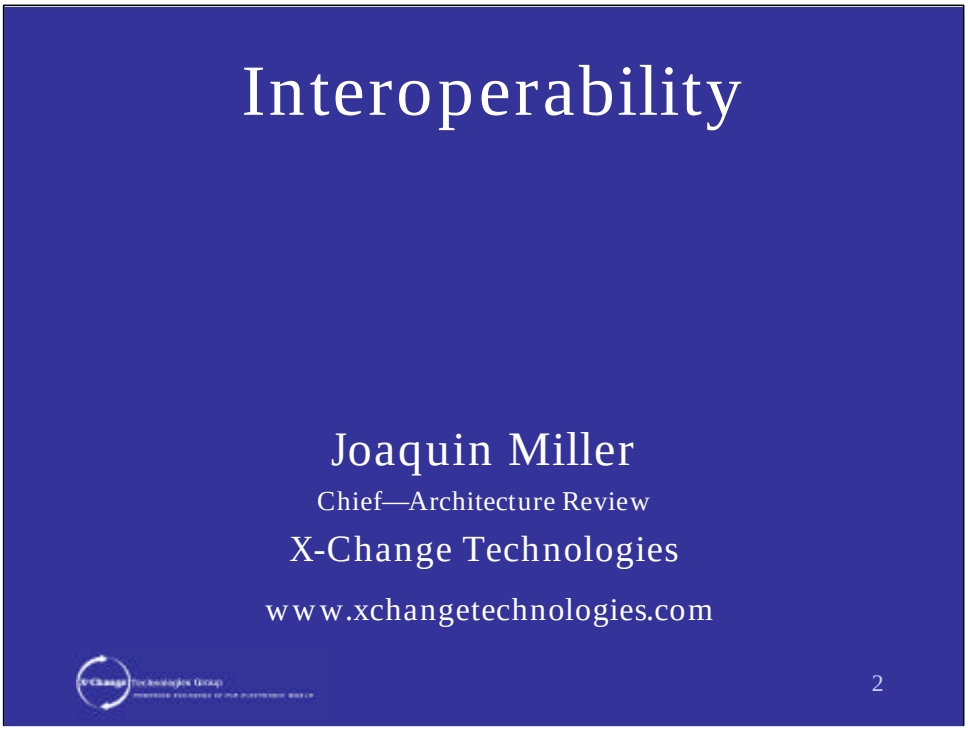
Interoperability

What does it mean for enterprises of emerging economic powers?

Joaquin Miller
Chief—Architecture Review
X-Change Technologies
www.xchangetechnologies.com



This presentation is about interoperability…

what it means…

And, in particular, what interoperability means for enterprises of emerging economic powers

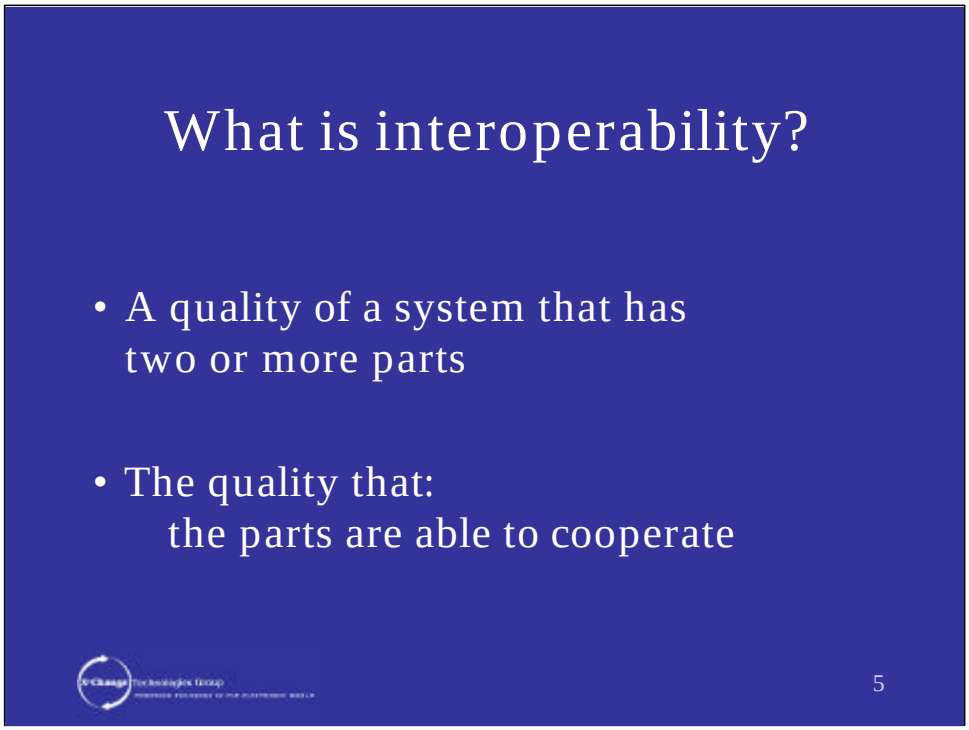
This is the question I will use to introduce the topic of this presentation. But the answers presented is not meant to tell you more about what the word, 'interoperability,' means in practice or in theory. Instead, they will be answers like:

— Why you better think about interoperability.

— The risks of not having it.

— How to get it.

— Decisions that can damage your business



This presentation is about interoperability. I am a software architect. Interoperability is what we call an architectural quality. It is a quality exhibited by software, which that software has because of its architecture.

Examples of architectural qualities of buildings:

— fire safety

— earthquake resistance

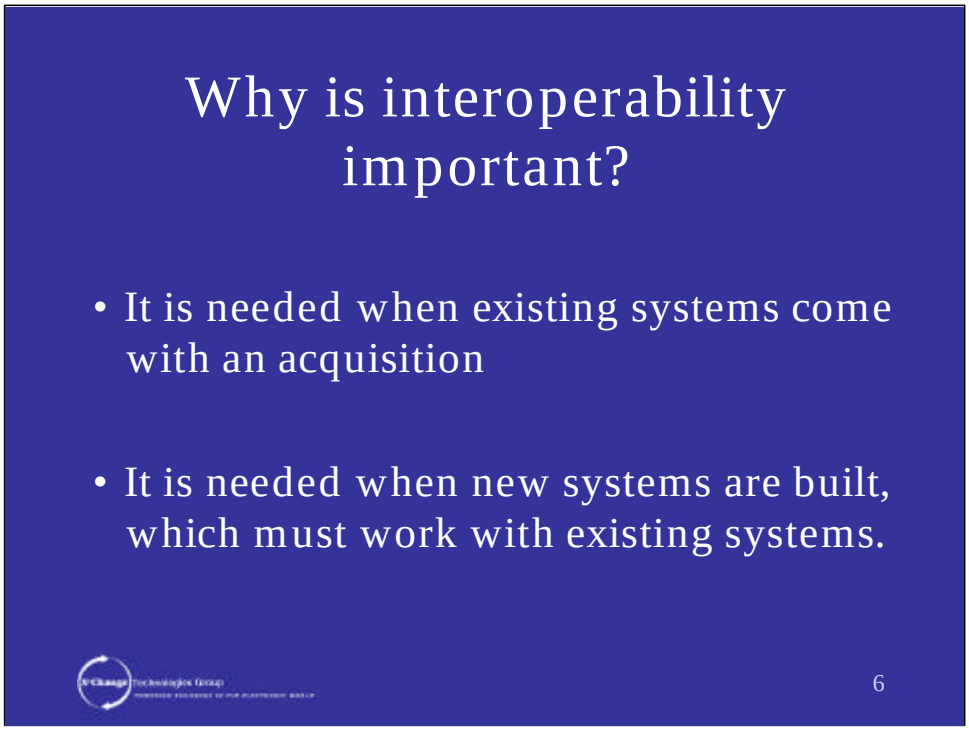
— fitness (to surroundings)

— the quality without a name

Interoperability means the quality that software systems can work together with other software systems, as part of a larger system of software.

Examples of some other qualities: correctness, completeness, usability, throughput, reliability, maintainability, location transparency, and transaction transparency.

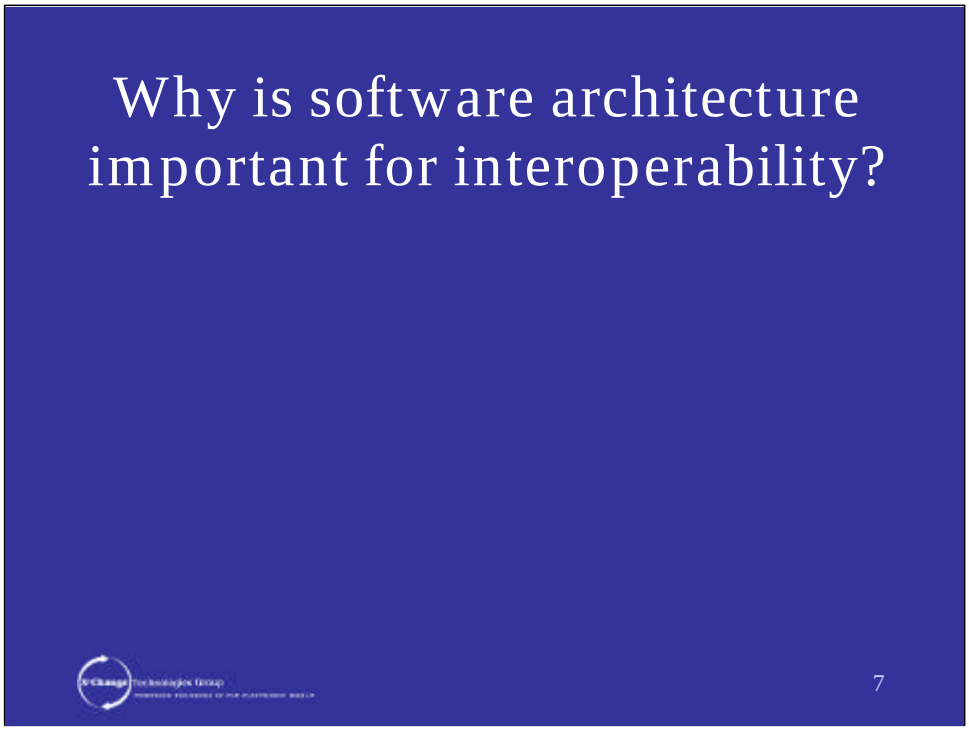
The architecture of an software system can contribute to or detract from any of these qualities. Many of them are impossible without an adequate architecture.



