

What is a Business Process?



Learning Objectives

After this lesson you will be able to...

- Provide an example of a business process
- Identify a context diagram
- Describe the types of diagrams used to document business processes

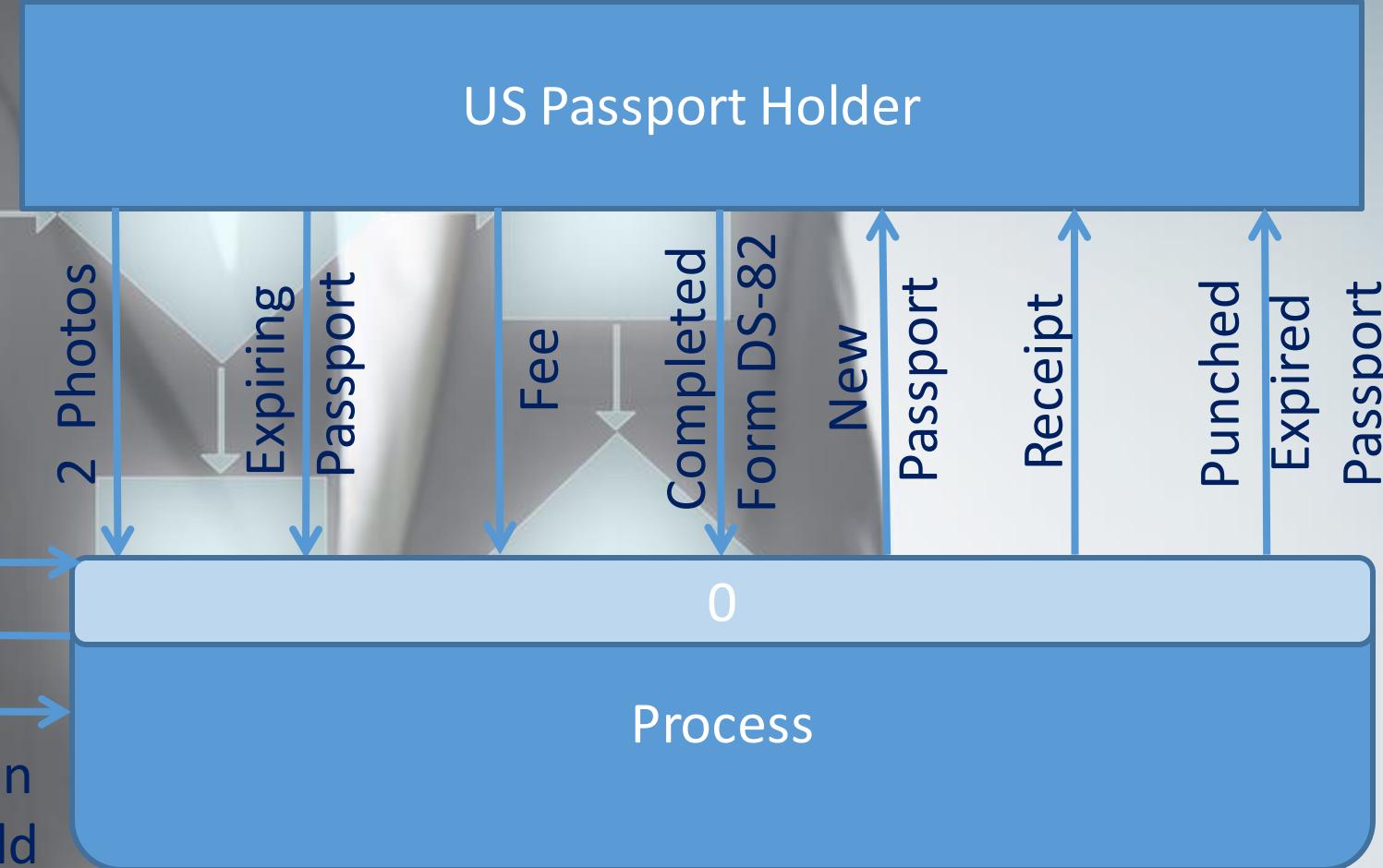
What is a Business Process?

US Dept of State
Central Name
Check System

US Dept of
Health &
Human Services

Pass or Fail
Name, SSN

List of those in
arrears of child
support payments



What is a Business Process?

