going after the fault with the highest probability of being the cause. That hypothesis, i.e., the model that governs how we think about the problem, has built into it all of our assumptions. For example, “Of course the bulb itself is good, I just changed it yesterday.”

With a hypothesis in mind, we typically determine what the repair should be. When the repair works, we often assume our hypothesis is correct. This conclusion is premature, and could lead us to some unwarranted follow-up activities.

Actually, we need to clearly separate “repair” and “test.” Just because the repair action worked doesn't mean that, the hypothesis, our *model*, was correct. There are four possible situations:

Case A. Correct fault identification, repair works.

Case B. Incorrect fault identification, repair works.

Case C. Correct fault identification, repair doesn't work.

Case D. Incorrect fault identification, repair doesn't work.

You may be wondering, “Okay, cases A and B make sense, but why bother with cases C and D?” Case C is valuable because sometimes we get discouraged when a repair doesn't work, and we abandon our original hypothesis. If the hypothesis is correct, the correct action is perseverance, which can be hard to do when we lose faith. Case D is the degenerative case - we simply aren't on the right track. Now that we have a perspective, let's look at the problem-solving events for each case.

Here are the problem solving events in Case A:

Case A: Correct fault identification, repair works

Symptom:	light won't go on
Hypothesis:	loose bulb
Test:	step 1: flip lamp switch off
	step 2: tighten bulb
	step 3: flip lamp switch
Confirmation:	light goes on
Diagnostic conclusion:	loose bulb
Follow up:	none

Straightforward, right? As long as the problem is fixed, everything's okay, or is it?

What Happens when the Model is Wrong

What if the problem is apparently fixed, i.e., the repair action results in the light bulb turning “on,” but the original hypothesis was incorrect? Do we care? We should.

Case B: Incorrect fault identification, repair works

The hypothesis is that bulb is bad. We replace the bulb et voilà, light! We assume the bulb was bad. Since it was our last “back-up” bulb, we go out and buy a 6-pack of new bulbs.

Comment: In fact, the bulb was loose, and not broken. Certainly, we'll eventually use all the bulbs, but the capital expenditure at this time was unnecessary.

Case C: Correct fault identification, repair doesn't work

The hypothesis is that bulb is bad, but replacing it doesn't cause the light to turn on. A possible follow-up would be to conclude the lamp is broken and it's time to get a new one.

Comment: In fact, there are two faults, the lamp isn't plugged in all the way AND the bulb is bad. It was unnecessary to buy a new lamp.

Case D. Incorrect fault identification, repair doesn't work

This is classification for all the degenerative cases. Here's a somewhat extreme conclusion: Since the lamp didn't work when we turned on the wall switch, we hypothesis that there has been a power failure and we go to the circuit and flip all the breaker switches.

Comment: Unless we can see other evidence of a power failure - flashing 12:00 on the VCR for example, this power-failure hypothesis has low probability and isn't worth testing early in the hypothesize-test cycle of machine repair. Nevertheless, people jump to conclusions like that all the time.

Solving the “room is dark” problem is trivial, but it illustrates some fundamental principles:

- ♦ The choice of model in essence lays out the problem-solving plan. Realize that our internal model of a problem affects how we go about solving it.
- ♦ As we are following the plan, we may realize that we need to change our model.
- ♦ We don't need sophisticated testing equipment or methodologies; a great deal of the assumptions can be verified with simple tests.
- ♦ Once you have confirming test data for the model; persevere with the model. A common reason for project failure is giving up too soon.

What Happens when the Problem Definition is Wrong

Case B is an example of what can happen when we don't get the business model right. The business model is the first fundamental to get right. The next step is to understand how the business people perceive the problem.

In business, the reasons for wanting to make changes are usually related to profit and loss, distinguishing oneself from one's competition. For example, the perceived culprit may be the paperwork associated with fulfilling an order. The solution provider is told, “Get rid of those data entry clerks! Just have the users enter the orders and go from there. Let's become

a paper-less business!” However, is the paper management the real issue? Is it the only issue? What does “paperless” mean?

If we listen carefully, we can hear underlying problems:

- ♦ Employing data entry clerks costs money
- ♦ There's so much paper and it moves through so many departments that we lose the paper
- ♦ We have to generate and process the paper
- ♦ The information on the paper isn't legible

Why does the paperwork need to move through so many departments? Why is the information on the paper not legible? Without a clear model of how the business functions, we should not presume that paper management is the sole problem to be resolved.