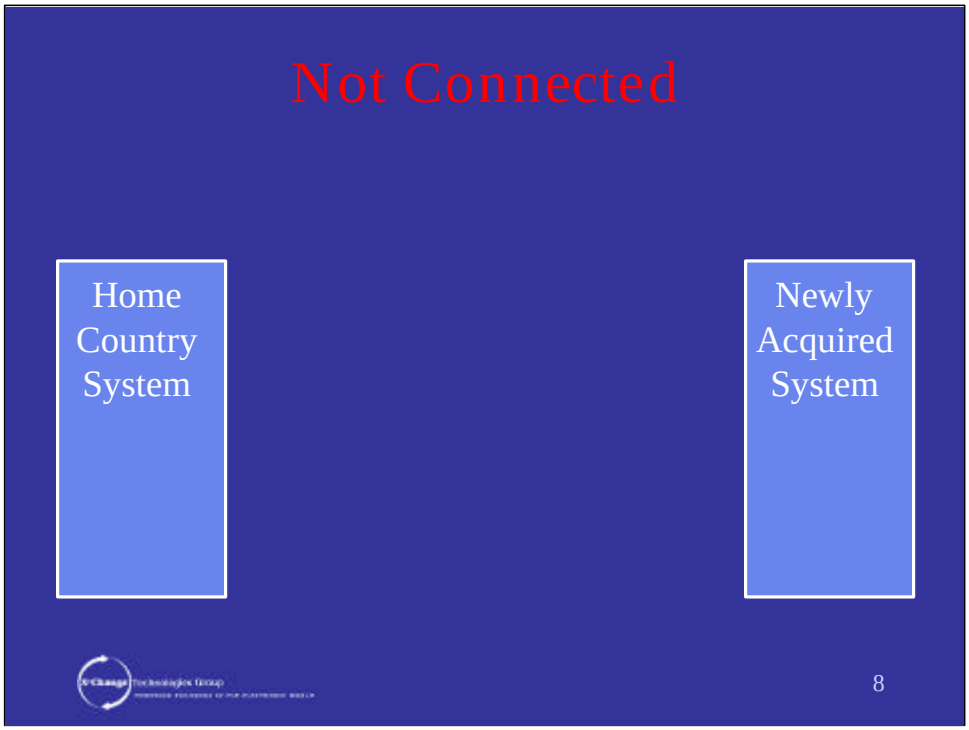
Why is interoperability important for expanding enterprises?

Because expansion is often by acquisition. And that means interoperating with the newly acquired systems.

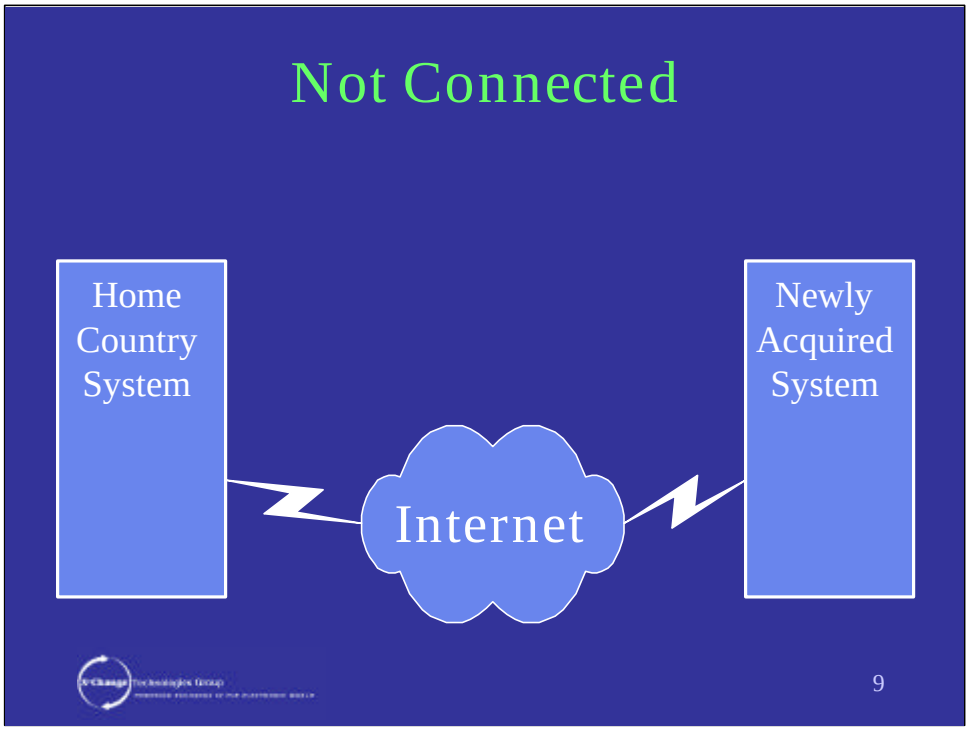
And because, even when expansion is not by acquisition, it results in the need to build new systems. That means interoperating with the systems already in use.



To answer this question, let’s look at some examples of barriers to interoperability, and an example solution for each problem.



Here we have a system in place in the home country, and a newly acquired system, perhaps in a different country. The two systems can't interoperate, because they are not connected.



Today, this barrier can be overcome using the internet. That is, it can be overcome if an internet connection is available in the location of each system. That is, the barrier can be overcome if both systems are enabled to communicate using the internet; it neither is limited to some communication protocol that does not work with the internet protocols.

This barrier is largely a problem of the past.

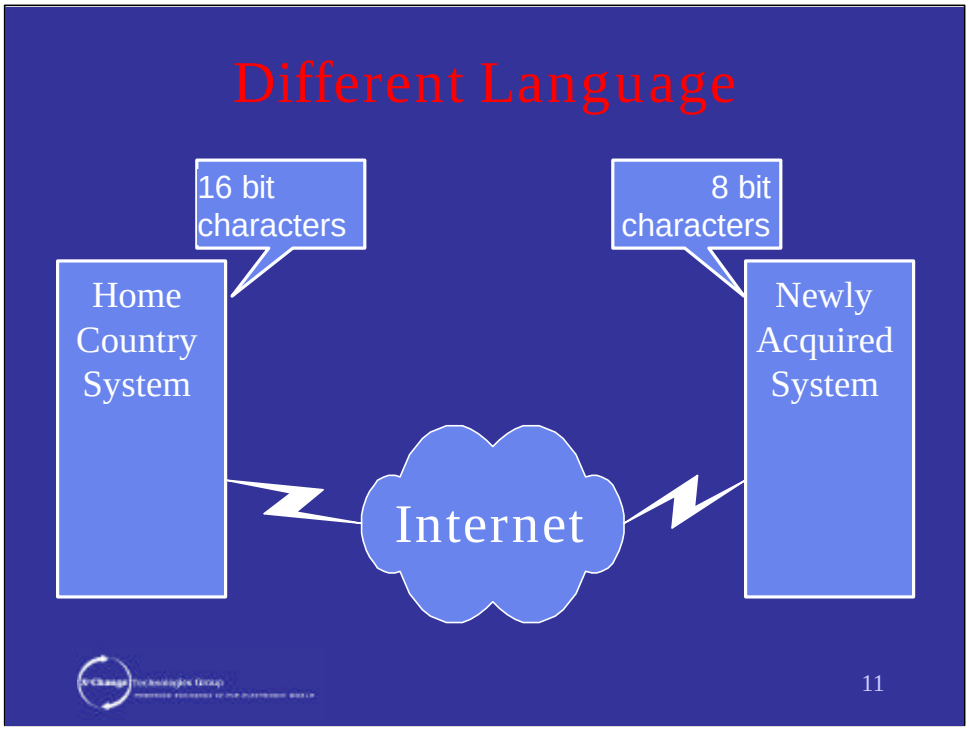
[My friend, Lee Felsenstein, recently installed pedal powered computers in villages in Laos, connected to the nearest internet access point by wireless links.]

Interworking

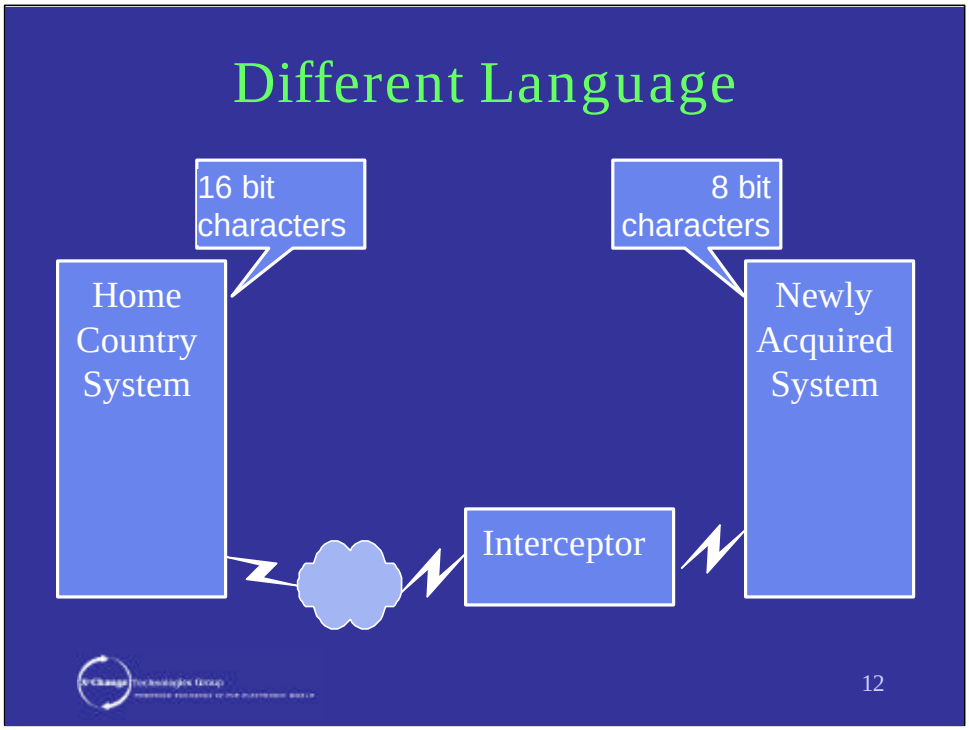
We have only provided the ability to exchange communications.
Nothing more

Architects don't call this interoperability.

It is called **interworking**. We'll see the use of this distinction later.

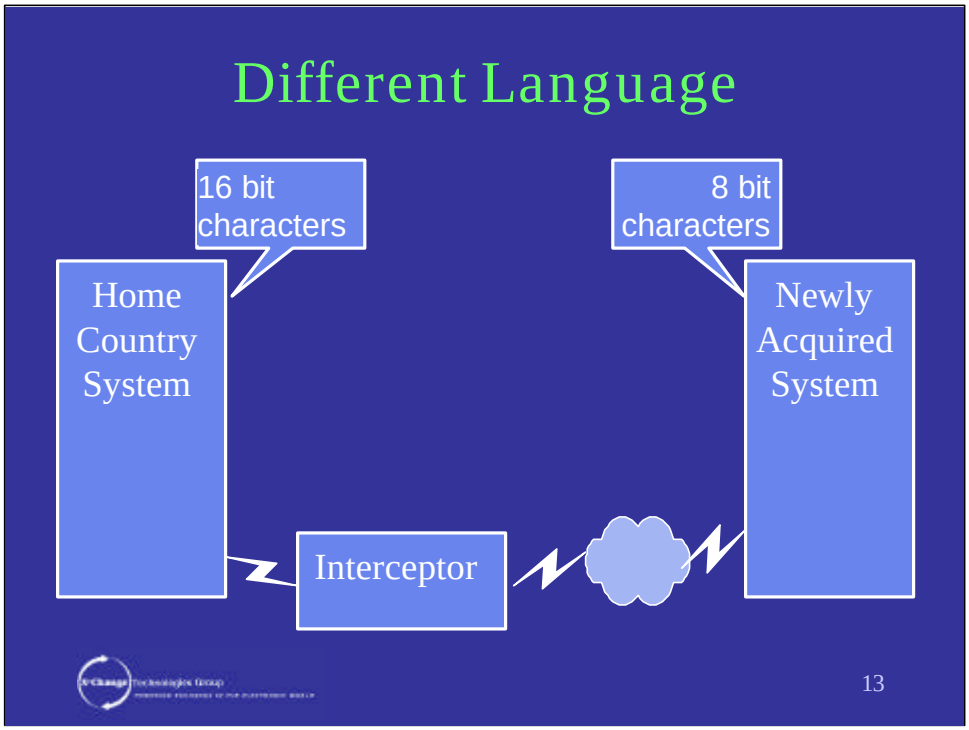


Another barrier is different languages. If one of your companies acquires a company in another country, you are very likely to encounter this barrier. One example of a language barrier is the representation of text. There is much software in place that can not handle sixteen bit text characters.