US Dept of State
Central Name
Check System

Pass or Fail
Name, SSN

US Dept of
Health &
Human Services

List of those in
arrears of child
support payments

US Passport Holder

2 Photos
Expiring
Passport

Fee

Completed
Form DS-82

New
Passport

Receipt

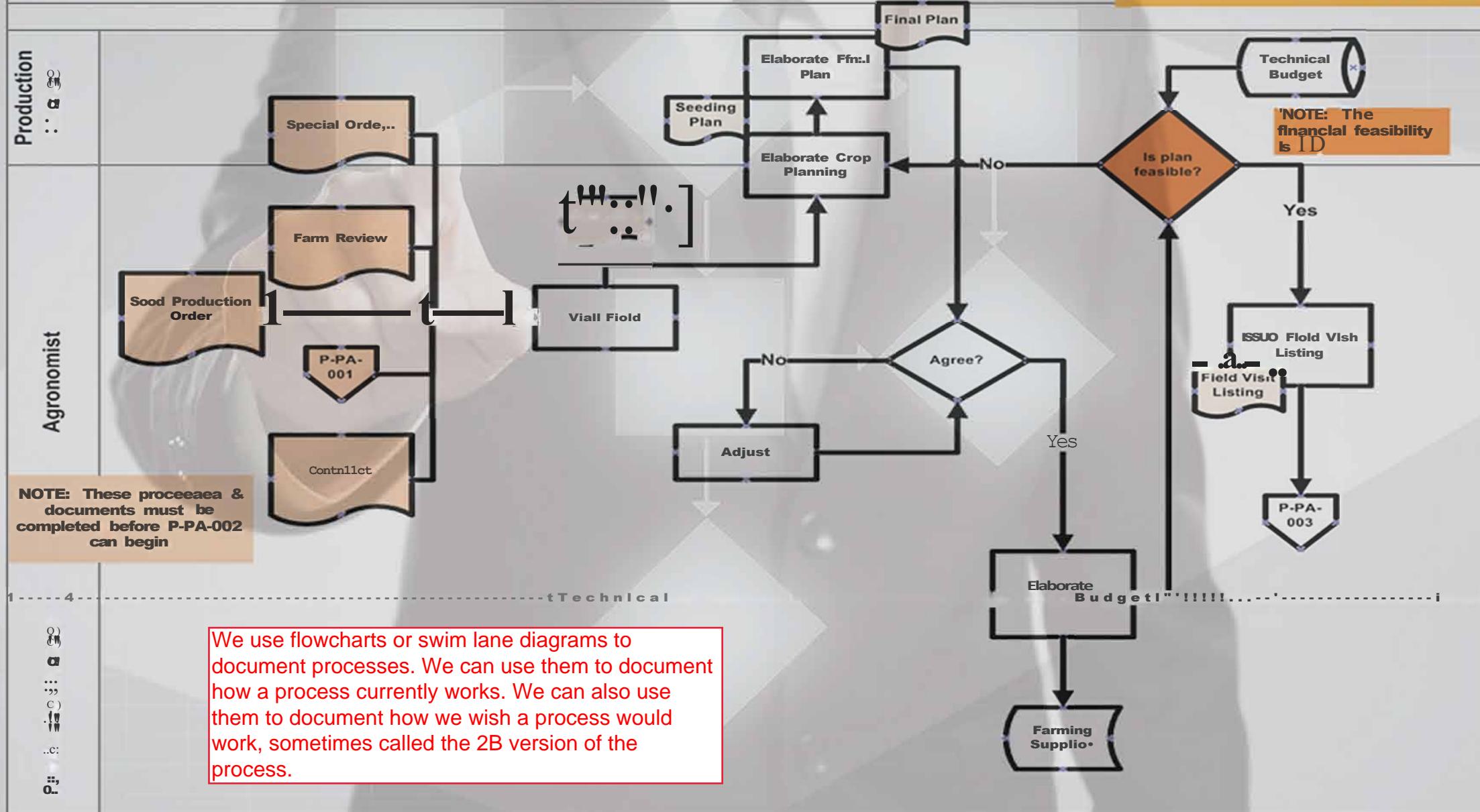
0

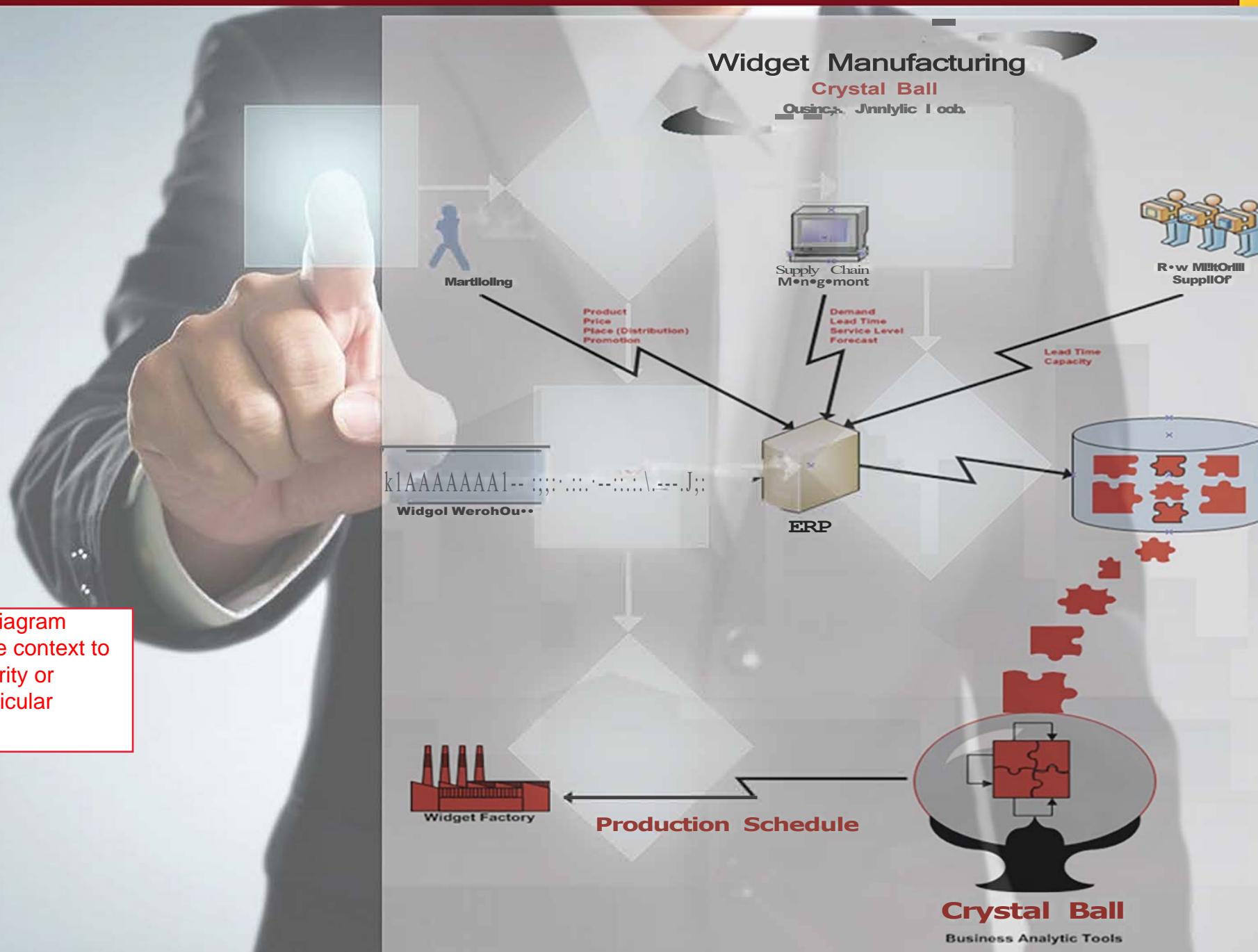
Punched
Expired
Passport

US Passport Renewal Process

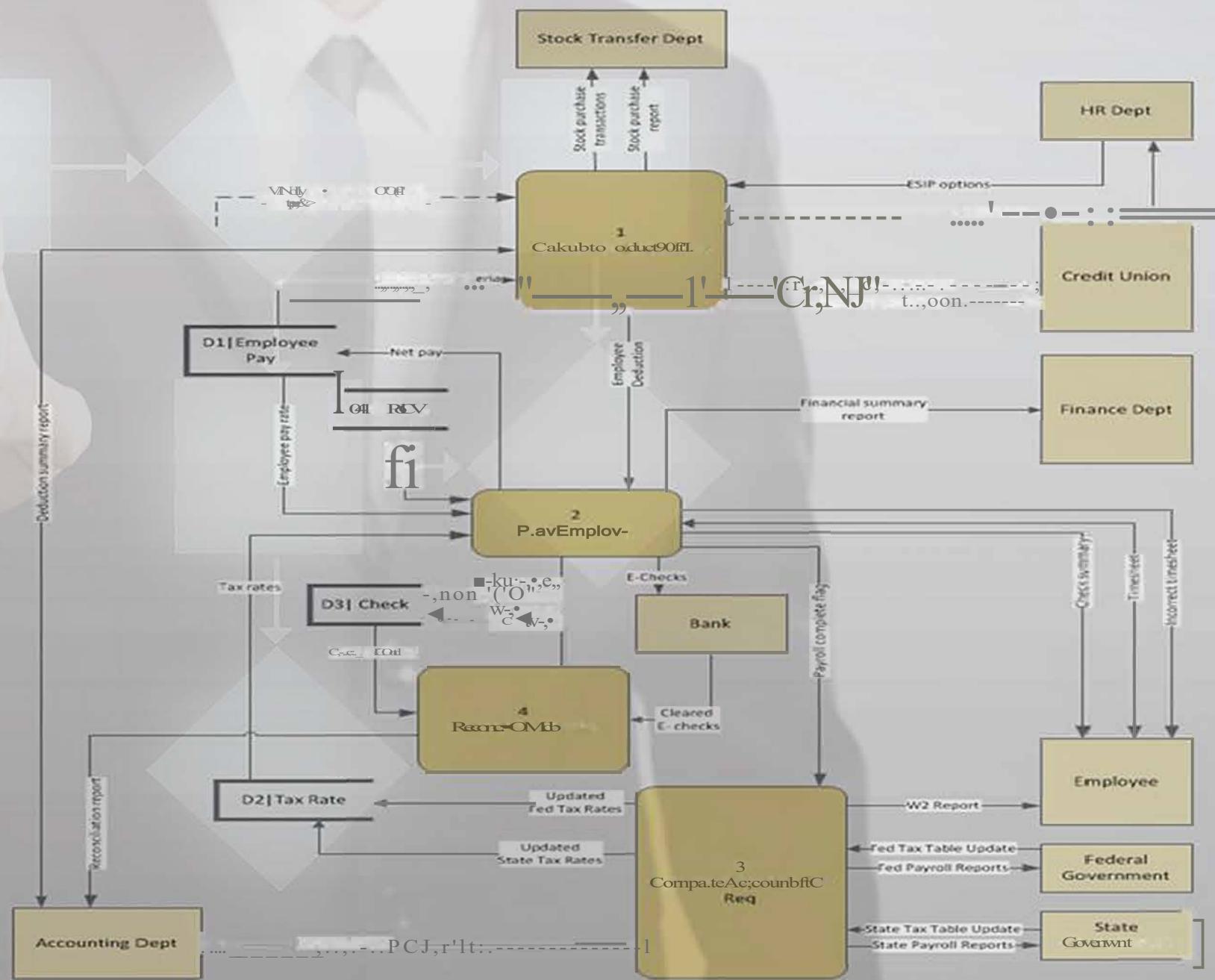
Context Diagram

The above diagram is a context diagram for the US passport renewal process. A person would give the passport office information such as her/his name, some photos, an expired passport, and a payment. All of these go or inputs effectively into the process, and the output of the process is a renewed passport for the person.

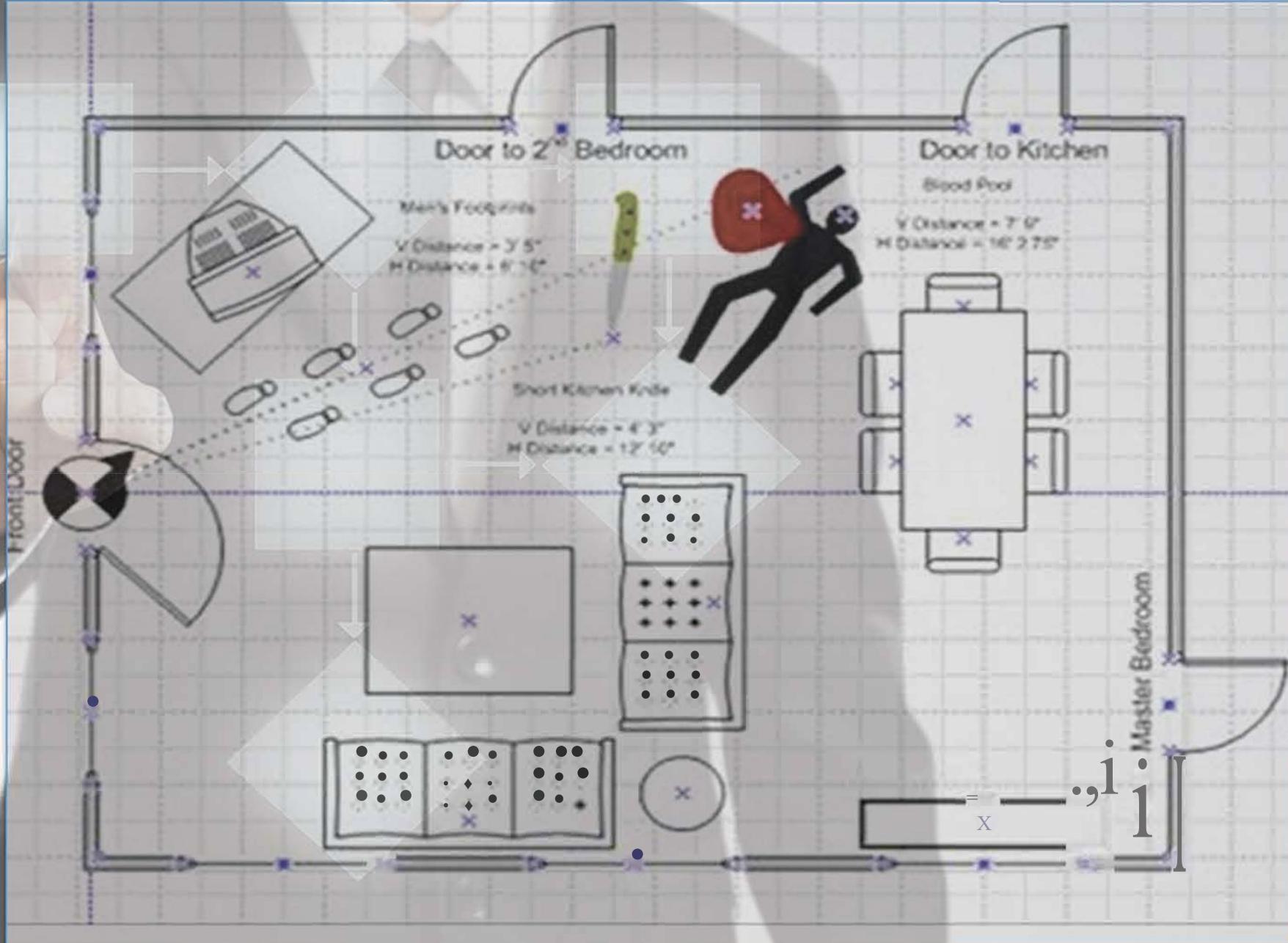
Crops Planning (Process P-PA-002)




Data flow diagrams follow a very specific syntax, very much like flowcharts or swim lane diagrams, and this specific syntax helps contribute to greater understanding of where data might be flowing in a process.



Informal Diagrams:
There are any number of informal diagrams that help contribute to the understanding of business processes.



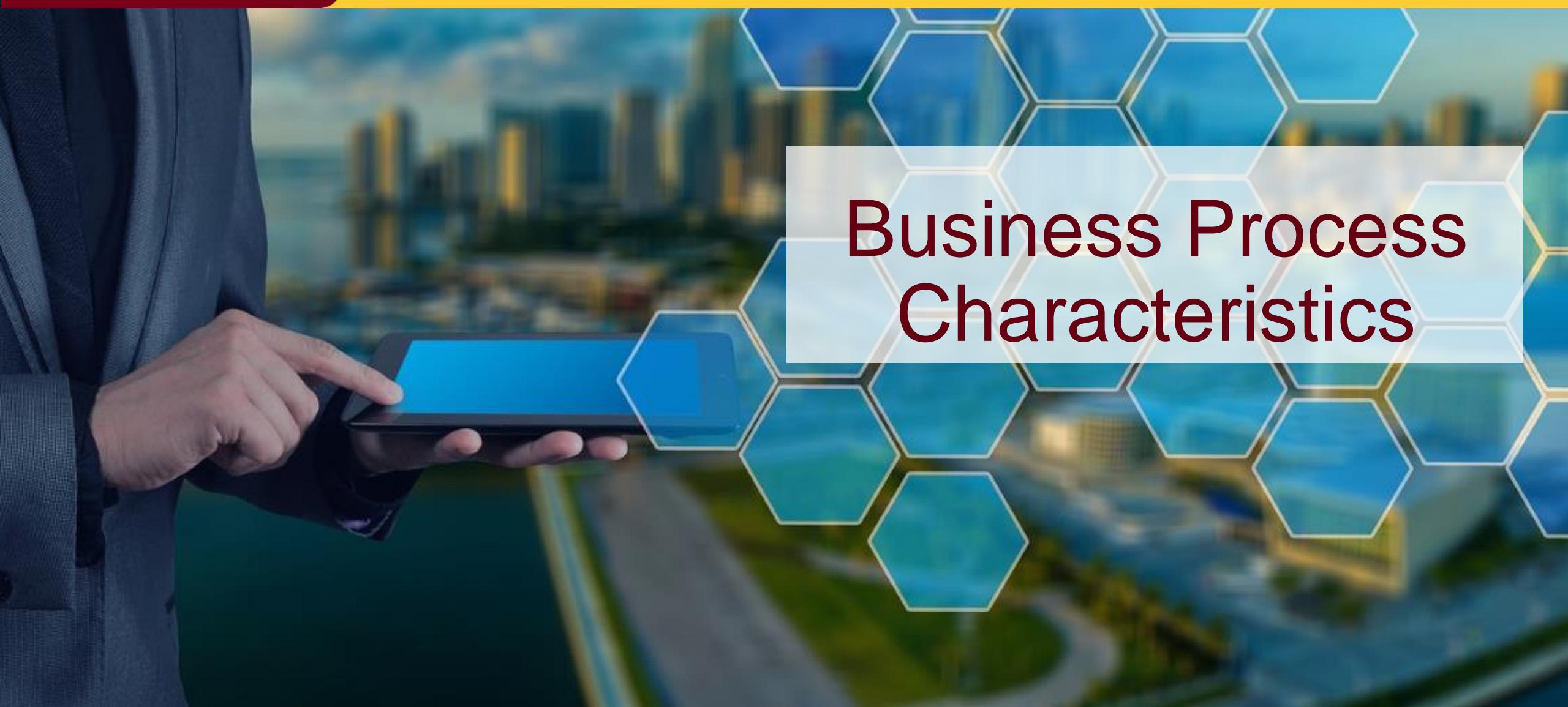


CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA



Business Process Characteristics



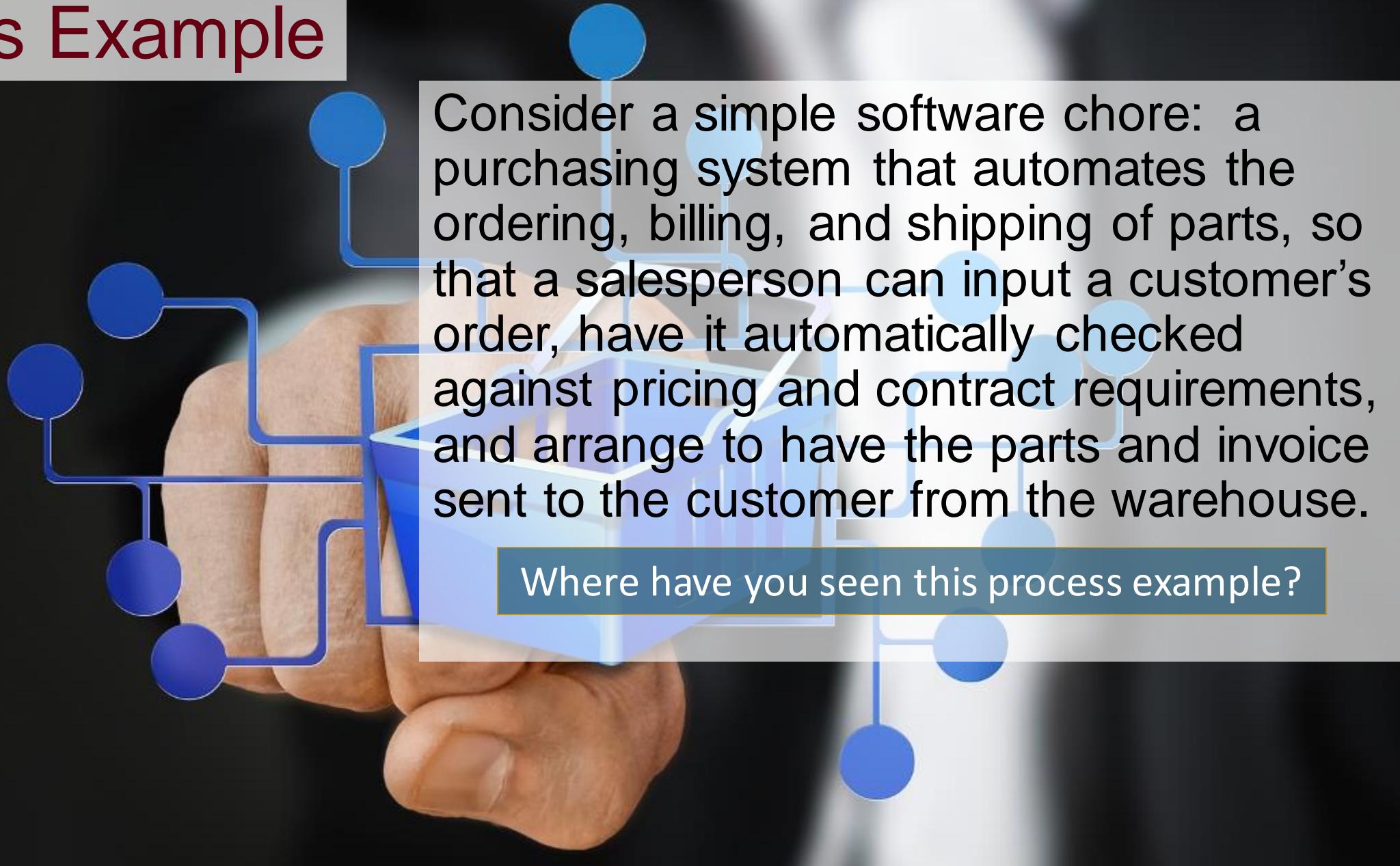
Learning Objectives

A close-up photograph of a white and blue soccer ball that has just been kicked into a dark-colored goal net. The ball is positioned on the left side of the frame, with the net filling the background.