In this case, blindly going forth and installing scanners, OCR systems, and workflow would be as foolish as taking antacids for continuing chest pain. Conversely, when the business model is correct, it will not only solve these problems, but it is likely to apply to other situations as well.

Models are Re-Usable

Here are some relatively common commercial business models:

- ♦ Customer Service - Help Desks
- ♦ Real Time Data Analysis - Heating and Cooling systems
- ♦ Static Data Analysis - Credit Worthiness
- ♦ Design - Integrated Circuit
- ♦ Policy and Regulatory Interpretation - Who qualifies for benefits?
- ♦ Order Management - Purchasing just about anything
- ♦ Routing and Scheduling - Transporting a package from point A to point B

For example, in the transportation industry we can assume that a large number of the business issues we will deal with will have routing and scheduling components. What we may not think about are the customer service and order management issues that are also part of the core of a successful business. Improving a company's information infrastructure requires that we perceive all aspects of the business, not just what has been presented to us as “the problem”.

We don't need to be a walking library of business models to build good business applications. Given a specific business activity, the objective is to recognize the general model and test it to validate that it will solve 80% of the problem. We must balance our intuition about the problem with an analytic approach and empirical test results.

The Fallacy of Gut Feelings

In the 1980's, the consumer banking business invested huge capital into Automated Teller Machines (ATMs) believing that they would be able to cut down on the operating costs

associated with branch offices. Recent data for a large bank shows that of all possible transactions that can be performed at an ATM, only 18% are actually being done at the ATM. The advantage of 24-hour banking has not been sufficient to alter people's habits. People have altered their habits for certain transactions such as cash withdrawal but not for others such as check deposits.

In the past two years, the authors have taken informal surveys of a unique population in Silicon Valley, software developers. Most software developers are quite proud of the fact that they either do their banking through ATMs. We will go to great lengths to avoid going into a bank. That's fine for software developers, but what about the majority of the consumer banking population? Most people prefer to go into the bank, stand in line, and talk with a human teller.

The bank still wants to find ways to cut cost and distinguish itself from its competition. Technology is still perceived as the most effective way to make a difference in business processes. Whom does the bank rely on to provide technology-based solutions? Software developers. What do we already know about the headset of software developers with respect to technology? Not only are we predisposed to use it; we will make an extra effort to use it. We do not represent the general population.

It is incorrect to assume that based on gut feeling alone, software developers could design and implement a system that would appeal to non-ATM users. In this situation, the business has made a decision based on other factors to go ahead with ATMs. Consider what has developed since the ATMs ... *Online* home banking services.

To be fair, the personal computer is becoming increasingly common in the homes of middle class Americans. Nevertheless, who is most likely to have home computing equipment? Software developers or people in the information service industries. What about the majority of the consumer banking population?

The Business Model is the Requirements Set

Although you may be thinking that the authors are vilifying software developers, that is not the goal. Our goal is to clarify the role of a business model. The business model provides the broad perspective necessary to identifying appropriate solutions. One of our principles is, *at some level of abstraction, everything is the same*. The question is, *which level of abstraction yields the leverage* for solving the problem?

A measure of success for a business solution is that it is adopted by a large percentage of the targeted community of their own accord.

In determining an appropriate business model from which to drive a solution, we need to consider the requirements, the trade-offs, and the time and money available. In our next discussion, we will address how the right business model yields 80% of the business requirements.

Consider the confusion and complexity of the following problem:

“Our Help Desk needs help. We established it so we could track how many requests we were really getting - we were just consolidating all the little projects that various departments had started. Now, the Help Desk people want to attend the Help Desk Institute conference each year - the Help Desk has gone from being a project to being a department. The trouble is, we still don't know if we've made things any better for our customers.”

- the Help Desk manager's manager

Companies put Help Desks in place when standard business processes break down. If things actually worked the way people think they should work, then people wouldn't need help. In fact, things often work in ways that were never planned for, and since the plan and the reality don't match, Help Desks spring up. We can easily imagine a situation in which employees (internal **customers**) can't get what they need or want, and frustrated corporate administration departments (**providers**) who are doing the best they can but are always perceived as being difficult to work with.