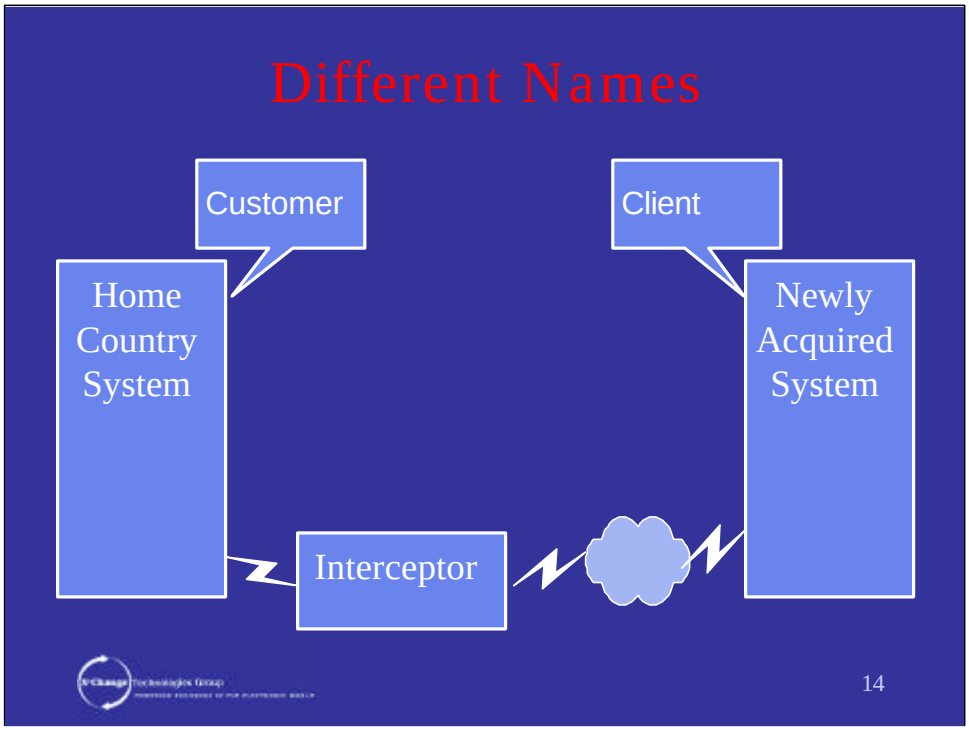


You can overcome this barrier by introducing what we software architects call an interceptor. This is a part of the system that does necessary conversions to enable interoperability. In this case, the interceptor converts each eight bit character to the corresponding character in the sixteen bit system. And, going in the other direction, converts some of the sixteen bit characters to the corresponding eight bit character. (Of course, most of the sixteen bit characters can't be handled in this way. But this will work in many situations, for example, if customer names are being transmitted between countries, and are always displayed in the original form.)

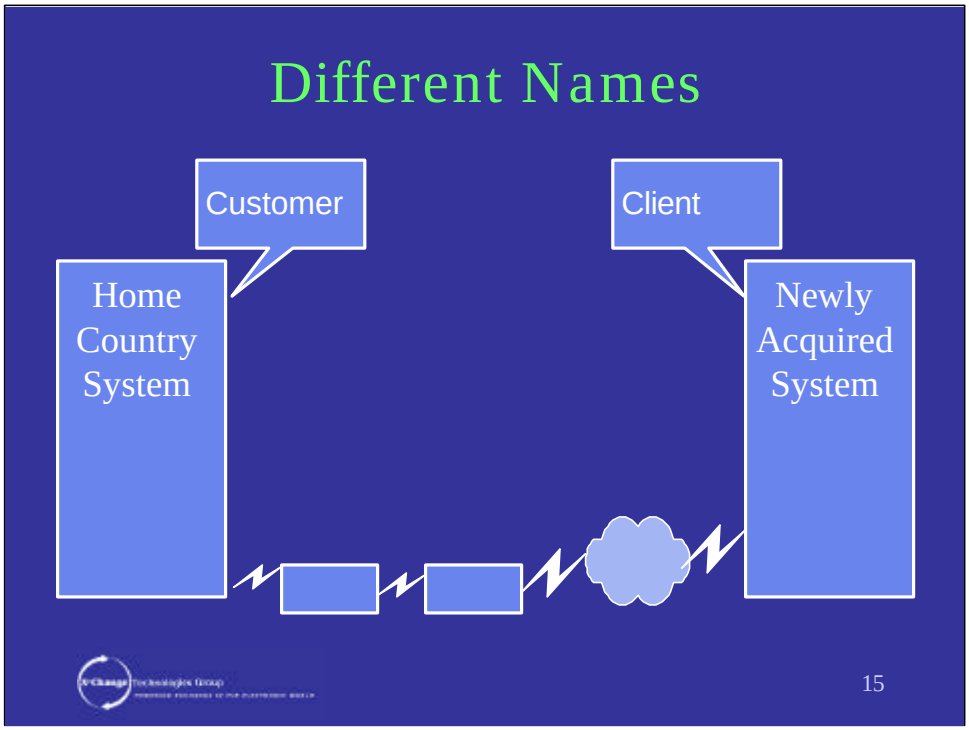
This is just one of many examples of different languages.



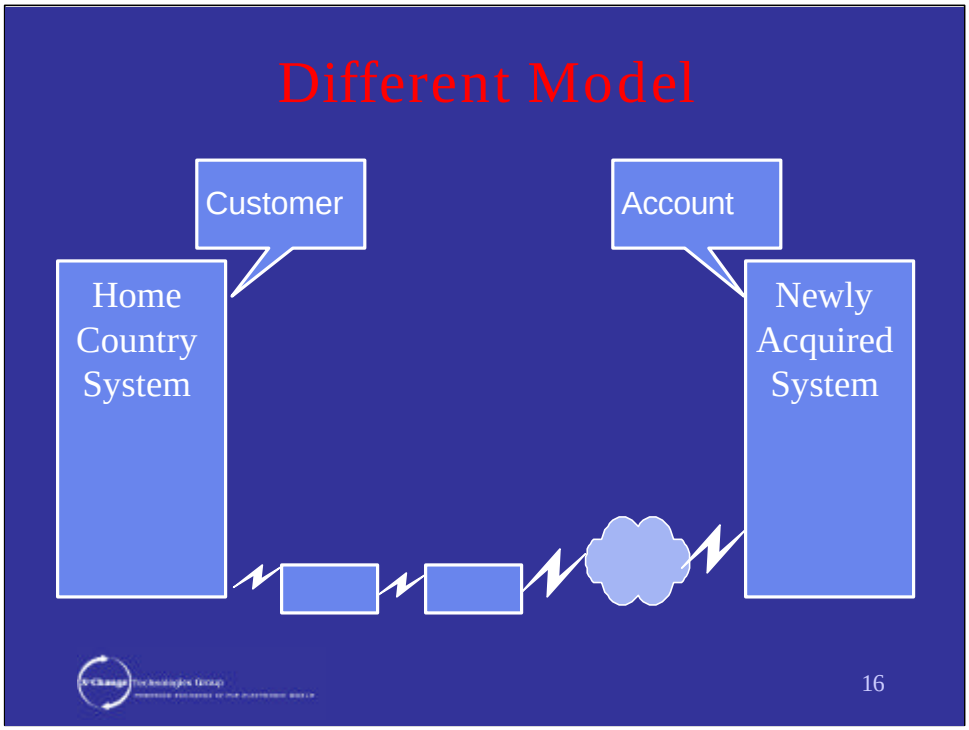
The placement of the interceptor may decided on architectural considerations or for other reasons. Perhaps there is a single, general purpose interceptor that handles several different language conversions. Or perhaps the interceptor is placed by the home country system, simply so it can more easily be managed at the home data center.



It is very likely that the a data element with one name in the home country system will have different a name in an acquired system.



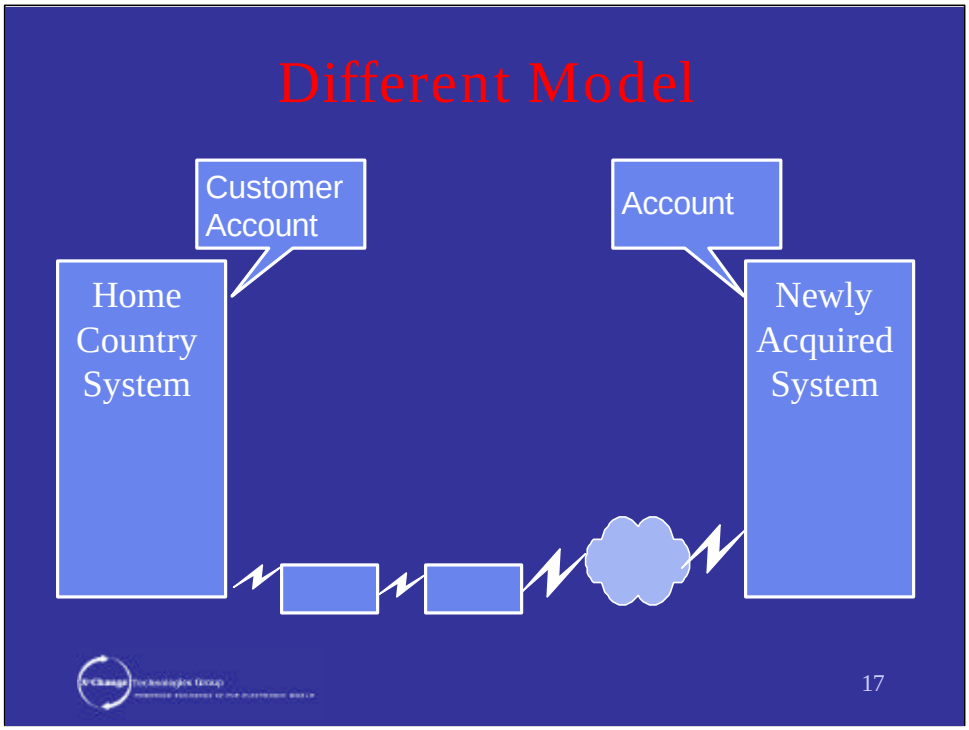
This barrier can be overcome with a second interceptor to convert the data names that are being communicated.