After this lesson, you will be able to...

- Define a business process
- Describe the components of a business process
- Explain the varying levels of automation with respect to business processes

Process Example

A close-up photograph of a person's hand, wearing a dark shirt, interacting with a blue process flow diagram. The diagram consists of several blue circular nodes connected by blue lines, forming a network. The hand is pointing at one of the nodes. The background is dark, making the blue elements stand out.

Consider a simple software chore: a purchasing system that automates the ordering, billing, and shipping of parts, so that a salesperson can input a customer's order, have it automatically checked against pricing and contract requirements, and arrange to have the parts and invoice sent to the customer from the warehouse.

Where have you seen this process example?

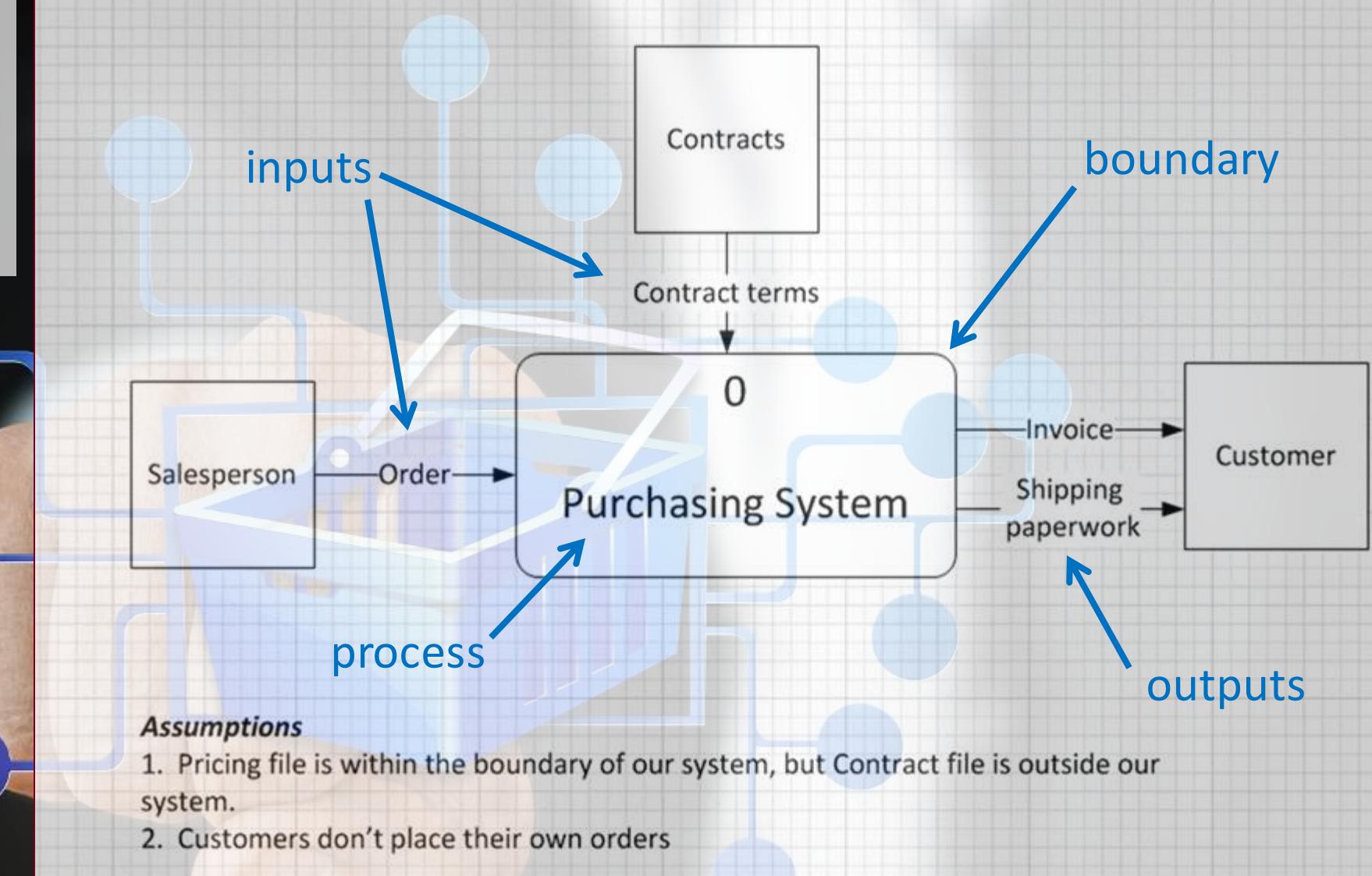
Process Components & Characteristics

A context diagram shows the system and external entities as rectangular boxes.

A process has inputs. In this case, the salesperson is entering the order and contracts department might be inputting the contract terms such as net 30 or net 60. Also the process does some processing, and that's represented by the central box here, the purchasing system.

The pricing file is within the boundary of our system but the contracts file is not.

We assume that customers don't place their own orders.

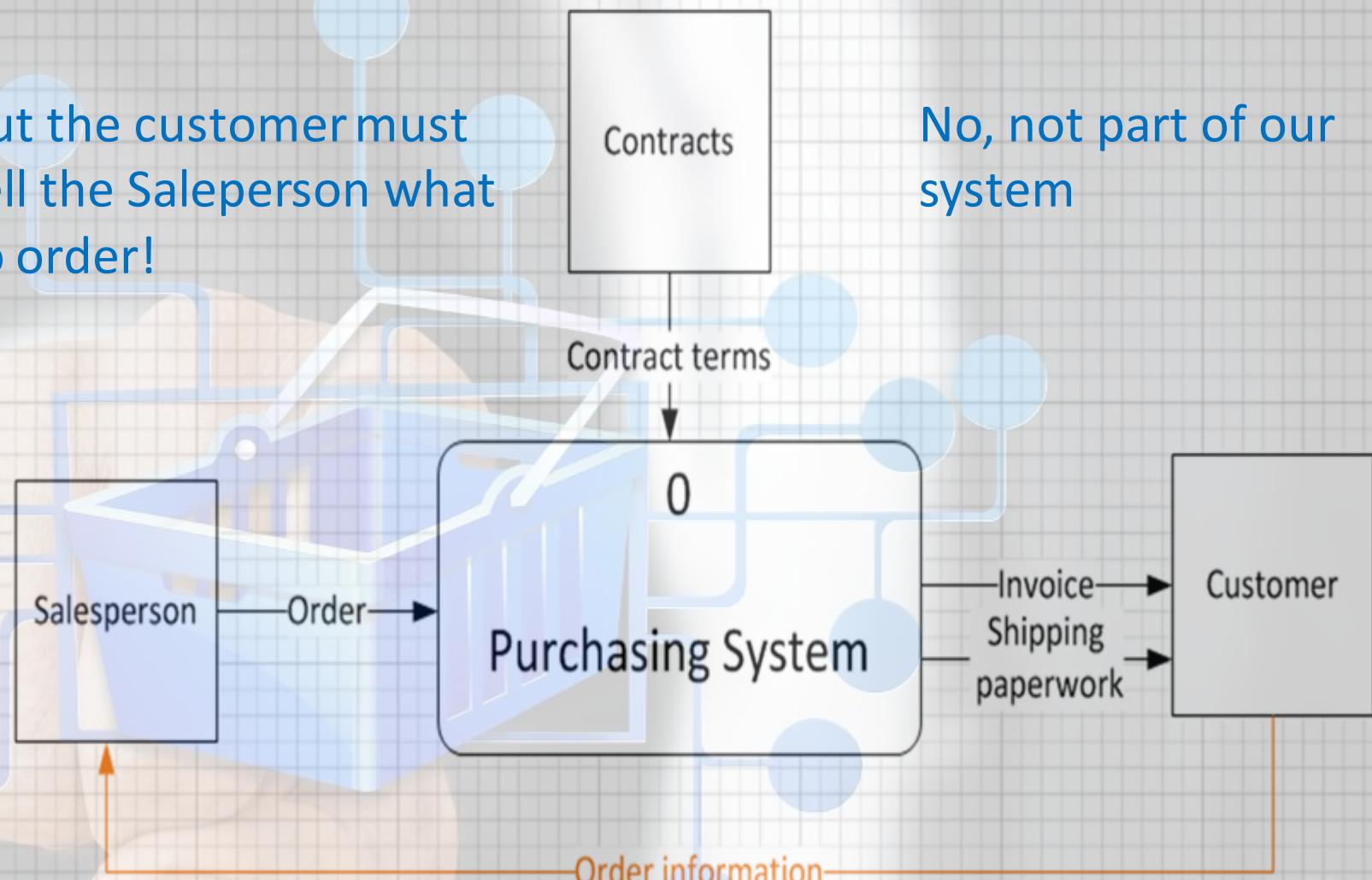


Process Components & Characteristics

We made an assumption that all orders are placed by a salesperson, even if the customer is communicating with them by phone or email.

But the customer must tell the Salesperson what to order!

No, not part of our system



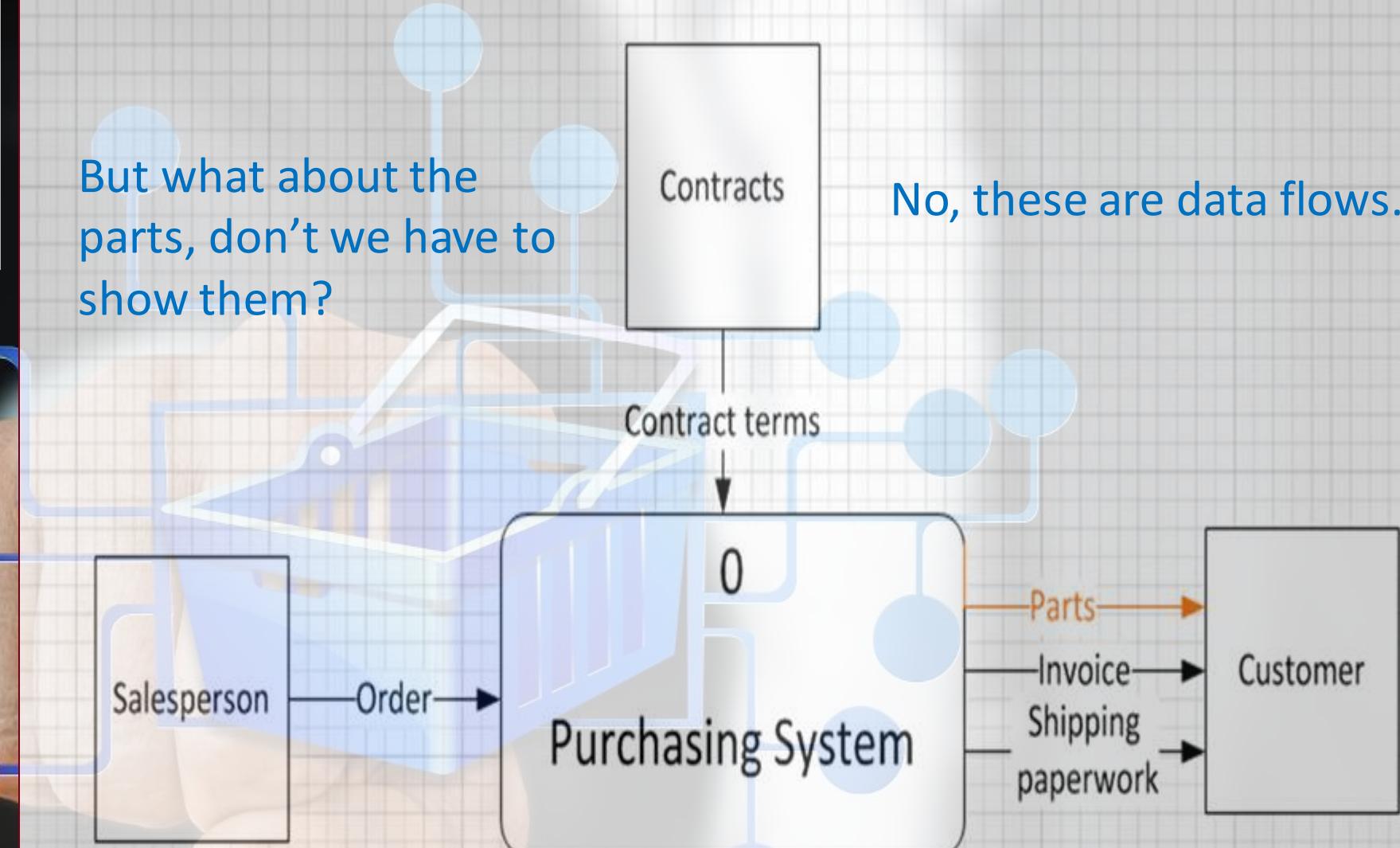
Process Components & Characteristics

Do we have to show physical goods in our diagram?