Now, let's set up a model of our innate understanding of “the customer?” How much do we know? Quite a lot, actually.

Customer

Let's take a moment to consider what we think should happen when we are the customer. We expect to receive goods or services of a certain quality, delivered quickly and politely. We expect to pay a reasonable amount. We do understand that sometimes unexpected things happen; but we don't want to be one who discovers that something is wrong. Ideally, if there is a delay, we want to be notified and told when we can expect delivery.

Still, in some cases, we will have to notify the provider that something is not right with what we have received. We expect that the provider will apologize, and make every effort to correct the situation as quickly as possible with no extra cost to us. If we are satisfied with the service, we are likely to be pleasant to work with, and we will tell our friends about how easy it was to get what we needed. If we aren't satisfied with the service, we might show our annoyance. Certainly, we will complain to our friends about the poor service.

Service Provider

What about the service provider? Standing in the provider's shoes, what do we expect? We know our job is to deliver goods or services. We expect to make a reasonable profit. We know that we must provide the service in a manner that will make customers happy so that they will come back to us. In addition, a little recognition wouldn't hurt at all; we want the customer to speak well of us.

Generalizing the Model of Customer and Provider

Think about buying a car, getting your computer repaired, or submitting receipts for health care reimbursement. In each case, it is likely that the person you are talking to face-to-face or on the telephone has a different view of the service than you do. In a software solution, both views must be addressed for the business activity to be conducted smoothly.

Common Business Activities across Industries

Looking across industry types, such as manufacturing, transportation, and financial services, we can see that most business enterprises need data analysis to assess the financial health of the company. They also need service management, policy and regulatory interpretation for corporate administration, and order management for purchasing.

Within the transportation industry, whether the business is expedited freight or trans-oceanic shipping, whether the goods travel by air, or on rubber wheels or steel wheels, consigned goods will all have business requirements focussed on the following data:

- ♦ who consigned the goods
- ♦ who the goods are to be delivered to and how quickly
- ♦ the commodity type
- ♦ the weight and volume
- ♦ the value of the goods
- ♦ whether or not the goods require special services such as refrigeration, and
- ♦ whether or not the goods are controlled or hazardous.

The provider will have business requirements related to the following no matter what the transportation method:

- ♦ Yield management - controls for profit margin
- ♦ Regulations on the transport of commodities
- ♦ Inter-state, inter-province, inter-national customs procedures
- ♦ What the competition is doing

We could do this same exercise for each individual industry, but our point is made.

The business model IS the requirements set.

Summary

At least, the business model is the initial requirements step. Using the business model, we then define user scenarios to communicate the fundamental system-user interactions to the users. We obtain the buy-in from the users by verifying that the model represents the customer and the provider in their correct roles; it is easy to confuse what the provider desires to provide with what the consumer actually wants.

It is unnecessary to do a six-month requirements collection exercise because we can identify a business model in less than a month. We can test the model in less than three months using simple substitution tests. The test of the business model is the implementation of the first end-to-end thread. Then, once we have run the first end-to-end thread, we have our first real set of reality-validated requirements. The reality may surprise our users. Hopefully, we won't be surprised

Let's look at an end-to-end thread. Then we will come back and discuss scenarios and the role of the architect.

The First End-to-End Thread

Overview

The purpose of an end-to-end thread is to validate our choice of business model and the implied requirements for the problem solution. Our example has nothing to do with software *per se*, and everything to do with common sense.

An early stage of design in a software project is the determination of the five most important threads. One of these threads will be done first. It is essentially a potential throwaway prototype. Software developers will recognize principles of *incremental build* and *rapid prototyping* in this story.

Cases

A while ago Shoko put a \$100,000 in a safe place in San Jose. She recently moved into a senior citizens' center in San Francisco and now she needs the money.

Case 1 - An Attempt at an End-to-End Thread

She calls a friend and asks him how much he would charge her to go pick up the money and bring it back to her new home in San Francisco. Ralph thinks to himself, "It will take me about an hour one-way on 101 if I drive after 7pm. I'll just ask for gas, a beer and pizza afterwards." Ralph tells Shoko he'll do it for fifty bucks.

"That's great!" she says. "Oh, did I tell you the money is in pennies?"