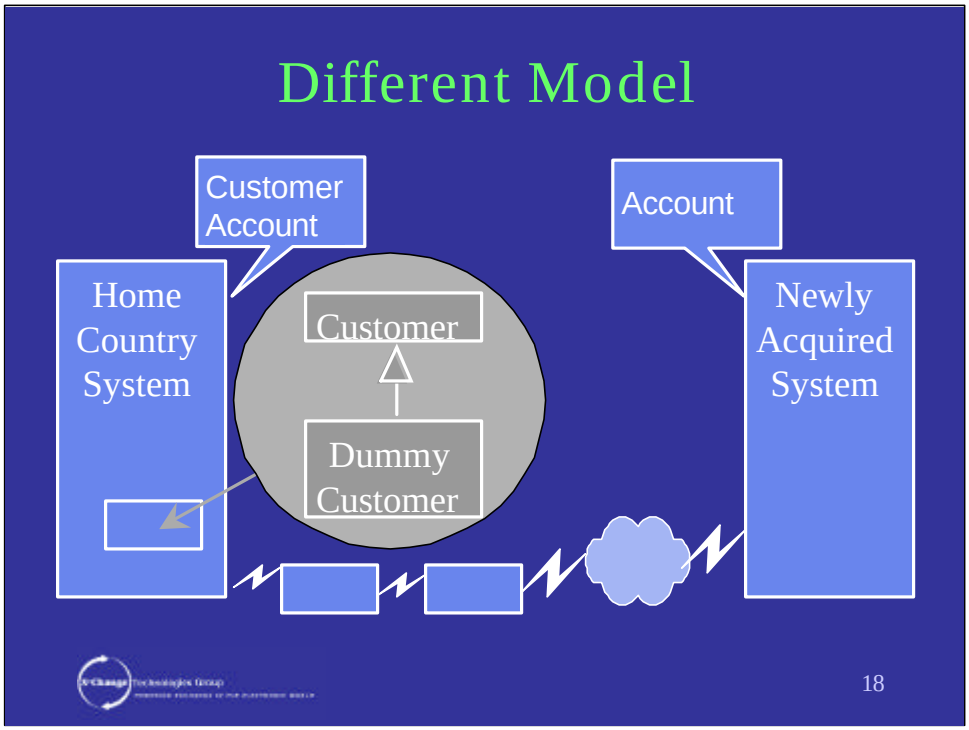
It is also very likely that an acquired system will have been built using a different model of the business.

Until recently it was quite common for the systems of banks in the USA to be based on accounts. The result was that it was very difficult, and often impractical, to find all the accounts of a particular customer. This means that it is not possible to report a single view of all the business relationships with a customer.

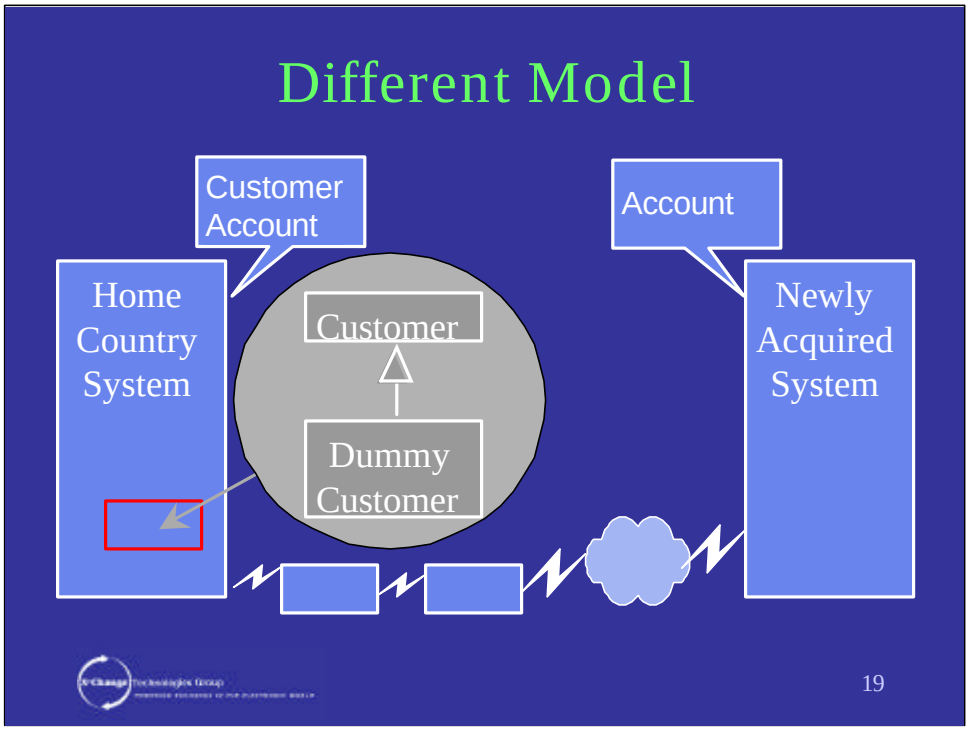
Notice that this kind of barrier can occur even within the same local part of the home country business. All of these barriers can show up at home.



The previous drawing did not mean to say that the home country system does not have account numbers; it does. But the model used to build the home country system keeps track of all relationships with a customer, using a single customer identifier, distinct from the account number.

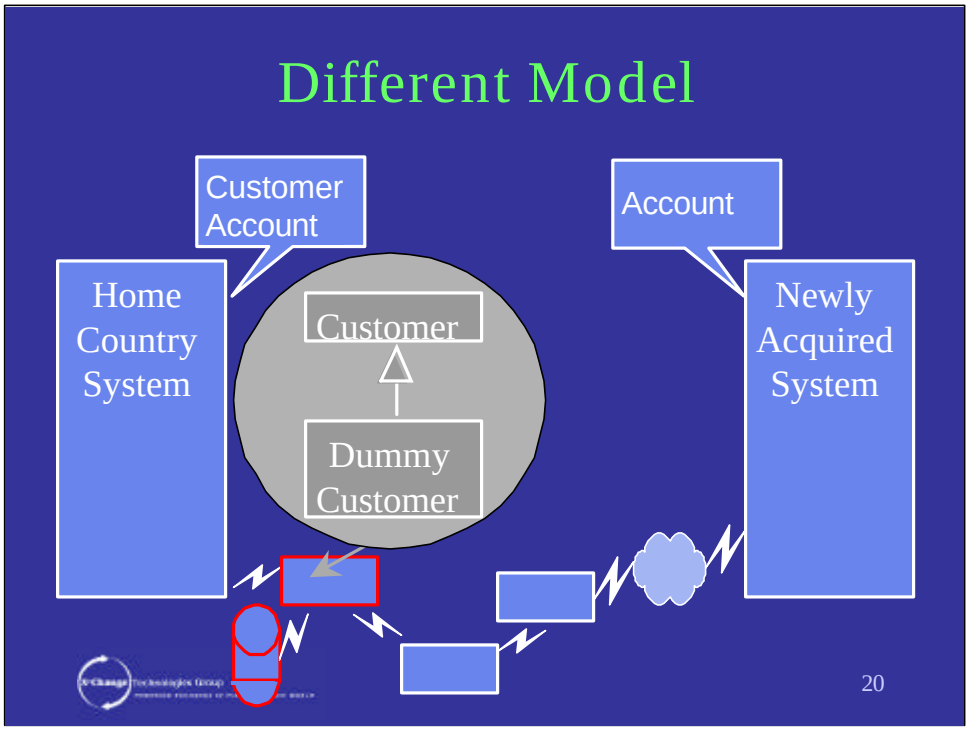


One way to overcome this barrier is to create a dummy customer for each account in the newly acquired system. It is still not possible to report a unified view of a customer from data in the newly acquired system, but at least the home country system is able to work with data communicated from the newly acquired system.



The newly added part, indicated in red, creates a dummy customer for every account in any acquired system, which does not have customer identifiers.

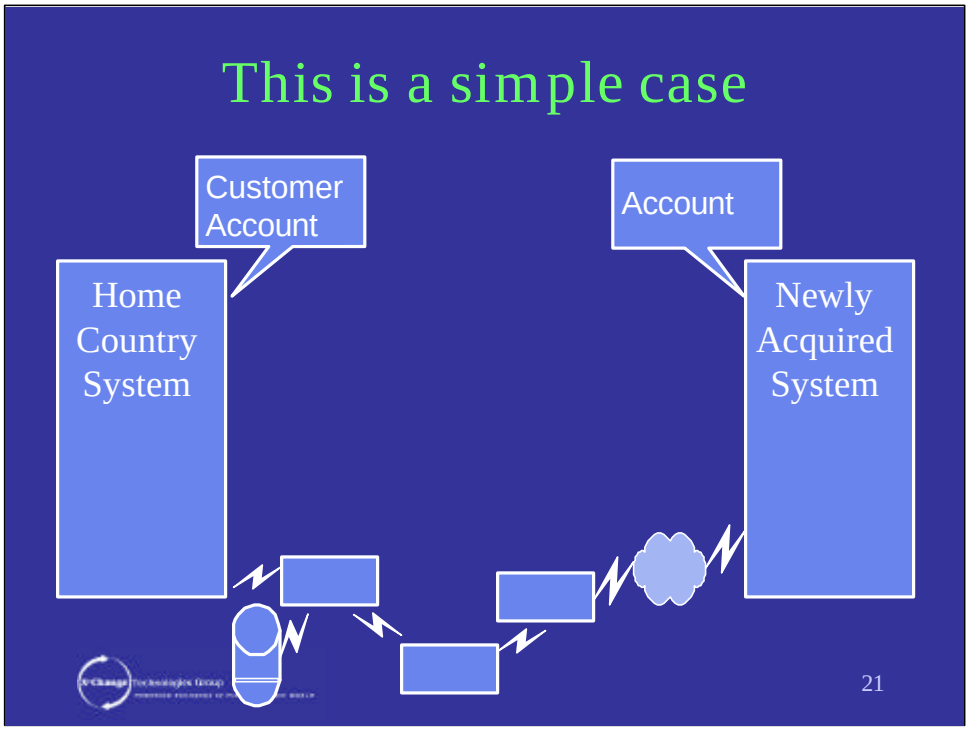
But there is a real risk here. This solution requires getting into and changing the home country system. I'm sure that at each of your companies all systems are well designed in anticipation of change, and it is not risky to make changes. (We call this quality 'modifiability.') But perhaps you know of a company where it is risky to make changes in some of the older systems.



An alternative to changing the home system is to introduce a new interceptor, which creates a dummy customer identifier for every account in any acquired system that does not have customer identifiers, and keeps track of those dummy customer identifiers, so it can add the correct dummy customer identifier to every incoming communication about an account, and remove the customer identifier from an outgoing communication, if that communication is with a system that does not keep customer identifiers.