Generally we do not because these diagrams represent data flows relative to the process and not necessarily the physical goods that are moving around.

But what about the parts, don't we have to show them?

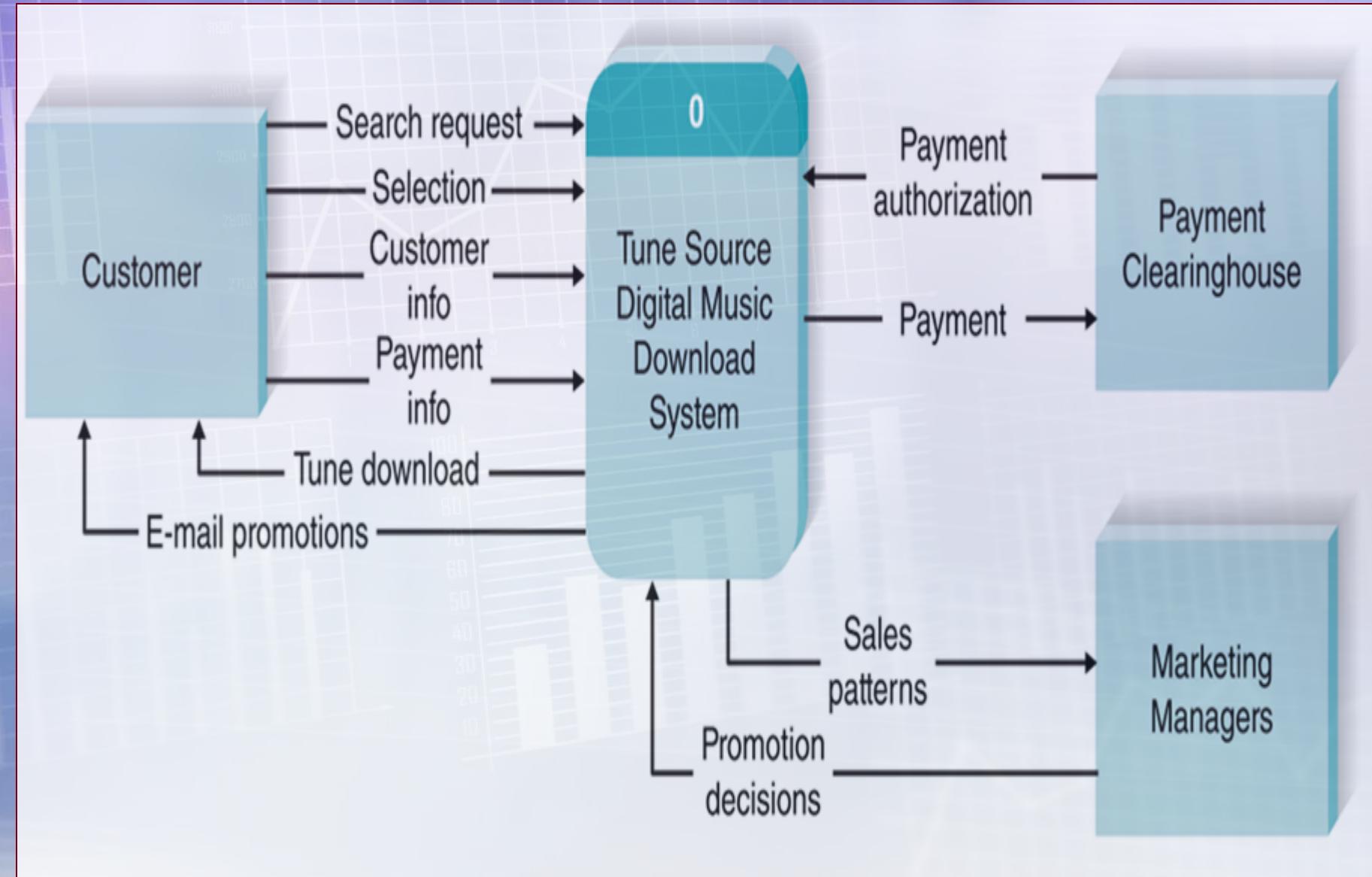
No, these are data flows.



Business Process Analysis 101

Digital music downloads system
(e.g. iTunes).

The process is in the middle,
external entities around the sides,
and data flows represented by
arrows in the diagram.



Technology Enablement

There are many manual processes that work just fine. On the other hand, a manual process, which causes a lot of pain, can be a candidate for automation and analysis.

- 
1. It might as well be 1950
 2. The process uses automated data in some steps
 3. Some steps of the process are automated
 4. The main process flow is automated
 5. The process and most exceptions are automated

Where do we focus in this course?



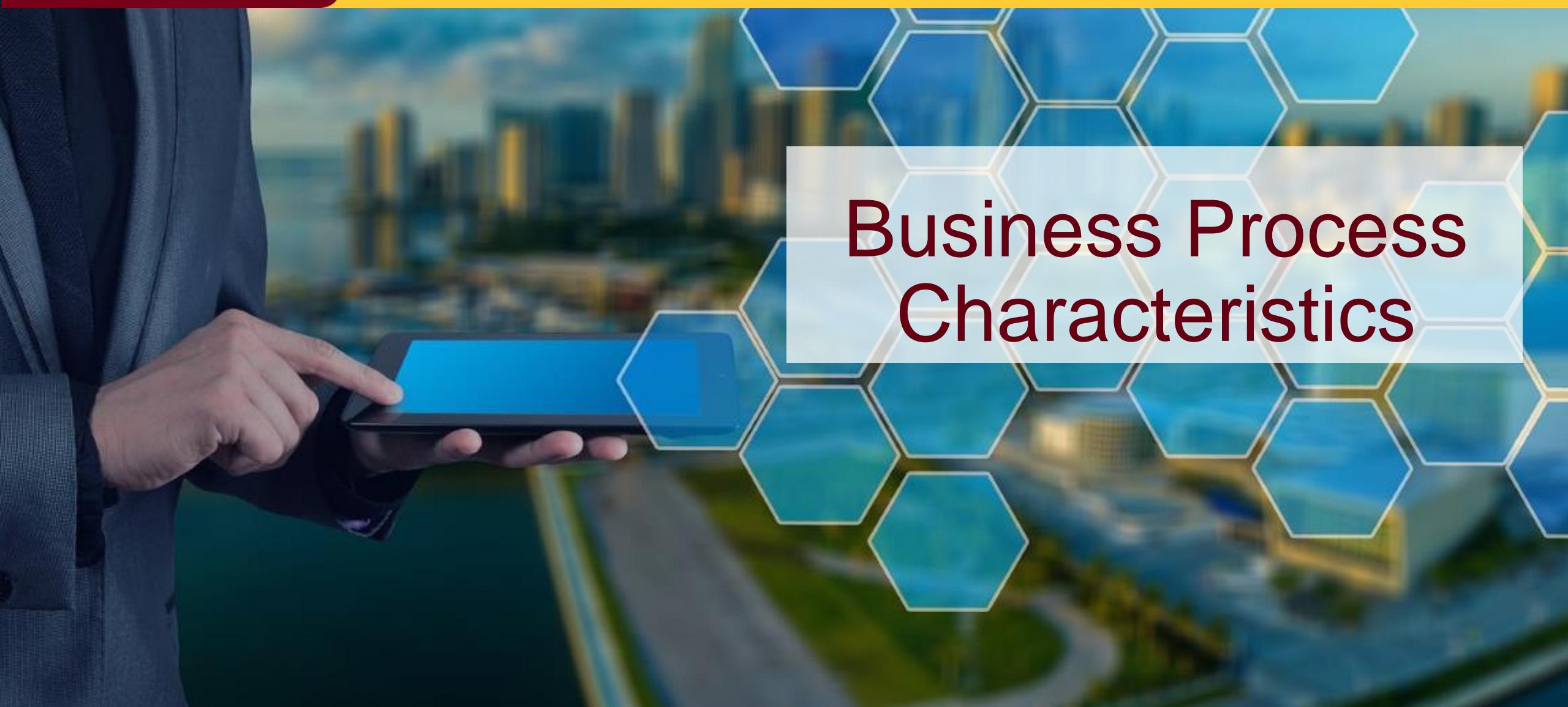
CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Image Credits

<u>Image/Source</u>	<u>Copyright</u>	<u>Location</u>	<u>Notes</u>
Business Process Analysis 101	Systems Analysis and Design, Figure 5 – 16 (p. 178)	Slide 6	

Business Process Characteristics



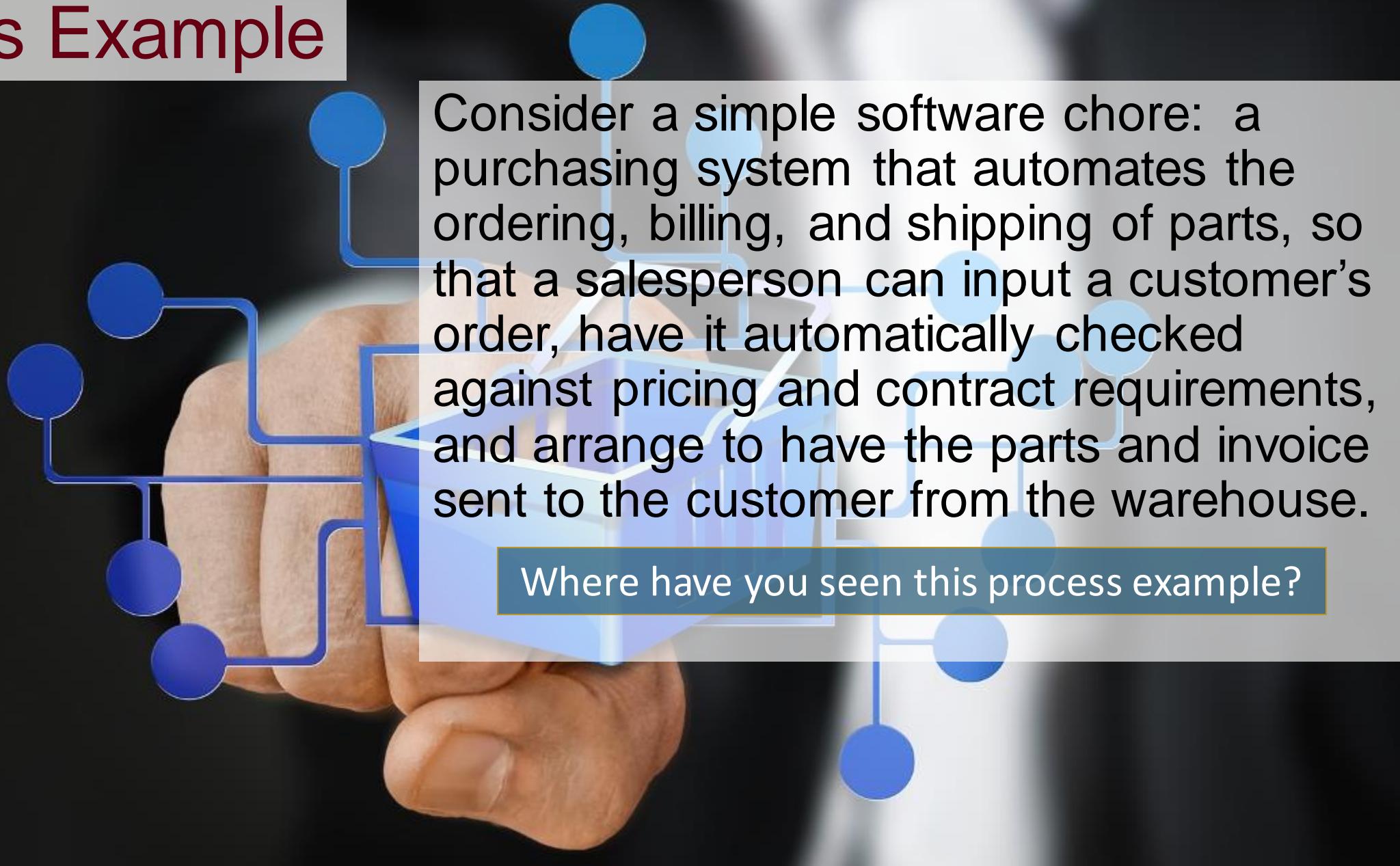
Learning Objectives

A close-up photograph of a white and blue soccer ball that has just been kicked into a dark-colored goal net. The ball is positioned on the left side of the frame, with the net filling the background.

After this lesson, you will be able to...

- Define a business process
- Describe the components of a business process
- Explain the varying levels of automation with respect to business processes

Process Example

A close-up photograph of a person's hand reaching towards a blue process flow diagram. The diagram consists of several blue circular nodes connected by blue lines, forming a network. The hand is positioned as if it is about to touch or interact with one of the nodes.