Ralph is momentarily shocked, he realizes his Honda Civic won't do the job. He does a series of calculations:

- ♦ For \$100,000 there will be 200,000 rolls of pennies
- ♦ Volume of one roll of pennies is about 2 cubic inches. The volume of 200,000 rolls of pennies is 400,000 cubic inches, or, 230 cubic feet
- ♦ One roll of pennies weighs 4.5 ounces. Therefore, for 2,000 rolls, the payload will be 56,250 pounds, or 28 tons.

This job is looking like it's going to be more trouble than he expected.

"Oh, did I tell you about the key? This key is for the front door, but it will be easier if you load from the back door. You may have a little difficulty with the back door, sometimes this key works on it and sometimes it doesn't. You may want to bring a crowbar with you. Oh, and, you may want to bring something to distract the dogs."

Ralph tells Shoko that he's really sorry, but he has a bad feeling about this. She had better find someone else.

Case 2 - A Second Attempt at an End-to-End Thread

Shoko calls another friend and asks her how much she would charge to go pick up the money and bring it back. Kuan-Yin has already heard the story from Ralph, but she's intrigued. What would it take to recover 1% of the money? By Kuan-Yin's calculations:

- ♦ The volume of 1% of 200,000 rolls of pennies is 2.3 cubic feet. The weight of 1% of 200,000 rolls of pennies is about 563 pounds.
- ♦ A 1/4 ton pickup would handle the job.

The cost estimate of the 1% recovery is as follows:

- ♦ Renting the pickup for 1 day is \$100.
- ♦ Gas for the round trip is \$30.
- ♦ Kuan-Yin has her own equipment for entering the building, no charge to the customer. Kuan-Yin's time, normally \$1000/day, will be pro bono for this trip because Shoko is an old friend.

The cost of the exploration will be about \$130. Expenses aside, Kuan-Yin is beginning to wonder if she's going about this in the right way. What does Shoko really need? Obviously she wants the \$100,000, but what is she going to do with 200,000 rolls of pennies.

And what about the dogs? If the money is ill-gotten goods, Kuan-Yin doesn't want any part of it.

Kuan-Yin asks Shoko directly and learns that the money was re-payment of a loan from a now-estranged family member who holds a long grudge against her. The loan is paid, but Shoko can't use the money. She never really needed it until now. As for the dogs, the building that the money is stored in is a property of Shoko's. There's a small apartment in the back - the tenant takes in stray dogs.

So, somehow the money needs to be moved to a resource who convert it into a convenient asset, like the cash form it's already in, but in a larger, paper denomination. What is Shoko expecting, sacks of \$100 bills that she can stuff in her mattress and sleep on?

Kuan-Yin will have to ask Shoko some more questions.

Case 3 - A Viable End-to-End Thread

Kuan-Yin decides she needs to be smarter about solving this problem. What is really feasible? More importantly, what is really needed?

Is it better to retrieve the pennies herself, or outsource the job? From a transportation point of view, it is big enough to require professional services.

Kuan-Yin decides she needs an architect, and contacts Jamie.

Jamie begins asking questions about the use of the money once it has been received. Jamie knows that she can solve the technical problem of retrieving the

money, but until she understands the “use case” she does not know what the end goal is.

Shoko says the money is not useful to her in its current form and its current location. She wants to invest most of the money and use some of the money for day to day expenses.

Jamie now has a clear picture of the problem and the objective. She then defines a scenario for Shoko to test her proposed solution.

Jamie contacts several banks and brokerages looking for the best deal for Shoko. Jamie is looking for the equivalent of a “cash management” account. One bank agrees to retrieve the money and place it in an investment account with check and debit card access. There will be no charge for the retrieval and coin processing as long as there is a minimum investment balance of \$50,000 for the next 12 months. Shoko can invest the money in any financial instruments available through the bank's investment service.

Jamie spent about 3 hours on the project, and bills Shoko her normal minimum 1-day rate of \$1,000.

Summary

What is the worst thing that could have happened to Ralph? He could have ignored the implications of the pennies, driven to San Jose, and found a fenced-in building with dogs guarding it. Most likely, he would have returned empty-handed. He'd have invested his own time and money just to discover that the job was bigger than expected. Had this been a software project Ralph would have disqualified himself as a solution provider.

Kuan-Yin, on the other hand, has learned from hard experience that if you don't know where you are going, it's best to map out the territory before you begin the expedition; and never trust someone else's map.