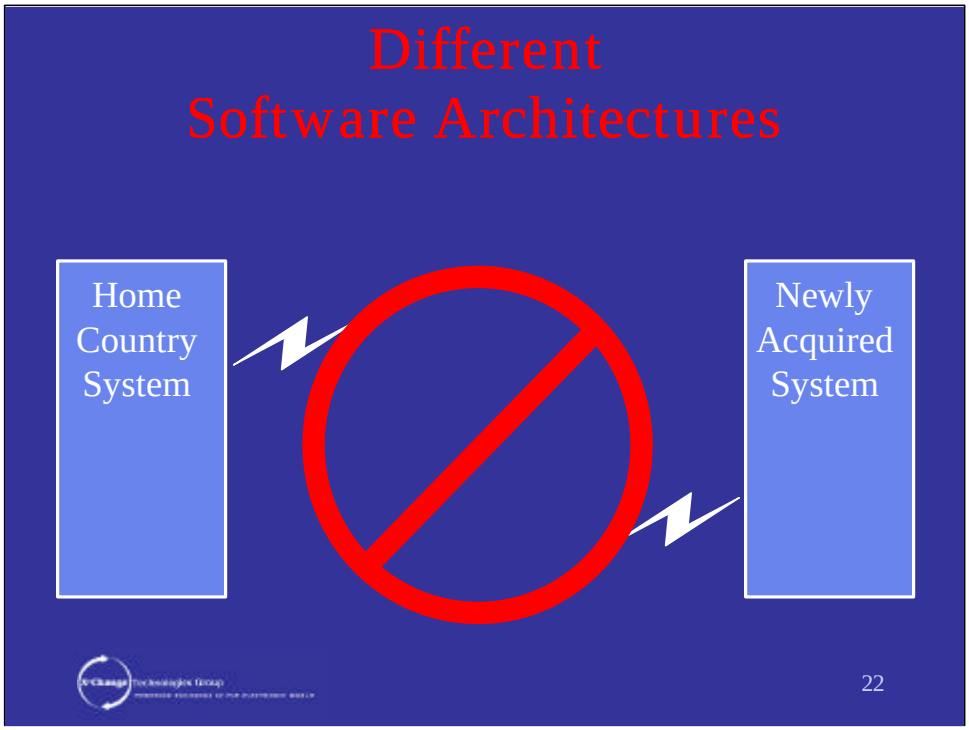
Now we have overcome the barrier without attempting to change the home country system.

Which of these two solutions to choose is an architectural decision.

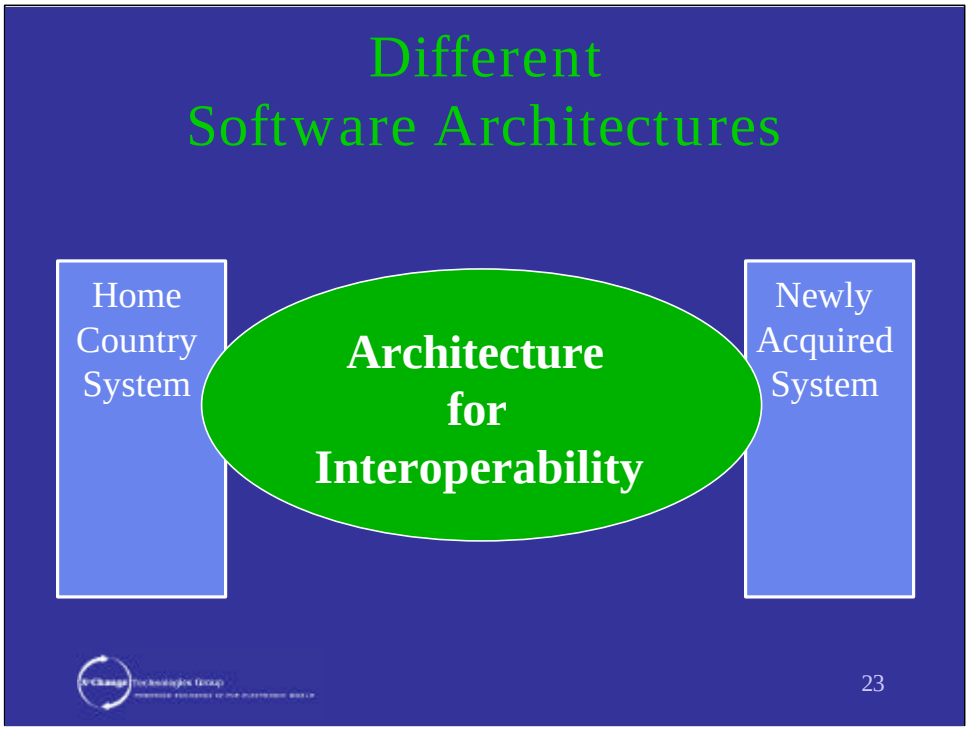


This is a simple case. We have encountered only three interoperability problems and have added only three interceptors and a new database. Real interoperability architectures are very often not this simple.

This is also a patchwork architecture, with a different fix for each problem. That may or may not be the best solution. A decision choosing a particular solution is an architectural decision.



The general case of failure of interoperability is that the two systems have different software architectures. Different languages, different names, and different models are all examples of different software architectures.



A general solution to barriers to interoperability is to build an architecture designed to provide interoperability, even between systems with different software architectures.

In a BOSC keynote Wednesday, I will be talking about the interoperability architecture of Wells Fargo Bank, an expanding enterprise.

I'll give a couple of examples of so-called solutions, which sound good when presented by the salesman. There are a lot more examples like this, but not time for them.

Single platform

"Overcome barriers to interoperability.
Use a single platform."

Single platform

A single platform enables **interworking**.

Because of interoperation failures between the various control subsystems, the captain was not able to restore the operation of the control system after a software failure, and so he was unable to control his ship.

It is reported that, in order to stop his ship and wait for a tow, he shut down power to the control systems.

……..

Using a single platform does not prevent interoperability problems.

It does overcome some of the barriers to interoperability. But these can also be overcome in other ways.

And it is certain that interoperation using different platforms can be provided easily, when attention is paid to architecture.

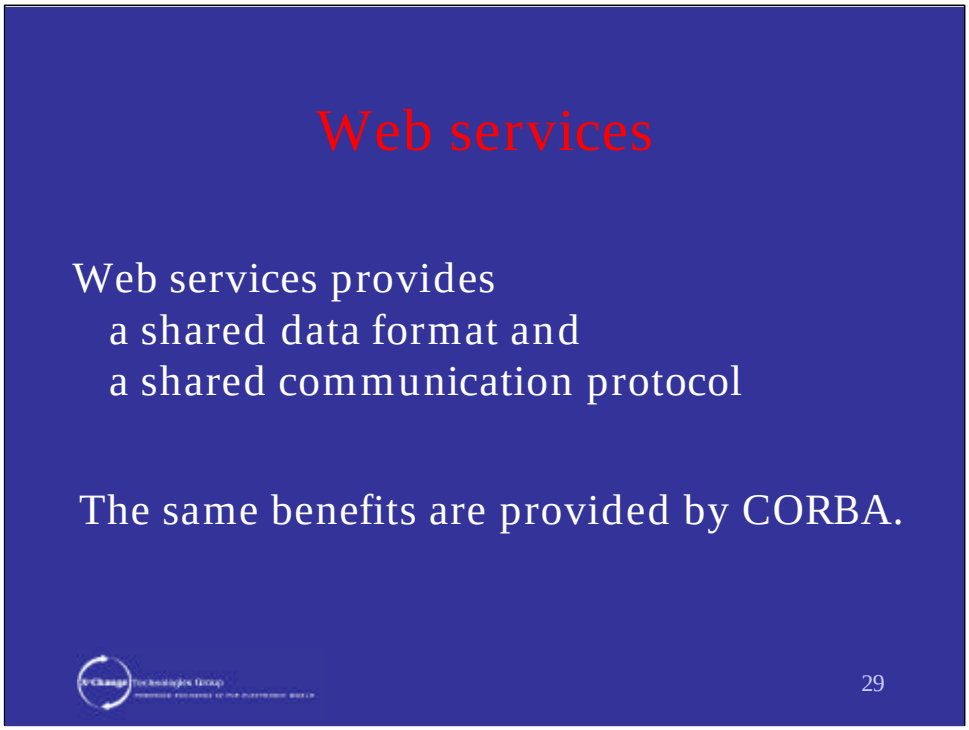
You experience trouble-free interoperation of software on one platform with software on an entirely different platform, whenever you use a standards-compliant web browser, such as Opera, on the Wintel PC platform, to visit a standards-compliant web site using a different platform, such as Apache on Linux.

Even, sometimes, when you use a compliant browser to visit a non-compliant web site.

If your customers are having trouble at your web site, the cause is often that you use non-standard technology, such as ActiveX.

Web services

"Overcome barriers to interoperability.
Use web services."



Using web services does not prevent interoperability problems.

It does overcome some of the barriers to interoperability. But these can also be overcome in other ways.

And it is certain that interoperation can be provided easily, when attention is paid to architecture.

Web services

Two added benefits of web services are:

- use of familiar tools
- defeat of firewall protection.

Web services

Because there is so much work being done on web services, in some cases it may be a good choice for an interworking protocol

Whether or not is an architectural question.

Design-time interoperability means simply: designing systems so they can interoperate.

An Architecture for Interoperability

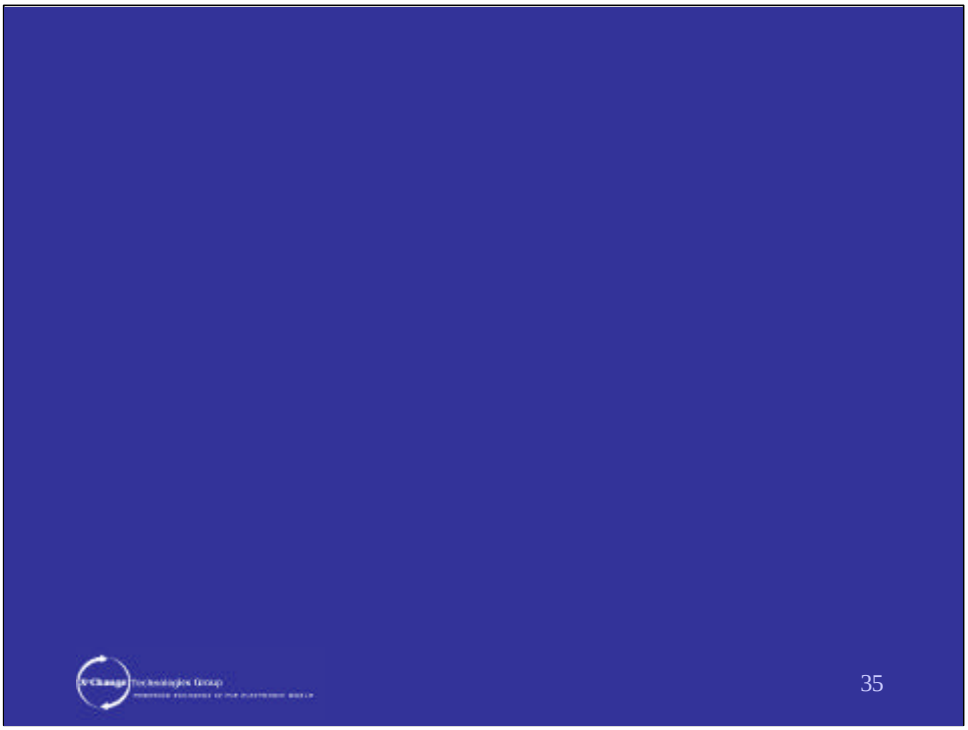
To ensure rapid and successful integration of existing systems