Consider a simple software chore: a purchasing system that automates the ordering, billing, and shipping of parts, so that a salesperson can input a customer's order, have it automatically checked against pricing and contract requirements, and arrange to have the parts and invoice sent to the customer from the warehouse.

Where have you seen this process example?

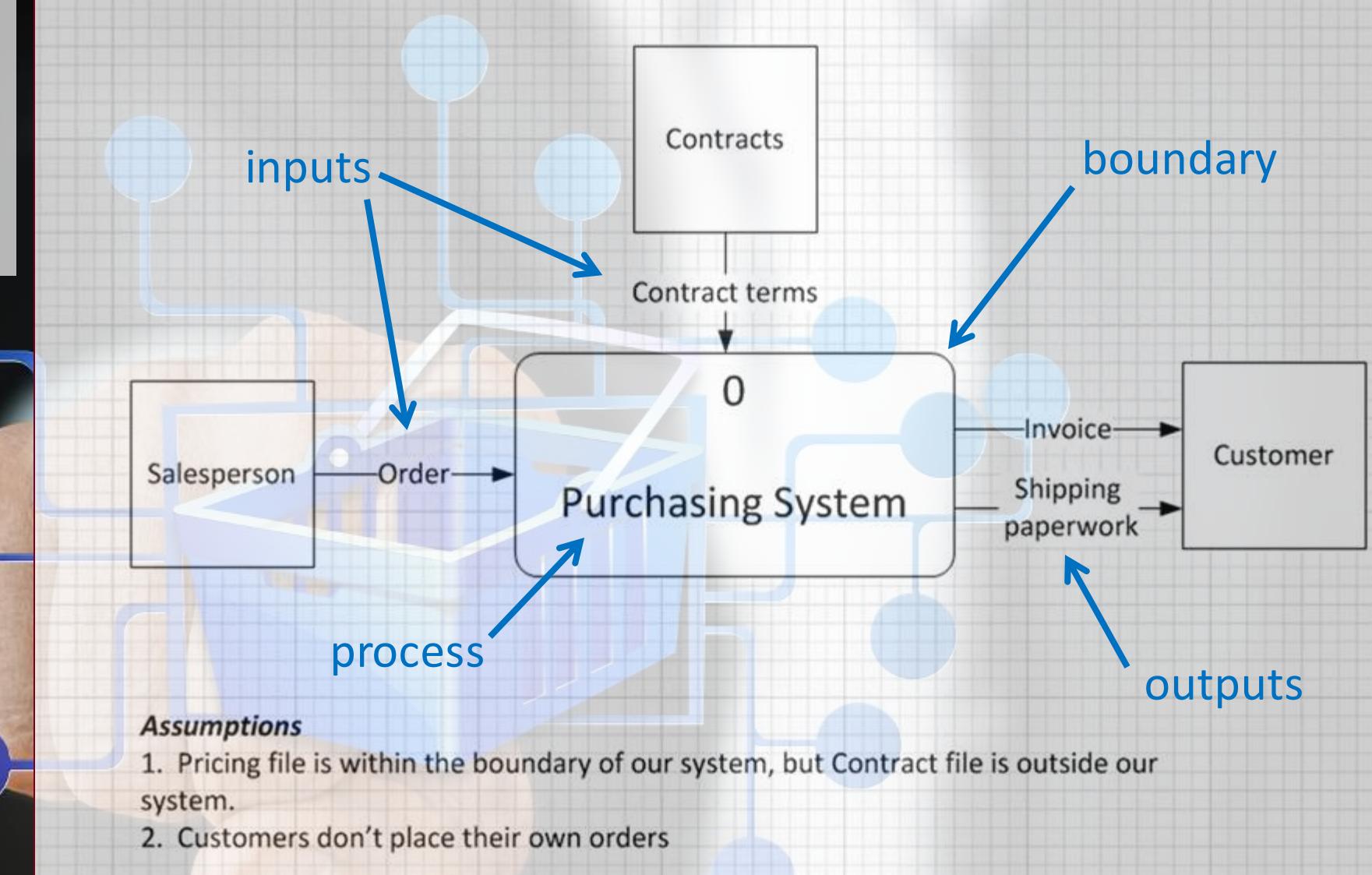
Process Components & Characteristics

A context diagram shows the system and external entities as rectangular boxes.

A process has inputs. In this case, the salesperson is entering the order and contracts department might be inputting the contract terms such as net 30 or net 60. Also the process does some processing, and that's represented by the central box here, the purchasing system.

The pricing file is within the boundary of our system but the contracts file is not.

We assume that customers don't place their own orders.

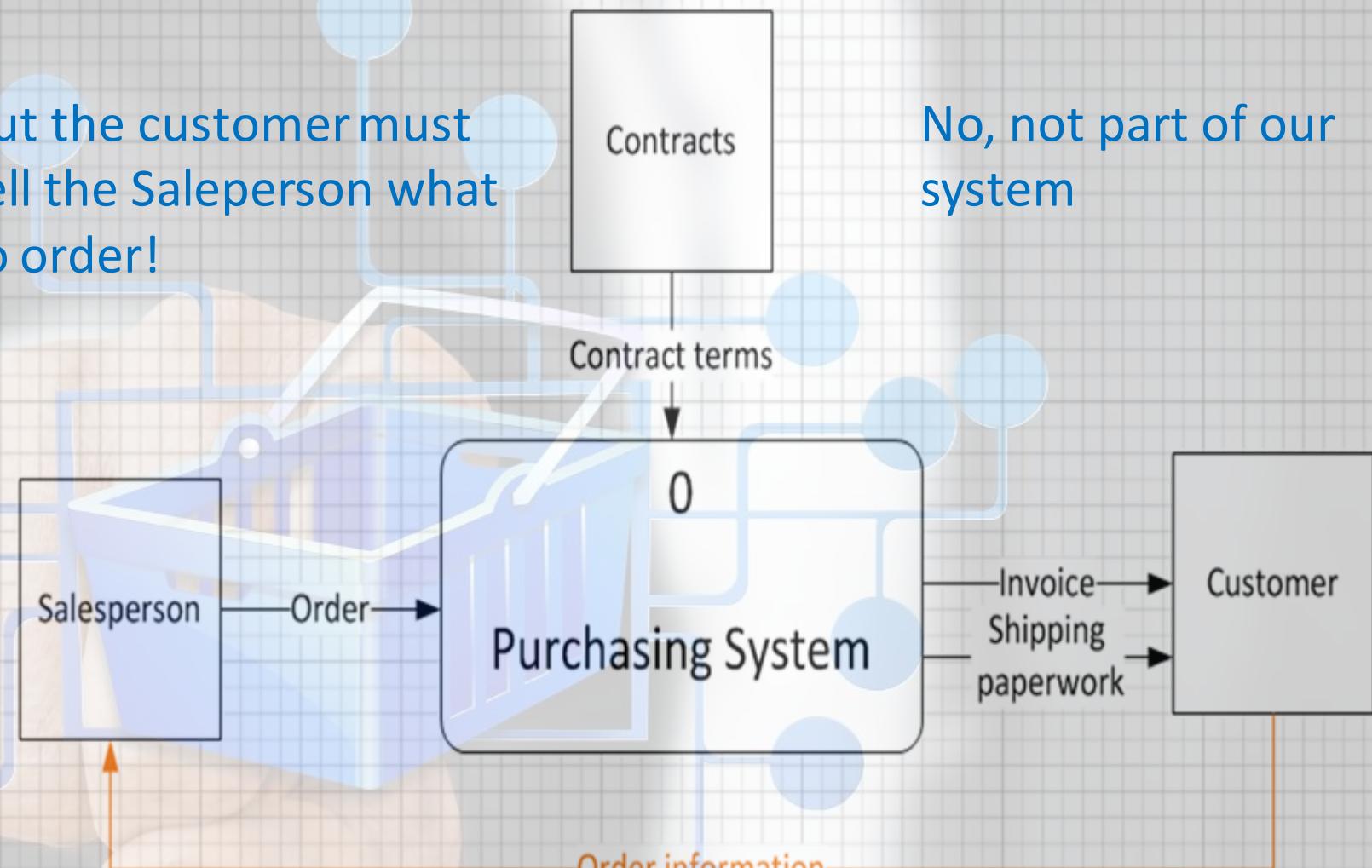


Process Components & Characteristics

We made an assumption that all orders are placed by a salesperson, even if the customer is communicating with them by phone or email.

But the customer must tell the Saleperson what to order!

No, not part of our system



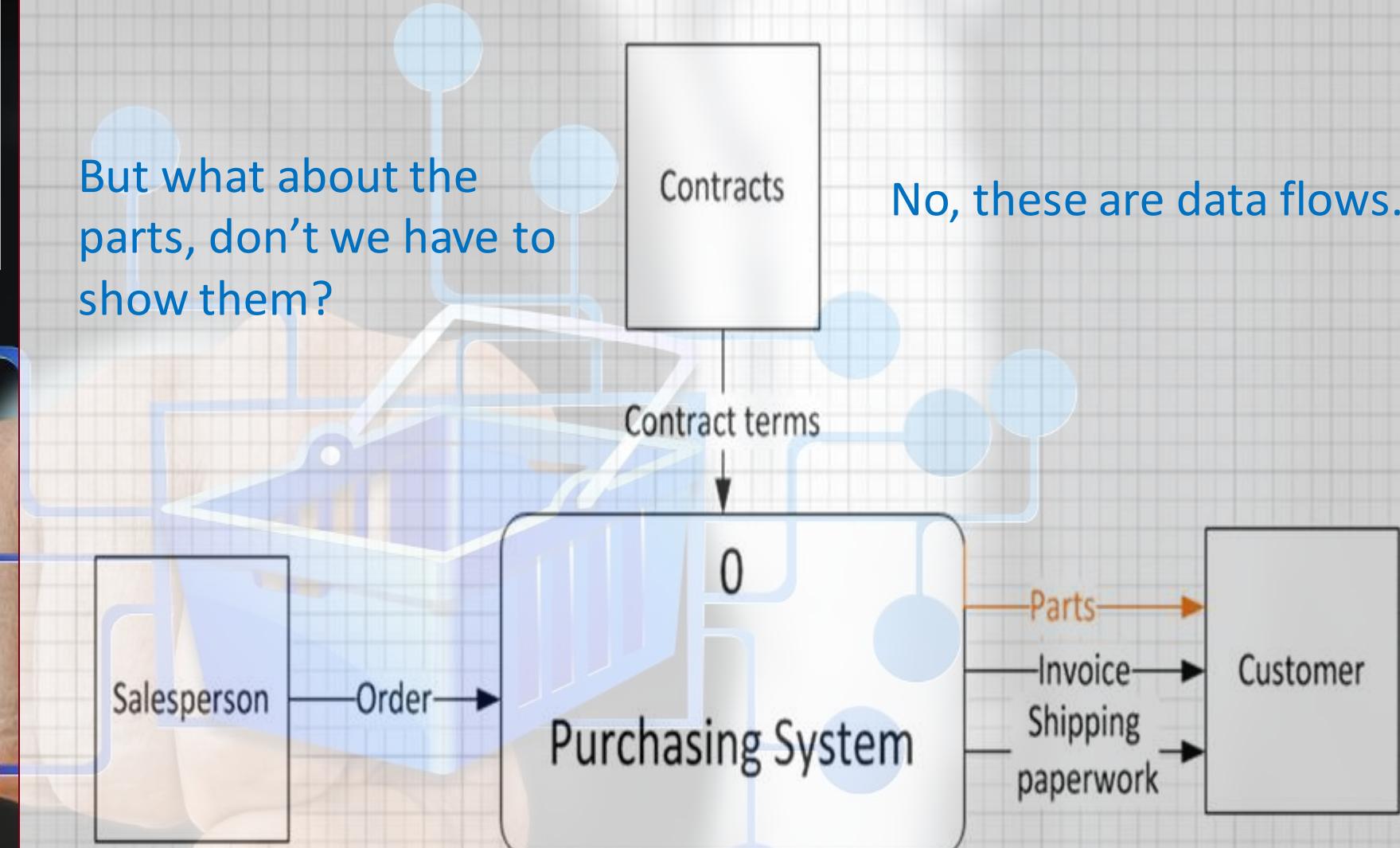
Process Components & Characteristics

Do we have to show physical goods in our diagram?

Generally we do not because these diagrams represent data flows relative to the process and not necessarily the physical goods that are moving around.

But what about the parts, don't we have to show them?