Let's return to our discussion of business models. **An end-to-end thread validates the business model.** Now, let's examine the problem description that we were given. Do we have a clear understanding of what is needed?

Let's take a look at how an architect goes about **problem definition**.

Defining the Business Problem, or, What's an Architect Good For?

The purpose of this discussion is to expose two common problems in software development: Defining the problem in terms of the solution, and, not understanding the role of the system architect.

Introduction

Once a business has determined that they (the customer) want to solve a particular problem, and they have identified a solution provider, the customer has to describe the problem. It is human nature for the customer to express requirements as solutions. It is the responsibility of the solution provider to educate the customer about the true nature of problem, and about the ramifications of the solution space. Often the availability of technology obscures the true problem.

In the example below, we describe three types of interactions between characters. A solution team typically consists of people who can serve the roles of architects, designers, analysts, and programmers. We have simplified the solution team by making the architect an amalgam of the four roles. Let's revisit the Pennies cases.

Cases

Case 1

Only the obvious attributes of the situation are examined, and, the problem data is accepted without asking any questions. Ralph must be a bit dim-witted to not be curious about why Shoko has \$100,000 tucked away in pennies. Ralph presumes that he has to provide the solution all by himself. When he realizes the job is a lot more than he expected he has one recourse, "no bid."

As an architect, Ralph is in the dark (uninformed).

Case 2

When Kuan-Yin begins to examine what it would take, she initially goes down the same uninformed path as Ralph. However, at least she addresses issues of scale to achieve an acceptable level of value.

With some reflection, Kuan-Yin sees a better solution, but she is still locked into Shoko's solution-oriented definition of the problem. A sack of hundred dollar bills stuffed in a mattress isn't a good solution.

As an architect, Kuan-Yin is not doing much conceptual block-busting.

Case 3

Jamie considers the nature of request:

“ How much will you charge me to go pick up the money from San Jose and bring it back to my new home in San Francisco..., and it's in pennies, and ... the key to the back door, and ... the dogs.”

Asking a few questions results in a less colorful but still unstructured statement of the problem:

“Get the \$100,000 in pennies from San Jose to a bank and provide me with a means to draw on the funds so I don't have to pay for groceries with a shopping cart full of pennies.”

Here's an even more structured version of the problem:

“Client has \$100,000 that is not accessible, not in a form that is easy to store safely, and not convenient for commerce. Make it accessible to her from a secure location so that she can conduct financial transactions via standard means.”

With a structured definition of the problem, she can go about it in a completely different way. Here are the issues she knows she must deal with:

Issue 1. That a shipment of many pennies weighs tons. She needs a transportation resource that can handle the job.

Issue 2. Counting that many pennies is going to be a big job for any coin processing service provider. She needs a coin processing resource that can handle the job, and it may not necessarily be a consumer bank.

Issue 3. Once the pennies are counted, then the amount must be converted to a monetary form that can be stored safely and easily accessed.

Issue 4. The dogs need to be kept safe during the transfer of the payload.

Issue 5. Shoko needs a way to manage the large sum of money.

Now both the tactical and the strategic attributes of situation have been addressed, not just the surface attributes.

Jamie has investigated a possible solution from an informed perspective.

However, what does this have to do with business models and business requirements?

Thus far, we have only seen two leverage points, the business model itself, and the expertise of the architect. Now it's time to look at the third leverage point, scenarios.

What Is A Scenario?

Here's a software domain definition:

Scenarios enable software developers to verify that we understand how the users expect to use system that we are about to build.

Here are two classic definitions of the term *scenario*:

a) An outline or synopsis of the plot of a drama, opera, etc., indicating scenes, characters, etc.

b) The outline of a motion picture, indicating the action in the order of its development, the scenes, the cast of characters and their appearances, etc.

What are the Uses of a Scenario in the real world?

A scenario serves the same purpose as a map. We know that it isn't possible to experience San Francisco by looking at a map of the city, but we can get an idea of the different business and residential neighborhoods in the city, and how to move across the city via highways or public transportation. As we all know, the map isn't the territory, however, the map provides a means to evaluate some aspects of the territory.

Let's say we have a friend who is a playwright. She has a concept in her head for a play, which is derived from a book. There's lots of description in the book about the thinking of the characters, who they are, how they came to be they way they are, and why they behave they way they do. In a play, this description must be represented to the audience without written words. The thoughts of the characters, while clear in the book, will be conveyed to the play audience explicitly through body language and implicitly through dialogue.