Put in place an architecture for interoperability

An Architecture for Interoperability

- Shared business model
  Model of the business community

  RM-ODP Enterprise Language
  www.joaquin.net/ODP/Part3/5.html
- Shared information model
- Shared integration architecture

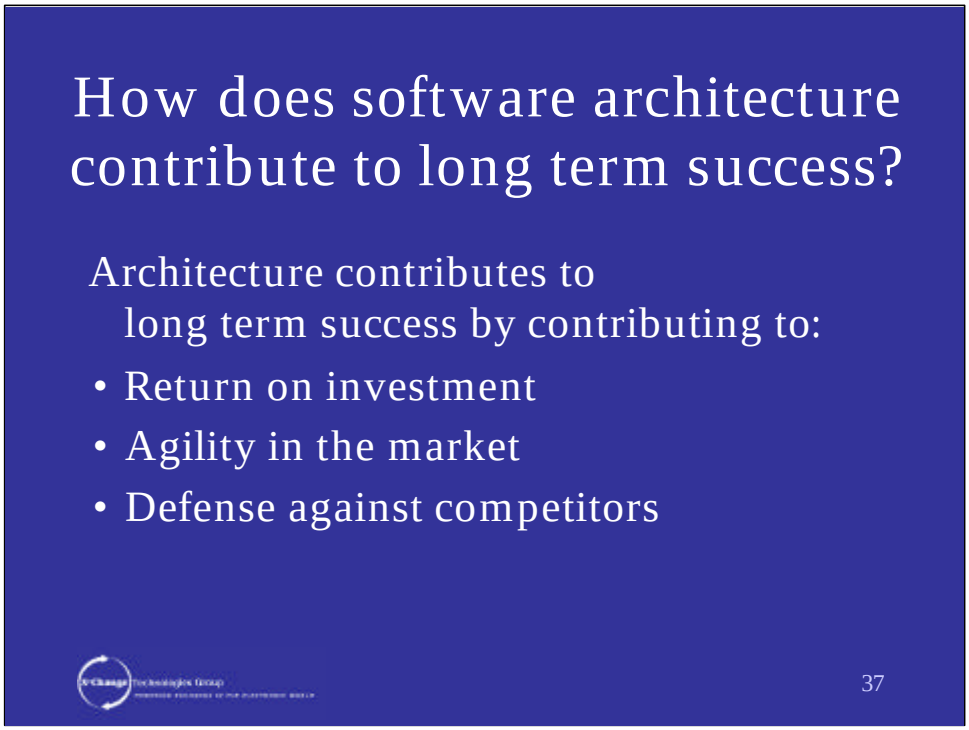


A pause, after all that technical material.

How does software architecture contribute to the bottom line?

This is a question that interests many executives in my home country.

Let's ask a better question.



These are just three of the important contributions good software architecture can make to success

How does software architecture provide a defense against competitors?

The classics on war teach

the best way to defeat your enemy is to bring the enemy to defeat himself.

Your competitors study these classics.

They will take care to give you opportunities to defeat yourself.

The classics on war teach that the very best way to defeat an enemy is to bring the enemy to defeat himself.

Your competitors study these classics. They will take care to give you opportunities to defeat yourself.

How do you defend against such an attack?

By taking care to not defeat yourself.

How does software architecture provide a defense against competitors?

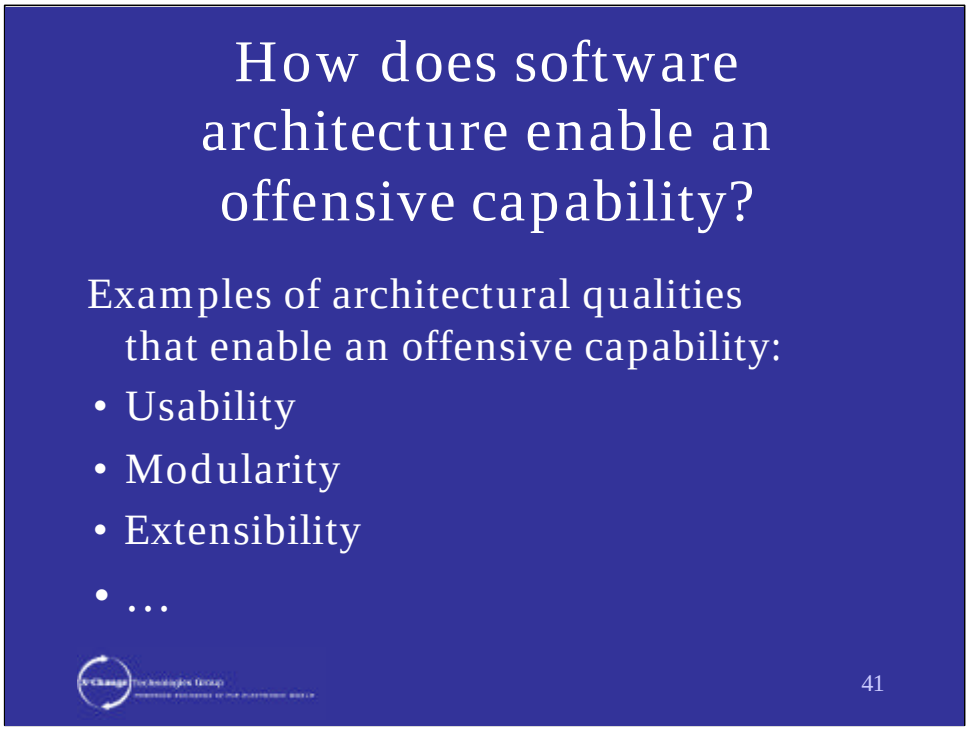
Failure to give proper attention to architecture is self defeating.

Good architecture provides a defense against competitors by defending against self-defeating practices.

Failure to give proper attention to architecture is self defeating.

Good architecture provides a defense against competitors
by defending against

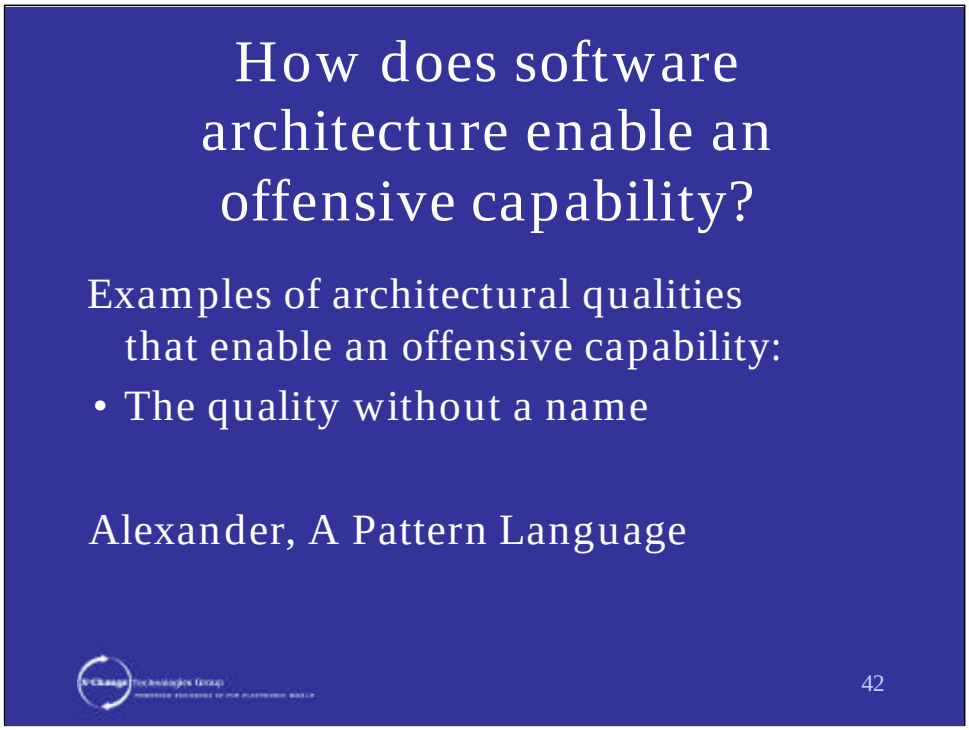
self-defeating practices.



An example in financial services software is the famous (among we software architects) Swiss financial derivative products—but the same applies to any kind of new product

It is important to recognize how the Swiss were able to do what they did. They used a thorough analysis of the domain and a toolkit of Smalltalk classes together with a framework for a derivative evaluation program. These two things enabled creating, in one day, a program to determine the value of a new kind of derivative.

This is an example of the value of a good software architecture combined with a creative analysis of the problem.



An example in financial services software is the famous (among we software architects) Swiss financial derivative products—but the same applies to any kind of new product

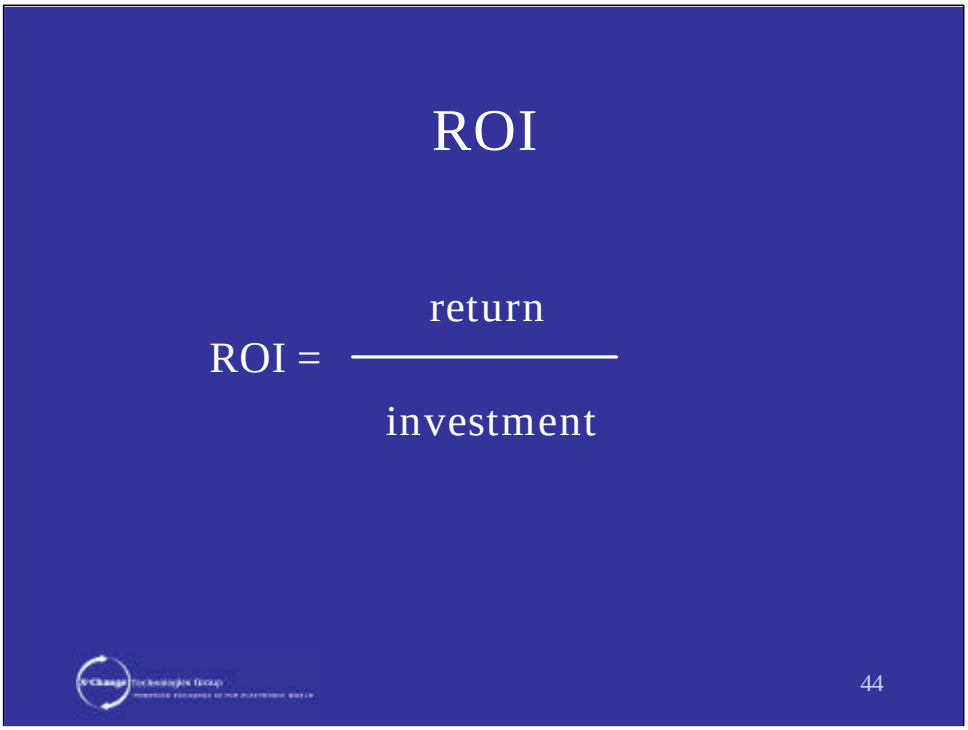
It is important to recognize how the Swiss were able to do what they did. They used a thorough analysis of the domain and a toolkit of Smalltalk classes together with a framework for a derivative evaluation program. These two things enabled creating, in one day, a program to determine the value of a new kind of derivative.