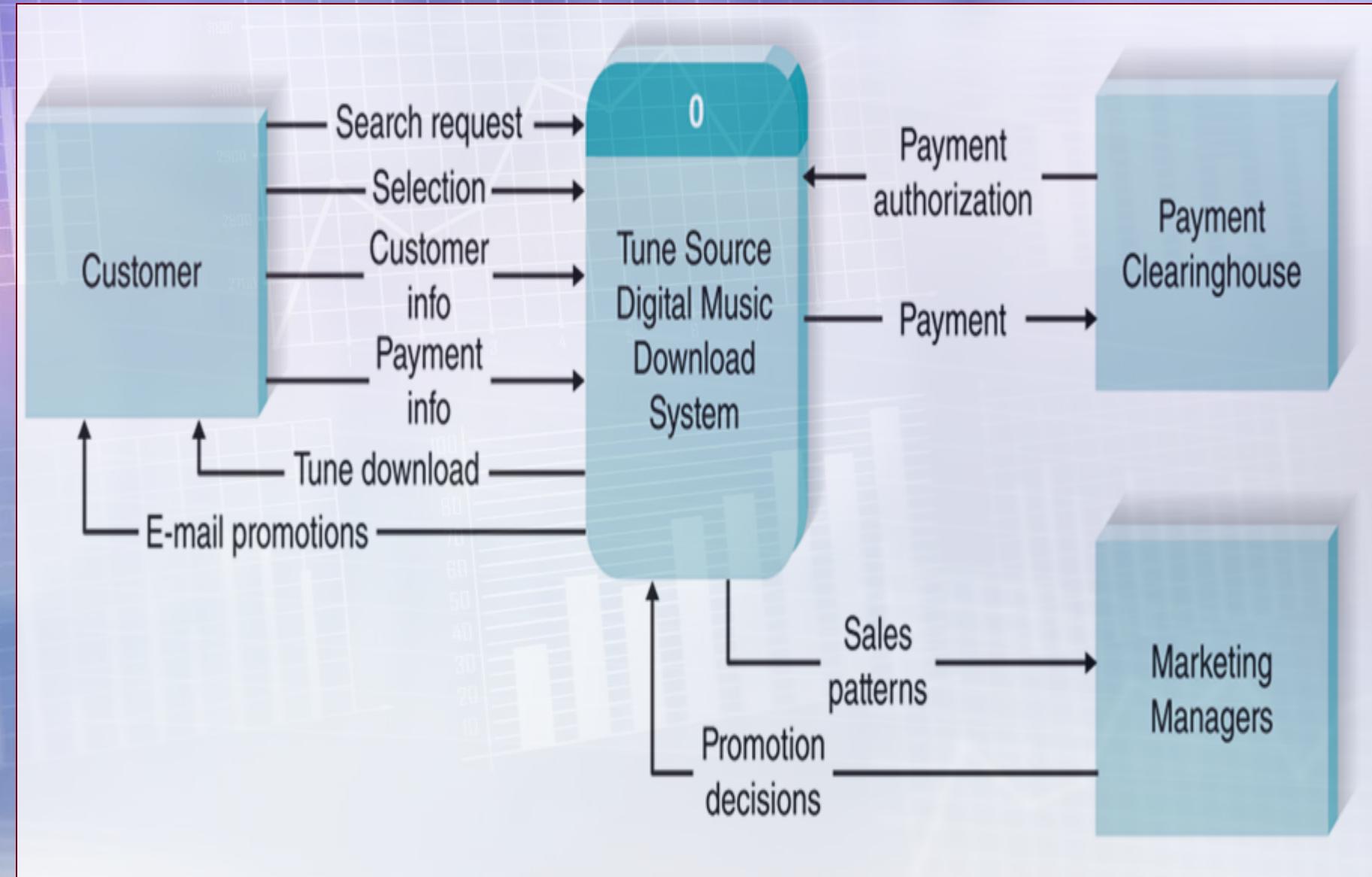
No, these are data flows.



Business Process Analysis 101

Digital music downloads system
(e.g. iTunes).

The process is in the middle,
external entities around the sides,
and data flows represented by
arrows in the diagram.



Technology Enablement

There are many manual processes that work just fine. On the other hand, a manual process, which causes a lot of pain, can be a candidate for automation and analysis.

- 
1. It might as well be 1950
 2. The process uses automated data in some steps
 3. Some steps of the process are automated
 4. The main process flow is automated
 5. The process and most exceptions are automated

Where do we focus in this course?



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Image Credits

<u>Image/Source</u>	<u>Copyright</u>	<u>Location</u>	<u>Notes</u>
Business Process Analysis 101	Systems Analysis and Design, Figure 5 – 16 (p. 178)	Slide 6	



Learning Objectives

A close-up photograph of a white and blue soccer ball that has been kicked into a dark-colored goal net. The ball is positioned in the upper left quadrant of the frame, with the net's mesh visible in the background.

After this lesson you will be able to...

- Describe the phases of the systems development lifecycle (SDLC)
- Explain the history of the SDLC
- Describe how the organization provides context for the SDLC
- List the characteristics of a process-oriented organization

The SDLC Methodology

Inefficient process

Technology enabled process

- Why?
- Id problem
- Calculate value
- Draft an approach
- Create the plan

Plan

- What?
- Clarify requirements
- Understand current process
- Develop changes to business process

Analyze

- How?
- Translate requirements into system specifications

Design

- Program and install automation
- Modify processes
- Ensure change is managed

Implement

The reference SDLC that we will use for this course has four phases: plan, analyze, design, and implement.

The SDLC Provides Structure

Inefficient process

Technology enabled process

- Why?
- Id problem
- Calculate value
- Draft an approach
- Create the plan

Plan

- What?
- Clarify requirements
- Understand current process
- Develop changes to business process

Analyze

- How?
- Translate requirements into system specifications

Design

- Program and install automation
- Modify processes
- Ensure change is managed

Implement

The business analyst does most of his/her work in the analyze or analysis phase.

SDLC History

To move a waterfall methodology into a supercharged waterfall, we add the following enhancements:

- Prototypes
- Joint Application Development (JAD) sessions
- CASE tools or computer-aided software engineering tools.

Rapid Application Development (RAD) is a family of methodologies that introduced the concept of iteration into the Waterfall methodology.

With a RAD type methodology, a part of the system is developed and then is shown to users to interact with the system and provide feedback. That feedback is used to further refine and enhance the system, and then we put it in front of the users for feedback again. This cycle until we feel like we have something that we're ready to introduce.

Agile addresses some of the rigidity and some of the shortcomings of traditional waterfall methodologies. Agile focuses much more on values than it does on process and methodology. Agile favors (i) individuals and interactions over processes and tools, (ii) working software over documentation, and (iii) responding to change over following a plan.

Before 1985

- Ad hoc (no process)

1985 – now

- Waterfall methodology

1990 – now

- Super-charged waterfall

1995 - now

- RAD methodologies

2000 – now

- Agile methodologies



Waterfall Characteristics

Waterfall Model Characteristics

1- Very well-defined phases with cutoff artifacts (documents) that denote the end of each phase and the transition into the next phase.

2-Inflexible or has little iteration between phases.

3- Flawed

The plan phase produces a complete project plan that is often a flawed one.

Waterfall Methodology Misconceptions

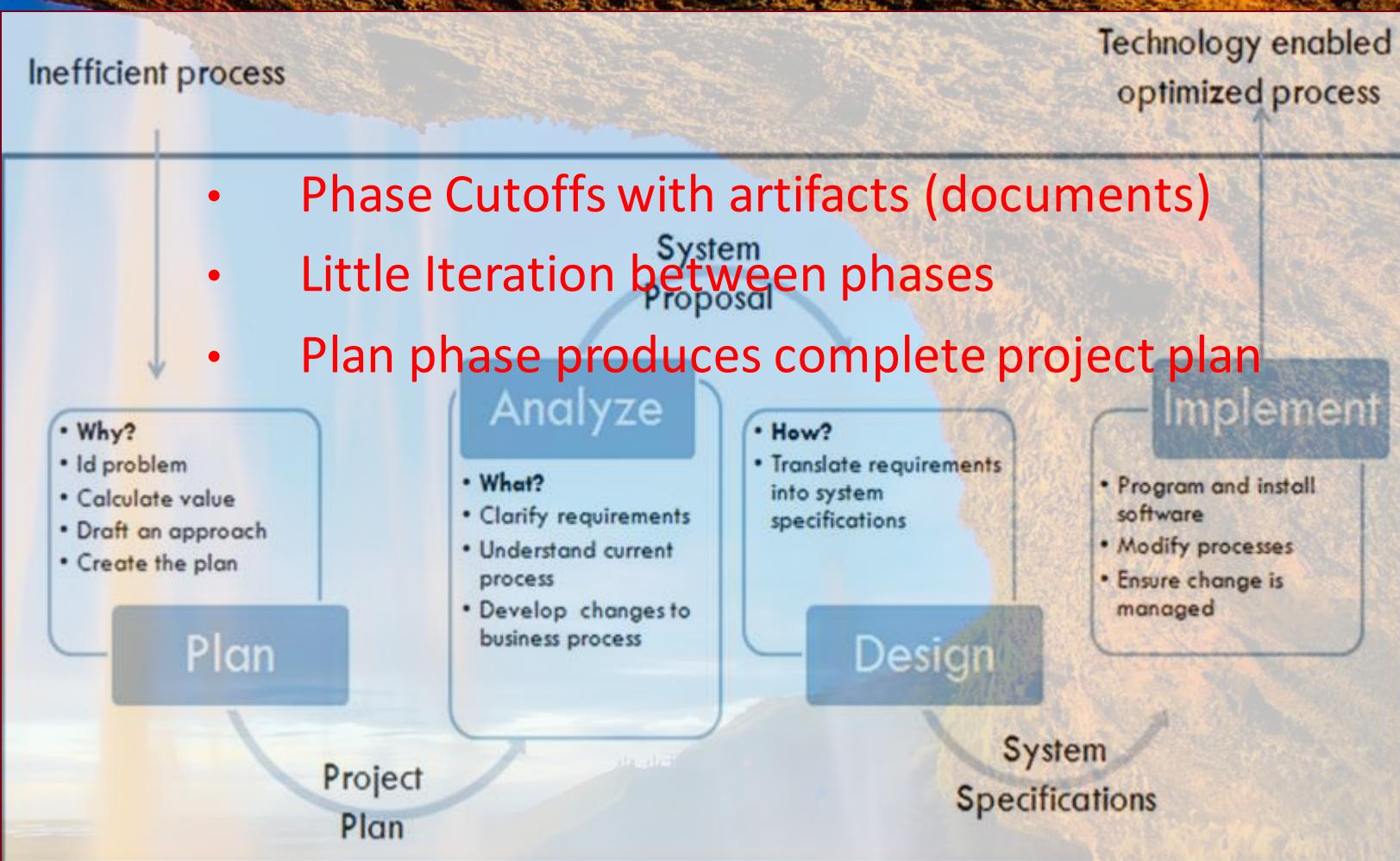
1- When do we do the programming?

This would typically happen in what you see on your screen as the last phase or the implement phase where you see program and install software.

2- Purchasing software off the shelf.

Why do we need to do SDLC?

Since we do not if the software will meet our needs, the choice of buying or building often happens in the design phase.



The Organizational Environment Provides Context

The success of the waterfall approach often depends on the organizational environment in which it's being implemented.

Organization Types:

1- Vertical Organization (e.g. Texas Instruments)

Employees identify with Units

2- Process oriented Organization

- Employees identify with Processes

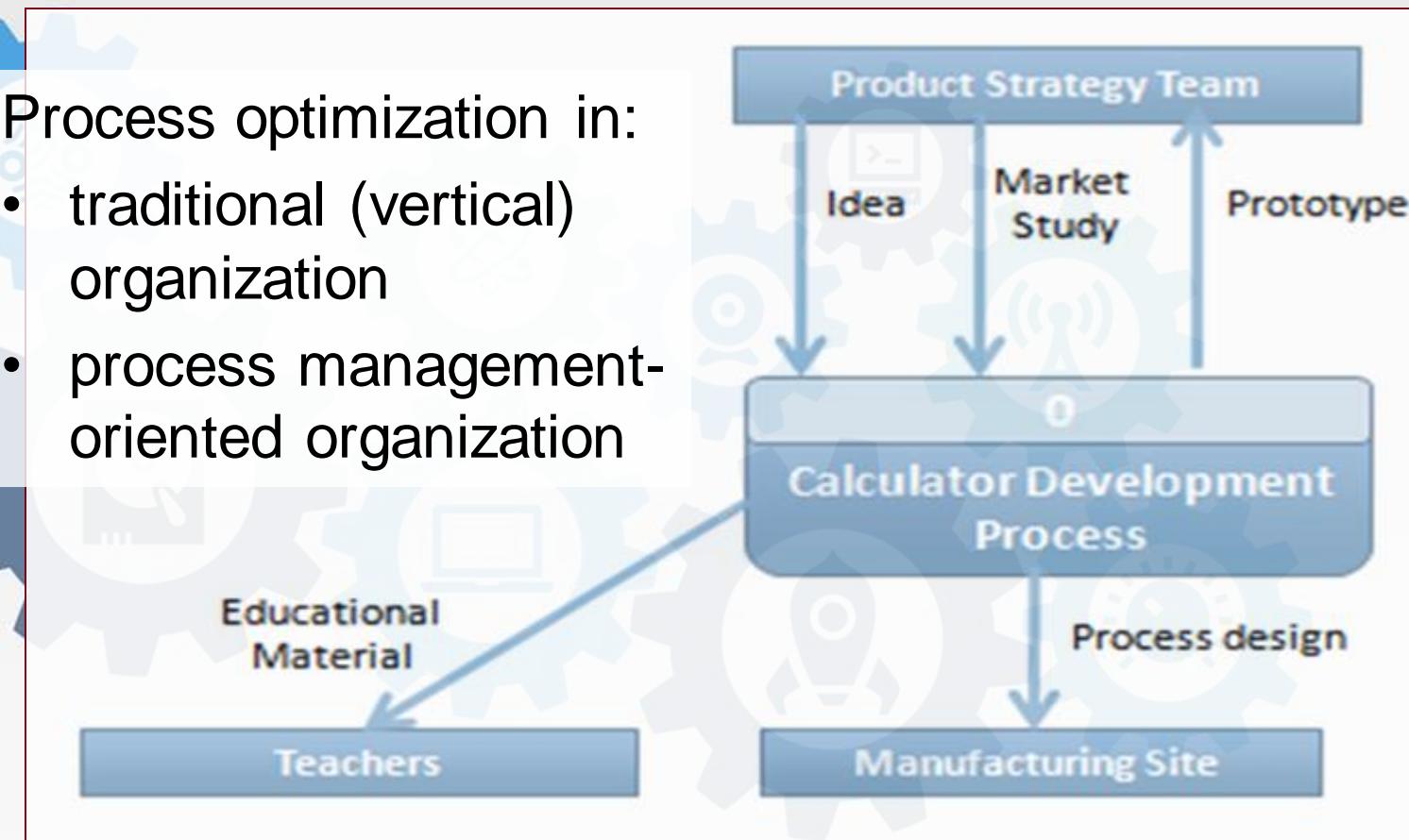
- A process owner owns a certain process.
- A decision rights matrix is used to clearly identify the roles between the functional manager who manages the employees, and the process owner who owns the process itself.