Our playwright needs to find financial backing and someone to produce the play. Although she perceives the performance in her head, she must distill it down to the essence so that the producer can quickly understand the concept and become emotionally involved. Asking the producer to sit through a play reading isn't appropriate at this early stage of development; the producer wants just enough information to determine whether go with the concept.

The book is a popular title; the producer most likely has read it. However, the playwright can't assume that the producer's understanding will be the same as hers. People can read the same book, or see the same movie, and each person will take away their own impressions.

She builds a scenario to convey "the big picture" of a play. The scenario will expose the major themes and characters that enable those themes to develop. A critical selling point is suggesting who might play the major characters. Imagine if Shirley Temple had been Dorothy in the Wizard of Oz.

What is the purpose of a scenario in the context of a software project?

In the context of software development projects, we have refined the common definition to be used for business purposes. Nevertheless, the fundamental definition of a scenario is still

applicable – **it's the example that everybody always gives when they want to make the point clear.**

The scenario isn't perfect solution, but it's better than many (most) of the popular methods of gathering requirements. Scenarios enable us to identify process and metrics at the time as we create the usage model. The process of formulating the scenario gets us to visualization (shared hallucination). Once we have the visualization, we still have to tease out the rest of the requirements.

Let's modify our definition of a scenario to be:

- a) *An unambiguous view of what the consumer wants*
- b) *An unambiguous view of what the consumer is doing*

What do we mean by an *unambiguous view*? An unambiguous view leaves no room for interpretation, i.e., the consumer either wants to accomplish something or not, the consumer either does things a certain way or doesn't. Earlier, we talked about how two people can read the same book, or see the same movie, and walk away with completely different impressions. An objective of the scenario is to minimize the ambiguity.

What do we mean by *consumer*? In the context of a software development project, a consumer is a person who has a specific business task to complete, and, using a software application facilitates completing that task. We write a scenario from the consumer's point of view so that the consumer can read the description and instantly relate to the events and activities as if she or he was physically involved with the task at this very moment.

When we tell a story, whether it be the background for a play or a movie, we describe a sequence of events. When we tell a story about accomplishing a business task, we specify the attributes and behaviors of the people involved as well as the task being accomplished. Sometime the task is a compilation of related tasks. When we talk with the consumer, we ask, "*What is it that you want to do?*" In a scenario, we articulate what *it* is, what *it* is not, and what *its* behaviors are.

A scenarist's *modus operandi* is objectivity and clarity in describing these attributes and behaviors. The more objective and clear the scenario, the easier it will be for the consumer to understand, relate to, and evaluate.

Why You Would Want One?

We talked about how a scenario captures the essence of "the big picture". For software development project purposes, a scenario is a means to bridge the gap between the business consumer's worldview and the software system designers' worldview.

In a scenario we describe a business task so that the consumer can recognize the task, "yes, that is what I do", or "yes, that is what I wish would happen". Our description is constructed as a sequence of deliberately chosen events so that we can measure the complexity, usefulness, and desirability of an application that would bring the scenario to life.

We often build scenarios in pairs or in multiples. Again, our objective is to obtain a basis for metrics. We often need to ascertain how big a change the consumer is asking for. A pair of

scenarios, one describing the current sequence of events, the other describing the accomplishment of the same task via the new or proposed sequence of events, gives us the ability to measure, and to quantify the *delta* between the current situation and the proposed new world.

Sometimes a business task is a complex combination of related tasks. In order to describe completely the global task, the scenarist often finds it necessary to write multiple scenarios. One scenario provides the context for the rest; however, each scenario can stand on its own for the purposes of validation by the consumer.

What is “In” and “not In” A Scenario?

Remember word problems?¹ Unlike the nice clean problem with just numbers and function symbols, the problem is articulated in text. Here are some examples:

Example 1

Andy has three times as many sweets as Bernice. If he gives Bernice six sweets, he will then have twice as many as Bernice has. How many sweets did they each have to start with?

Example 2

A train passes completely through a tunnel in 5 minutes. A second train, twice as long, passes through the tunnel in 6 minutes. If both trains are traveling at the same speed, 24km/hr, determine the length of the tunnel and the lengths of the trains. What is the distance traveled by the front while the train goes “through” the tunnel?