This is an example of the value of a good software architecture combined with a creative analysis of the problem.

Extensibility

Capture new markets quickly

by allowing rapid implementation of new products and product features

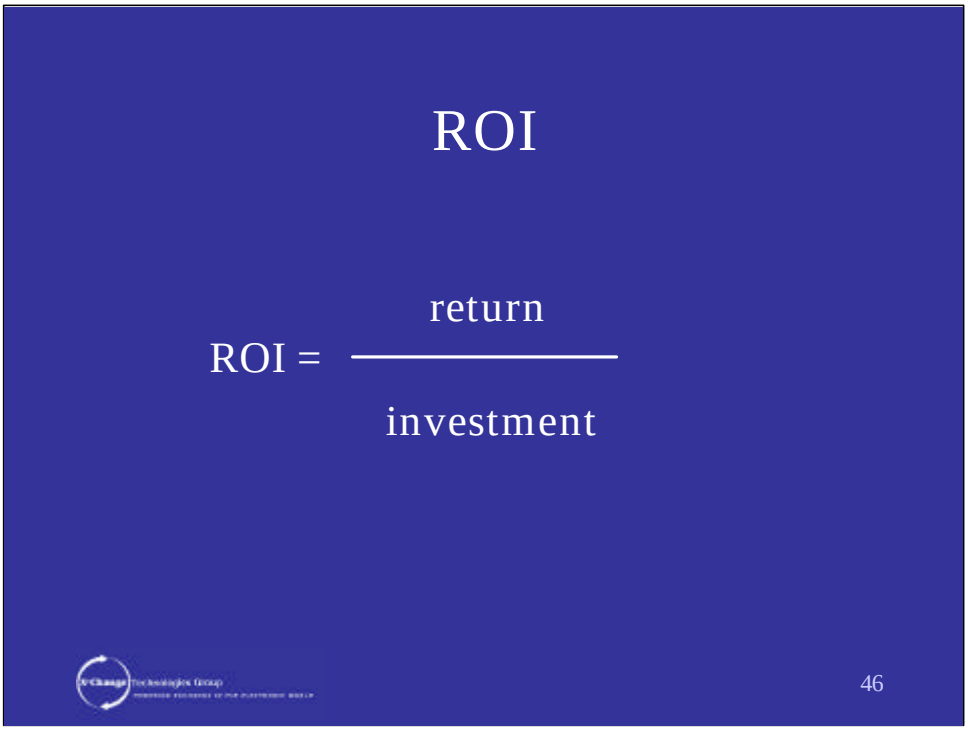


Most of the software people who talk to you about ROI, don't even know what ROI means.

Often people say ROI when they mean the time to recover the initial investment.

Often ROI is thought of as return divided by investment.

Of course, and as you know well, return on investment is a percentage rate of return that is a function of the amount of the original investment and the stream of returns over time.

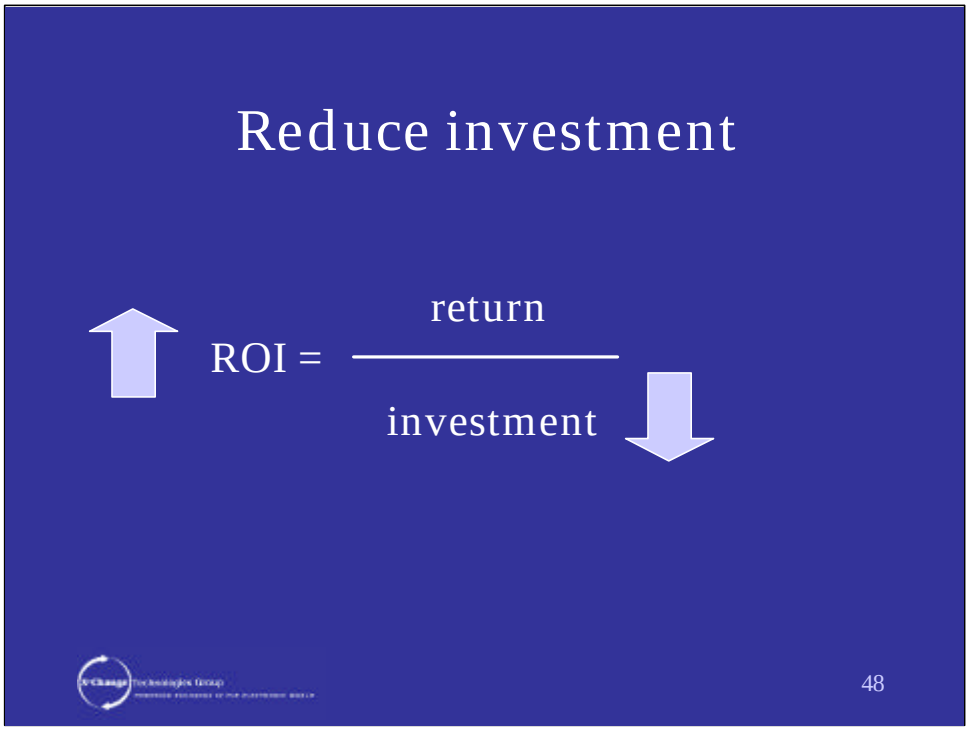


So this formula is incorrect, but will be a useful graphic to illustrate what I have to say next.

ROI

You have acquired a company:

What is your return on that investment?



This is one way to increase ROI. But at first glance this does not seem to be useful to improve the return on the investment in the case of an acquisition. After all, the original investment is fixed at the time of the purchase.

Of course, after a moments thought, we understand that the investment will continue after the acquisition.

Architecture to reduce investment

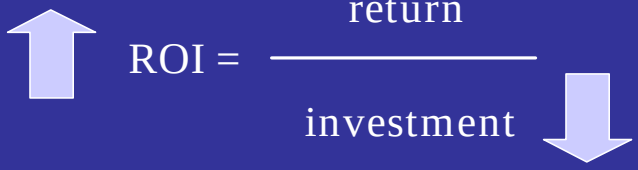
Effective application of an architecture for integration will greatly reduce the cost of integrating the acquired systems.

ROI

You need a new system
to support a new product,
or to improve customer service,
or … :

What will be your return on that investment?

Reduce investment



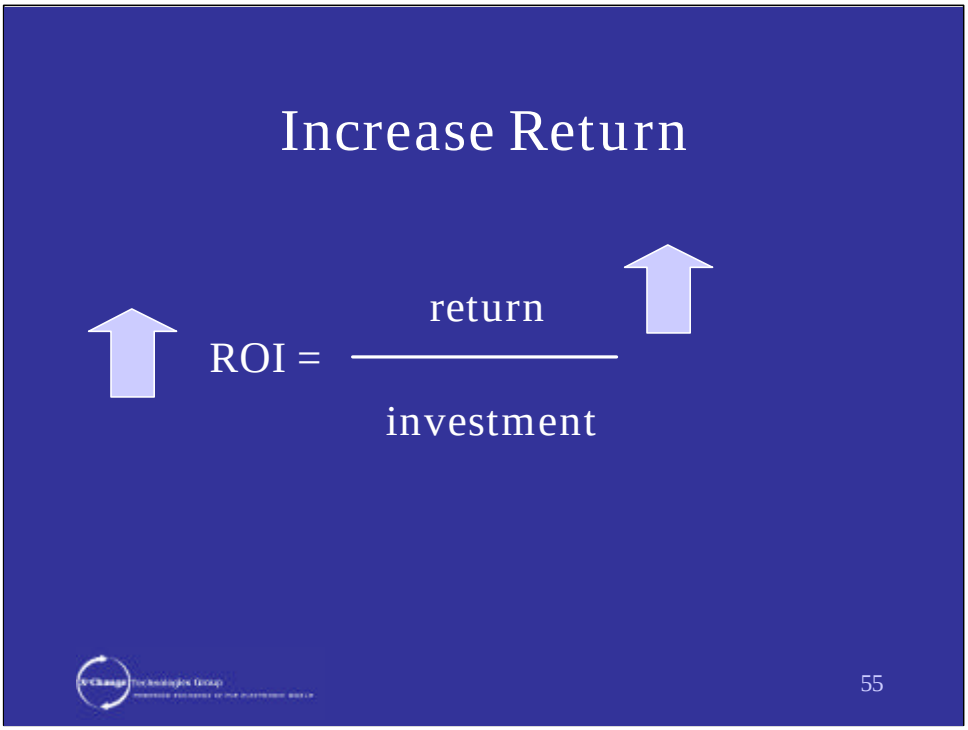
Architecture to reduce investment

Effective application of
an architecture for integration
will greatly reduce the cost
of integrating the acquired systems
and the cost of new systems.

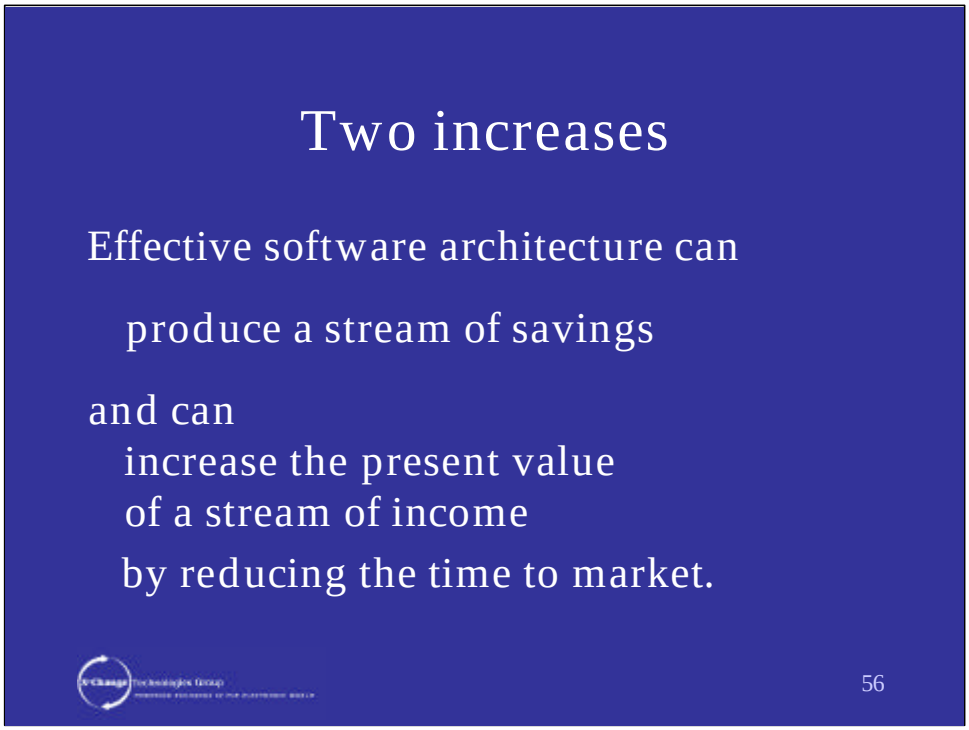
Approaches such as the OMG Model Driven Architecture can reduce the cost of new systems. One example, for a very small program, showed a development cost reduction of one third.