Waterfall is likely to be more successful in organizations that have a rich process organization history.



Process optimization in:

- traditional (vertical) organization
- process management-oriented organization



The Organizational Environment Provides Context

The Context Diagram for the Calculator Development Process at Texas Instruments.

They implemented a cross-functional process to develop these calculators, and it was designed to get input from teachers, as well as feedback for the manufacturing sites on the feasibility of different designs.

That process ultimately failed within TI's organization. Why?

TI had a very traditional hierarchical functional organization, rather than a process management-oriented organization.

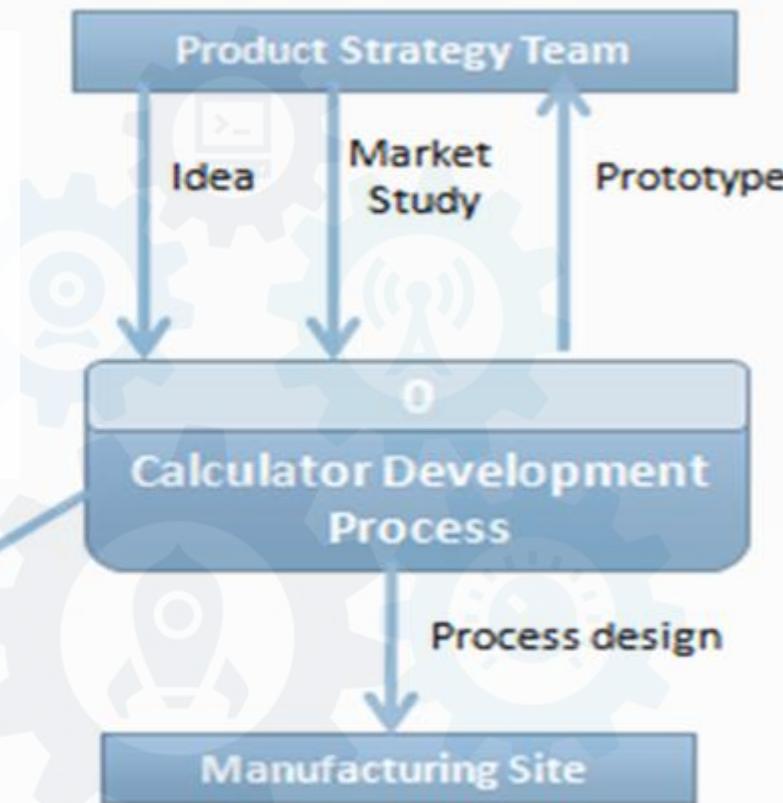
Process standardization does not imply that all business units should use the same process.

Process-oriented organizations often struggled when tried to apply processes into units where they might not apply.

The shift from a traditional environment to a process-oriented environment cannot happen quickly.



- How can we recognize a process-oriented enterprise?
- What are the downsides?



The Organizational Environment Provides Context



1. In a process-oriented organization:
 - I. A _____ owns each process.
 - II. The ***decision rights matrix*** clearly identifies roles between the _____ and the _____
 - III. The ***question of process standardization*** implies that all business units should use the same process? (T/F)
2. The shift from a traditional environment to a process-oriented environment can happen quickly (T/F)



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA



Learning Objectives



After this lesson you will be able to:

1. List the major variants of SDLC
2. Compare and contrast RAD with Waterfall
3. Describe the relationship between Waterfall and Agile
4. Identify the methodology used in your organization

SDLC Maturation

Before 1985

- Ad hoc (no process)

1985 – now

- Waterfall methodology

1990 – now

- Super-charged waterfall

1995 - now

- RAD methodologies

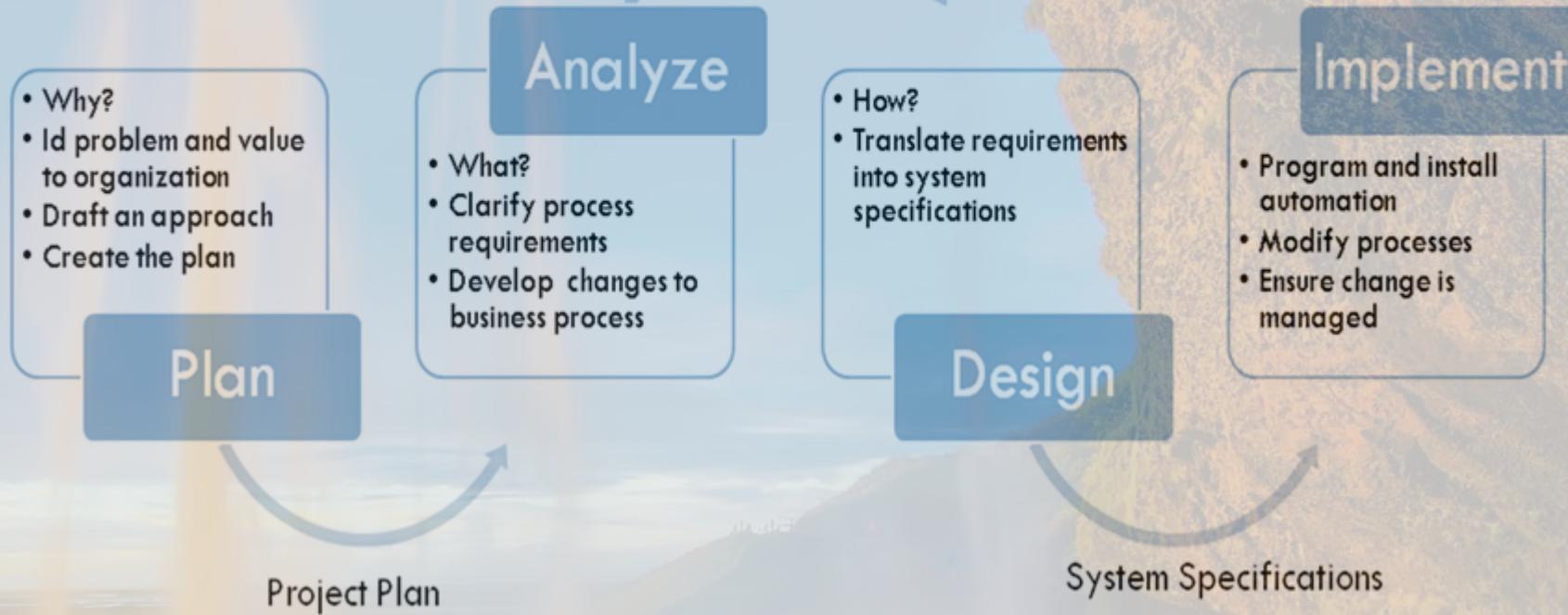
2000 – now

- Agile methodologies

Waterfall Methodology

The Waterfall methodology—also known as the Waterfall model—is a sequential development process that flows like a waterfall through all phases of a project (plan, analyze, design, implement, for example), with each phase completely wrapping up before the next phase begins.

What characteristics indicate Waterfall Methodology?



SDLC Maturation

Before 1985

- Ad hoc (no process)

1985 – now

- Waterfall methodology

1990 – now

- Super-charged waterfall

1995 - now

- RAD methodologies

2000 – now

- Agile methodologies

Supercharged Waterfall

To move a waterfall methodology into a supercharged waterfall, we add the following enhancements:

- Prototypes
- CASE tools or computer-aided software engineering tools.
- Joint Application Development (JAD) sessions

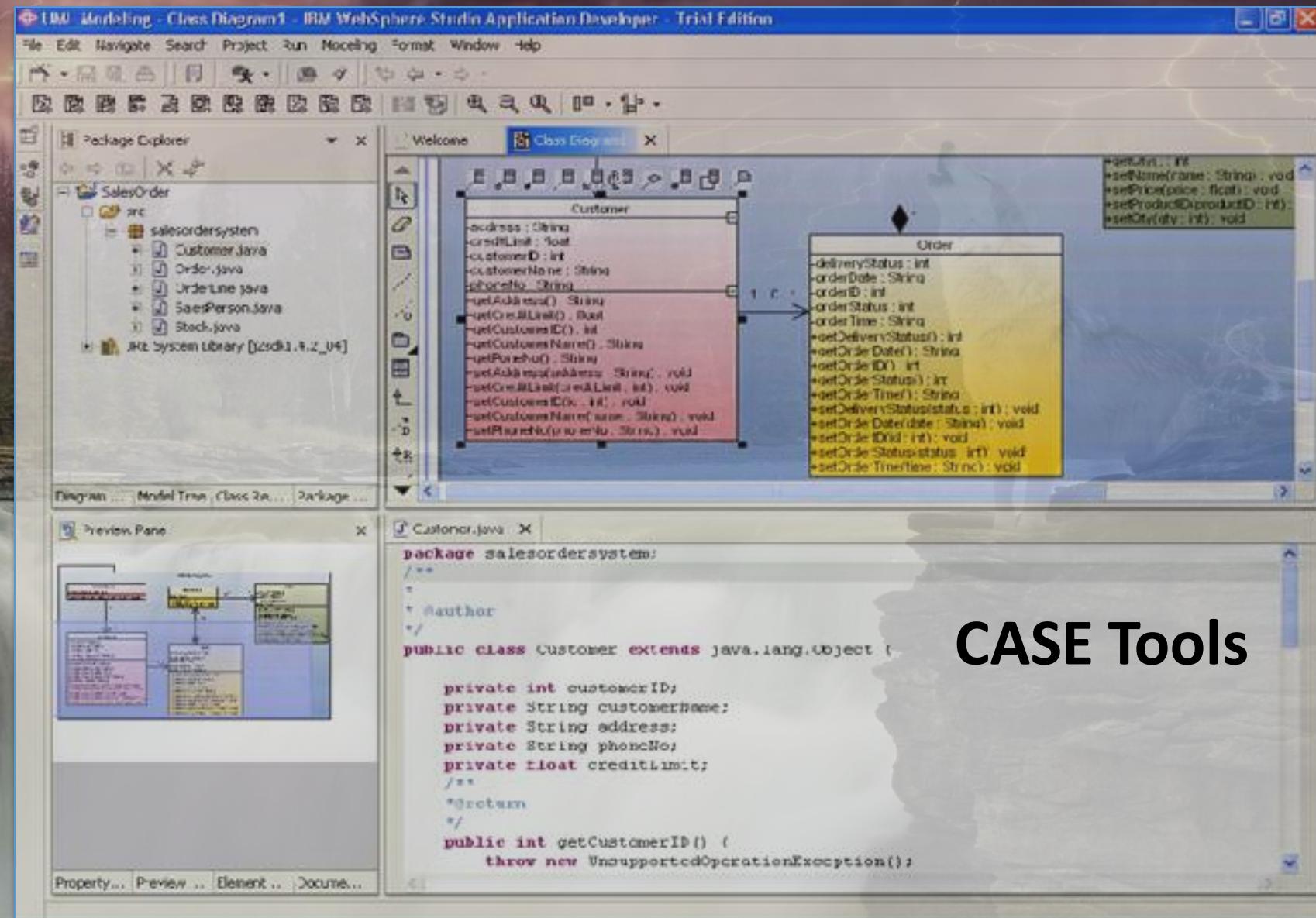
The screenshot shows a Mozilla Firefox browser window with a travel website titled "My Travel Site". The left sidebar displays a navigation tree with categories like "My Travel Site", "Sign In Scenario", "Search Scenario", and "Home". The main content area features a "Welcome!" message and a search form for "Destination", "Check-in", and "Check-out". Below this are fields for "Guests" (Rooms 1-3) and "Adults" and "Children" (Room 1 and Room 2). A "Search" button is present. To the right, there's a "Sign In Button" callout with details: Specification: Links to Sign in page, Status: Proposed, Benefit: Important. Below the search form, there's a placeholder text "Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut sed mauris. Aenean sagittis. ante". At the bottom, a "Home" section describes the page as the first where users come to reserve their hotel, rental car, and/or flight. The background of the slide features a waterfall image.

Prototypes

Supercharged Waterfall

- CASE tools or computer-aided software engineering tools.