Example 3

If a man walks to the station at 3 mph he misses his train by 1 minute. If he runs at 6 mph he has 2 minutes to spare. Find the distance to the station.

Example 4

The Haynes Parkway is a 20-mile road along a river that has no stoplights. The speed limit is 40 mph; but cars typically travel at 45-50 mph or perhaps faster. However, a sheriff regularly patrols the parkway, so speeds in excess of 50 mph are rare. During traffic, it is hard to pass, and traffic backs up behind any car that is going slower than the “average” speed.

What is the average speed of cars on Haynes Parkway given the following information? A car enters one end of Haynes Parkway and drives at 40 mph. Other cars begin to arrive behind him, and because they are traveling faster than 40 mph, a line of cars traveling 40 mph begins to form.

¹ <http://forum.swarthmore.edu/dr.math/> is the source for these word problems.

If after seven miles there is a line of 14 cars, what is the average speed of cars before they must slow behind the line of cars? There were only two cars that entered Haynes Park at the same time.

In the first three examples, you were given just the information you needed to solve the problem and nothing more. The fourth example includes information that is interesting, but does not appear relevant to solving the problem. How can you be sure? It is much harder to solve a “noisy” word problem.

Using the same reasoning, a scenario should clearly separate what is and is not information relevant to the business task at hand. A good scenario is **structured**. Here are the solutions to the first three problems:

Example 1

Andy has three times as many sweets as Bernice. If he gives Bernice six sweets, he will then have twice as many as Bernice has. How many sweets did they each have to start with?

(1) A has three times as many sweets as B. If he gives B six sweets, he will then have twice as many as B then has. How many sweets did they each have to start with?

Let x = number of sweets that Bernice has initially; then $3x$ is the number that Andy has. If now Andy gives 6 sweets to Bernice then Andy has $3x-6$ sweets and Bernice has $x+6$ sweets. Now we are told that after this transfer, Andy has twice as many sweets as Bernice, so we can write down an equation to represent this fact, i.e.

$$\begin{aligned} 3x-6 &= 2(x+6) \\ 3x-6 &= 2x + 12 \\ 3x-2x &= 12 + 6 \\ x &= 18 \end{aligned}$$

So initially Bernice had 18 sweets and Andy had $3 \cdot 18 = 54$ sweets. After transfer Bernice has $18+6 = 24$, Andy has $54-6 = 48$, and 48 is twice 24.

Example 2

A train passes completely through a tunnel in 5 minutes. A second train, twice as long, passes through the tunnel in 6 minutes. If both trains are traveling at the same speed, 24km/hr, determine the length of the tunnel and the lengths of the trains. What is the distance traveled by the front while the train goes “through” the tunnel?

To answer this question we must first define what is meant by a train “passing through a tunnel.” Thus, in order to solve this, we will assume that the time it takes for the train to pass through the tunnel is the time interval measured from the instant the front of the train enters the tunnel to the instant the back of the train leaves the tunnel.

What is the distance traveled by the front while the train goes “through” the tunnel?

Answer: $T+L$, where L is the length of the train, and T is the length of the tunnel (this is because when the back leaves the tunnel, the front is L meters away from the back).

Therefore, the first train (length l) passes the tunnel (length T) in 5 minutes, at 24 km/hr. The distance covered (by the front) is $(24 \text{ km/hr}) \cdot (5 \text{ mins}) \cdot (1/60 \text{ hr/mins}) \cdot (1000 \text{ m/km}) = 2000 \text{ m}$

Therefore, $T + l = 2000 \text{ m}$

The second train (length $2 \cdot l$) passes the tunnel (length T) in 6 minutes, at 24 km/hr. The distance covered (by the front) is $(24 \text{ km/hr}) \cdot (6 \text{ mins}) \cdot (1/60 \text{ hr/mins}) \cdot (1000 \text{ m/km}) = 2400 \text{ m}$

Therefore, $T + 2l = 2400 \text{ m}$

Now, we have two equations in two unknowns,

$$\begin{aligned}T + l &= 2000 \\T + 2l &= 2400\end{aligned}$$

from which the solution follows (we get: $T = 1600$ m, $l = 400$ m)

Example 3

If a man walks to the station at 3 mph he misses his train by 1 minute. If he runs at 6 mph he has 2 minutes to spare. Find the distance to the station.