But development cost is actually a very small part of the total investment in a system (the total cost of ownership).

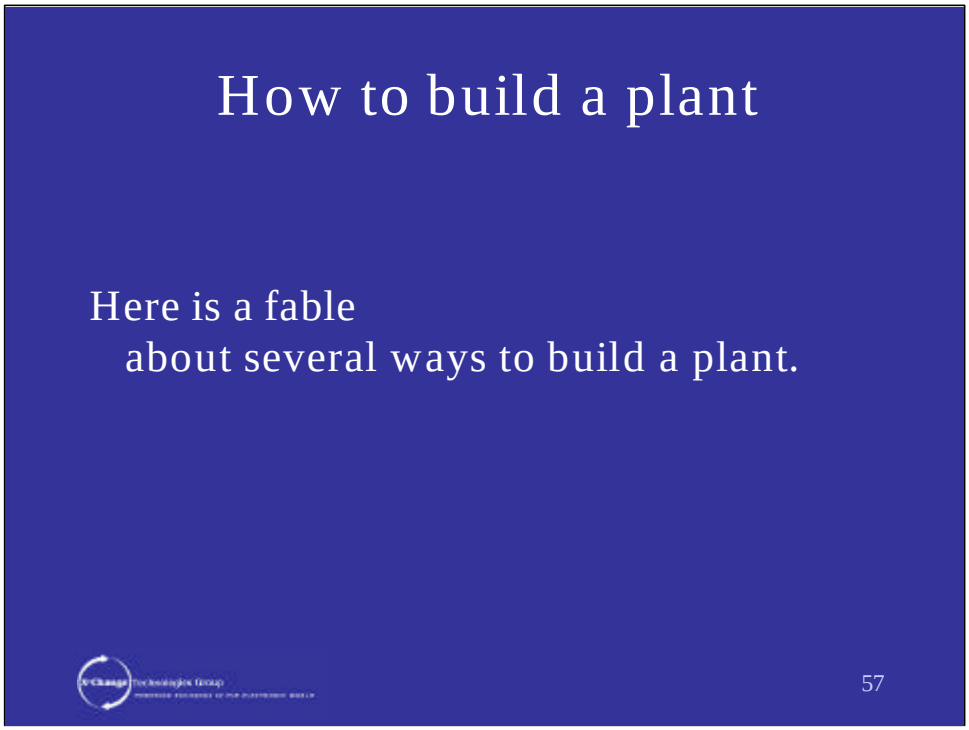
Good software architecture will reduce all these costs.



The other way to improve ROI is to increase the return.



Two ways to increase the return are
—to produce a stream of savings and
—to provide earlier returns.



Instead of continuing my pitch for software architecture, let me offer you a sort of fable. I will describe several different ways to build a plant.

How to build a plant

I:

Hire welders, electricians, and other craftsmen, and a project manager.

Give them money.

Tell them to build the plant.

How to build a plant

II:

Hire an architect and some engineers.

Have the architect design the plant
with the help of the engineers.

Give the plans and money to a contractor.

Tell him to build the plant.

How to build a plant

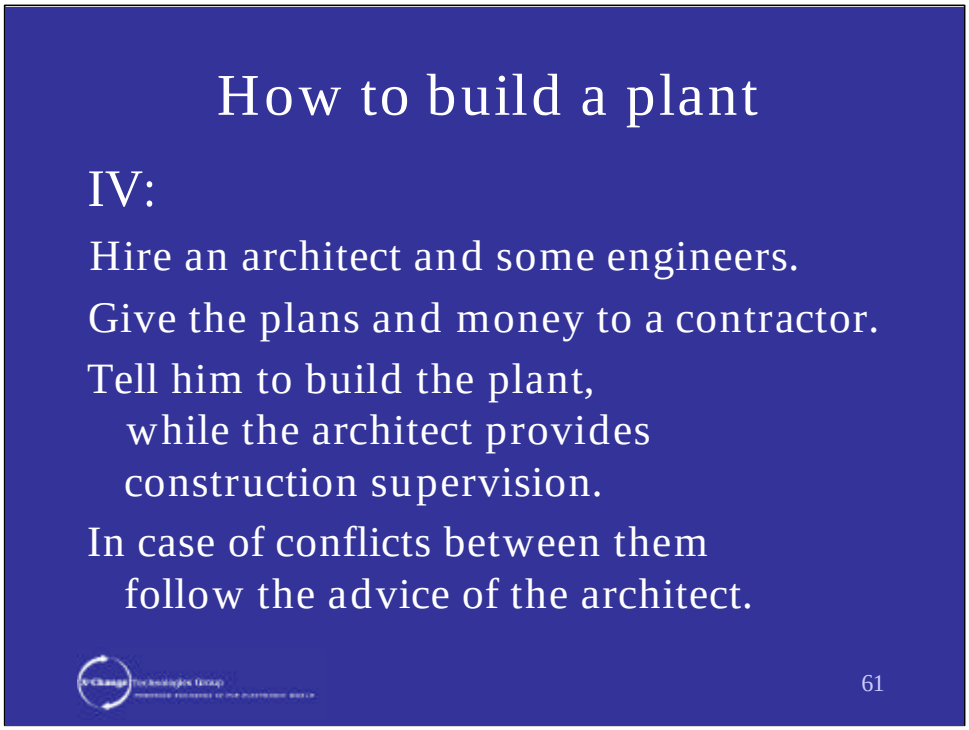
III:

Hire an architect and some engineers.

Give the plans and money to a contractor.

Tell him to build the plant,
while the architect provides
construction supervision.

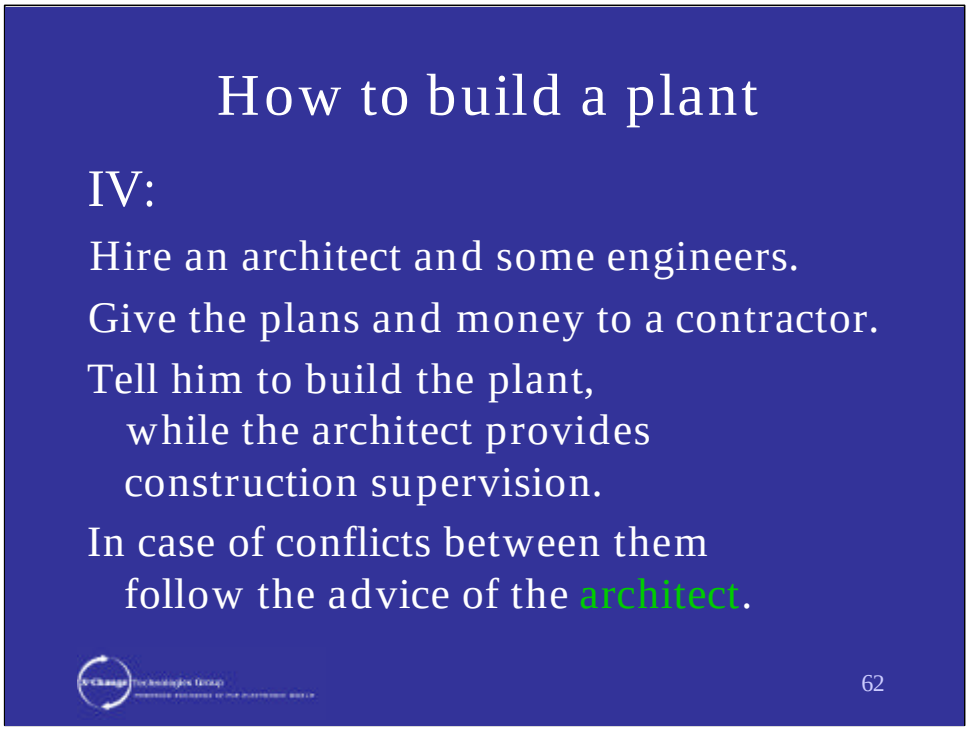
In case of conflicts between them
follow the advice of the contractor.



Of course, it is not as simple as always following the advice of the architect. You are the executive. Your job is to listen to both sides and make the decision.

But look back at III.

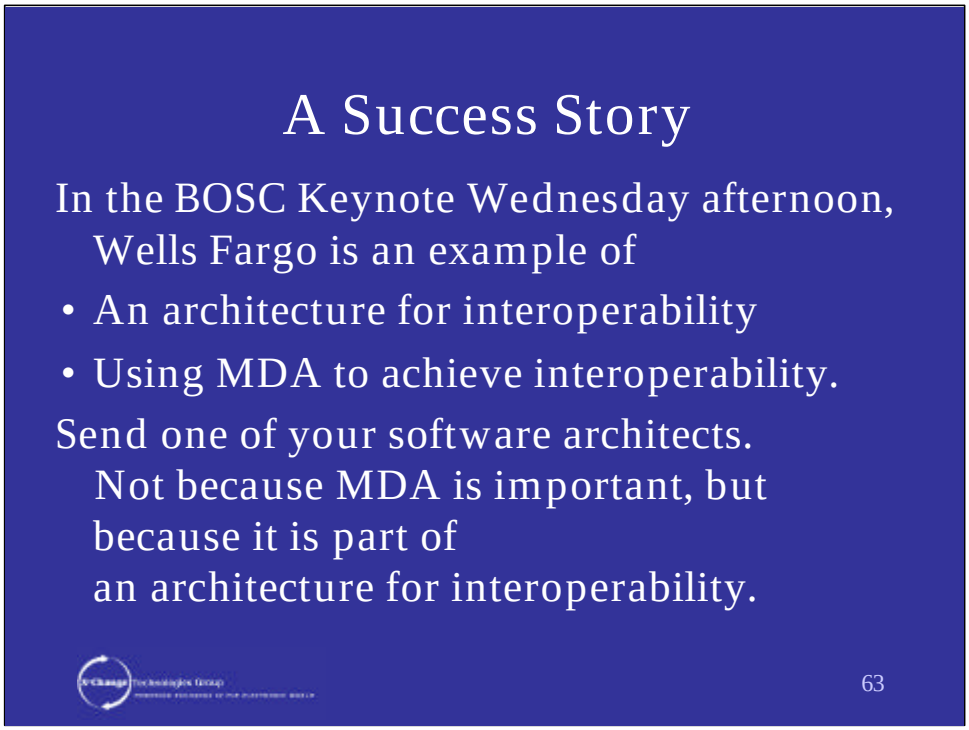
Are your managers always choosing what will get the job done "ahead of time and under budget," even when the architect explains that this choice will increase the total cost?



Of course, it is not as simple as always following the advice of the architect. You are the executive. Your job is to listen to both sides and make the decision.

But look back at III.

Are your managers always choosing what will get the job done "ahead of time and under budget," even when the architect explains that this choice will increase the total cost?



In the BOSC Keynote Wednesday afternoon, I mention Wells Fargo Bank in an example of using Model Driven Architecture (MDA) to achieve interoperability.

Send one of your software architects. Not because MDA is important, but because what Wells Fargo did is an example of an architecture for interoperability, whether MDA is used or not.

If you don't have software architects, get one.

Software Architects

If you don’t have software architects,

get one.

Interoperability

What does it mean for enterprises of emerging economic powers?

Joaquin Miller
Chief—Architecture Review
X-Change Technologies
www.xchangetechnologies.com