Let x = distance to the station. Now we know that the difference in TIME is 3 minutes between the two methods of travel, so we shall need to write down an equation involving TIME.

Time = Dist/Speed, so when he walks, time = $x/3$ (hours), when he runs, time = $x/6$ (hours)

Subtract these and equate to the time (in hours) between the two methods of travel.

$$\begin{aligned}x/3 - x/6 &= 3/60 \\x(2-1)/6 &= 3/60 \\x/6 &= 3/60 \\x &= 18/60 \\x &= 3/10 \text{ of a mile.}\end{aligned}$$

Therefore, the distance to the station is $3/10$ mile. You can check the result now by calculating the actual time to walk, and the time to run.

$$\begin{aligned}\text{Time to walk} &= (3/10)/3 = 1/10 \text{ hour} = 6 \text{ minutes} \\ \text{Time to run} &= (3/10)/6 = 1/20 \text{ hour} = 3 \text{ minutes} \\ \text{Difference} &= 3 \text{ minutes.}\end{aligned}$$

Part of why the fourth problem is hard to solve is the noise; the noise makes the problem **unstructured**. Here is the thought process that ensues when we try to filter out the noise:

“Well, is there really enough information to solve this problem? See, the number of cars that pile up behind Driver A isn't just a function of how fast they are going, it's also a function of how they're spaced along the parkway. Therefore, if seven cars arrive shortly after Car A, and they all travel at 41 mph, they might all pile up behind Car A. But if they arrive twenty minutes after Car A and they all travel at 41 mph, it's likely that none of them will pile up behind Car A. Therefore, it's heavily dependent on how those cars are spaced.

The information needed in order to make any conclusions is the time that the cars in the pile-up entered the road. Then we'd be able to look at the last car in the pile, and get a lower bound on how fast its driver was going (it would have to be going at least a certain speed if it wanted to make up the given distance in the given time).

Given this information, all we could conclude is that seven cars had gone faster than 40 mph.”

- Manager, Traffic Planning Department

What Does A Scenario Look Like?

There are two formats for a scenario, “simple English” and “highly structured”.

For an example of the simple English scenario, see the Appendix (in progress). Also the same Appendix, we discuss the benefit of having both the simple English and the highly structured formats for a scenario and we go into more detail about *how* to write scenarios.

Summary

So, let's wrap up scenarios:

- ♦ A scenario is the example that everybody always gives when they want to make the point clear.
- ♦ A scenario is a means to bridge the gap between the business consumer's worldview and the software system designers' worldview.
- ♦ A scenario shows how a user interacts with a system to accomplish a specific business activity.
- ♦ The information in a scenario should be relevant to the business task that the scenario is illuminating. Extraneous information can be confusing, and can lead to unhappy results.
- ♦ There are two formats for a scenario, "simple English" and "highly structured". The highly structured format often spawns a set of use cases. Scenarios can also be used by software developers to write Use Cases (see Erickson, Jacobsen, and Booch for further information on Use Cases). Use cases are a good tool for the software object modeling, but that is not the focus of our discussion here.
- ♦ A set of carefully selected and crafted scenarios is provides the critical validation of the business requirements.

Conclusions

Given a business problem, and software development project assigned to solve the problem:

- ♦ Identify the right business model, and most of the business requirements are complete. Business models are the first leverage point.
- ♦ Separate the definition of the problem from the definition of the solution. The architect is the second leverage point.
- ♦ Build scenarios to clarify the expectations of the business community for how the software solution will fit into their day-to-day work habits. Scenarios are the third leverage point.
- ♦ Software developers must acknowledge that by definition, only the business people know the business. However, software developers can hypothesize what the requirements are likely to be.
- ♦ Both the business and the software development team will feel more comfortable that they are on the right track when the business validates the first end-to-end thread. By validating the first end-to-end thread, the business model is also validated.

Philosophical View

The business problem that we're trying to solve must be a hard nut to crack; otherwise, you wouldn't have waded through twenty pages of stories. Let us conclude with two bits of philosophy:

There is nothing new in the commercial world; if you have seen one business framework, you can generalize from it.

At some level of abstraction, everything is the same. So, what level of abstraction yields the leverage to solve the problem?

The principles of *unification*, and *process decomposition* should be just standard tools in every software developer's repertoire. Use them wisely, and use them often.