Example: Unified Modeling Language (UML), which is a standardized language used for modeling information systems. UML provides a standard way to visualize the design of a system.



CASE Tools

Supercharged Waterfall

To move a waterfall methodology into a supercharged waterfall, we add the following enhancements:

- Joint Application Development (JAD) sessions



Methodologies

Before 1985

- Ad hoc (no process)

1985 – now

- Waterfall methodology

1990 – now

- Super-charged waterfall

1995 - now

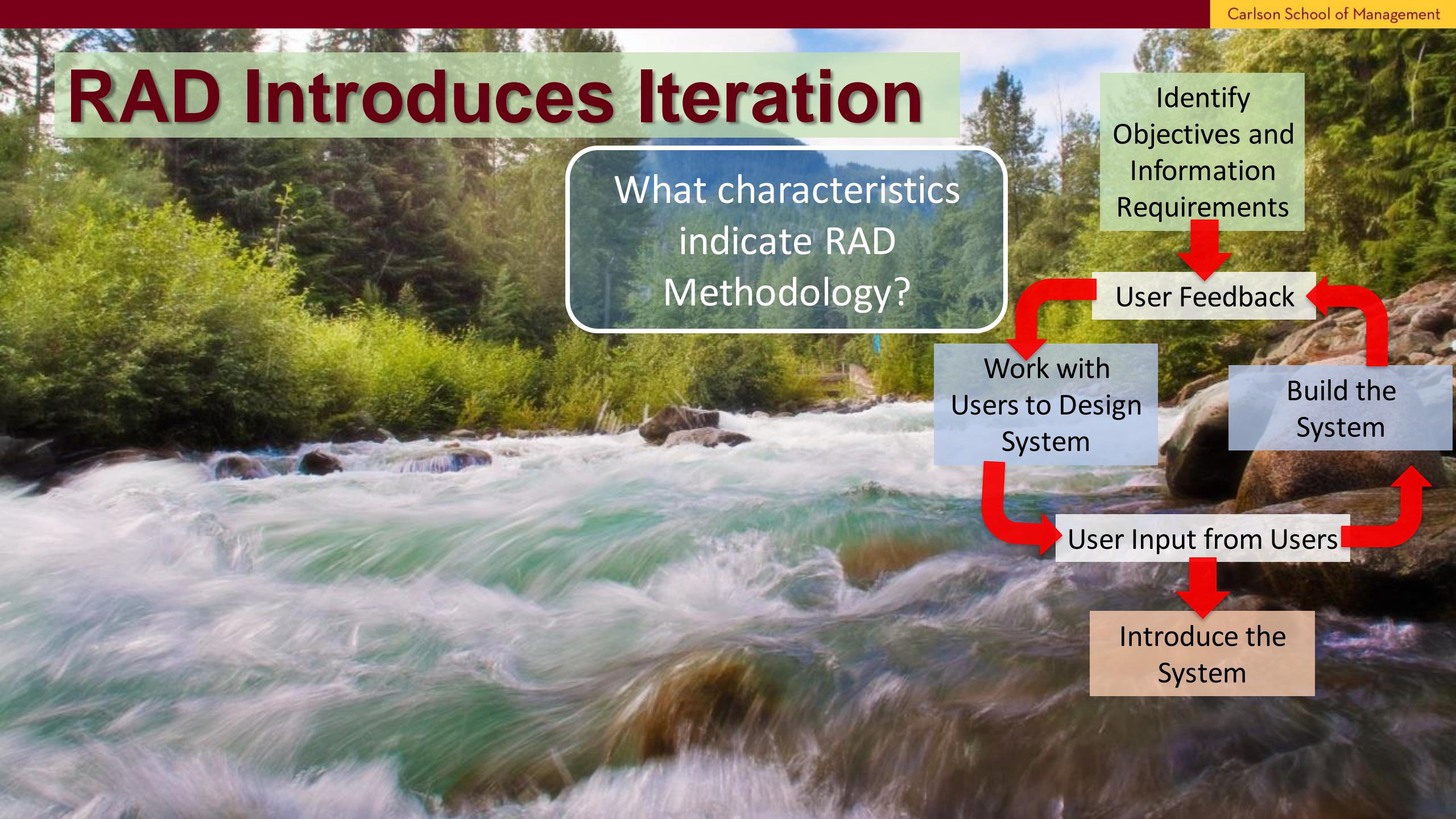
- RAD methodologies

2000 – now

- Agile methodologies

Review: how do we differentiate between waterfall and super-charge waterfall methodologies?

RAD Introduces Iteration



What characteristics indicate RAD Methodology?

Identify Objectives and Information Requirements

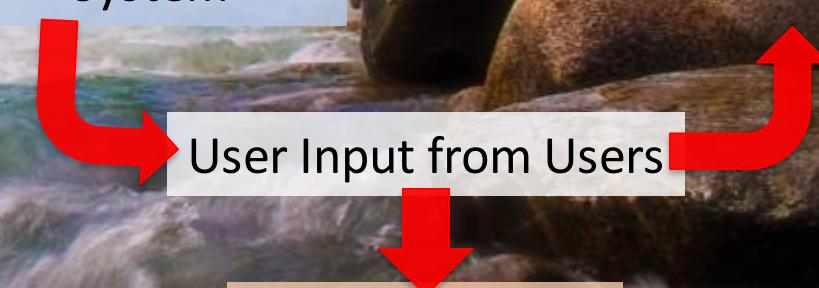
User Feedback

Work with Users to Design System

Build the System

User Input from Users

Introduce the System



RAD Introduces Iteration

There is another cycle of iteration!

Additional prioritized functions
(requirements)

Identify Objectives and Information Requirements

User Feedback

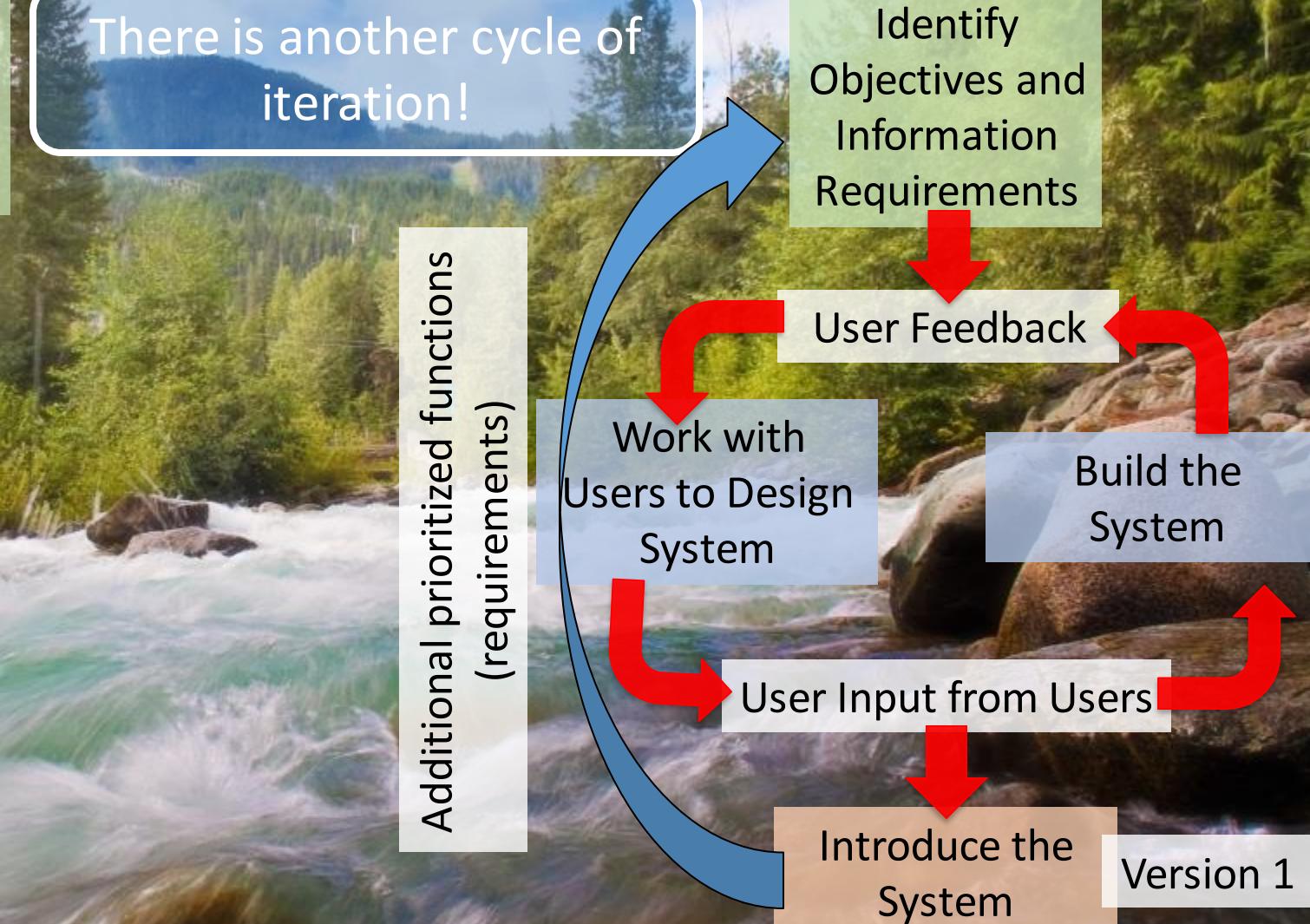
Work with Users to Design System

Build the System

User Input from Users

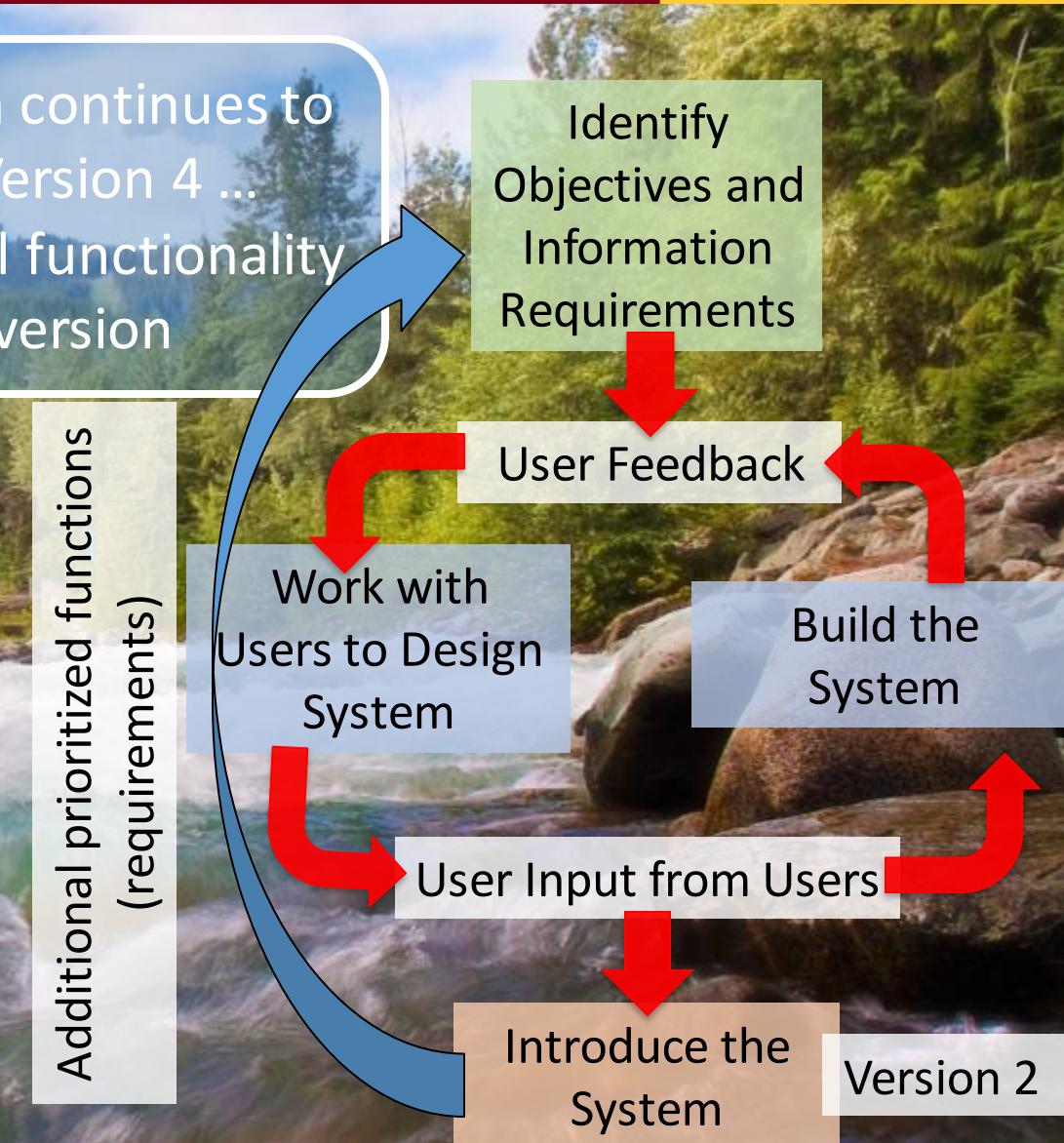
Introduce the System

Version 1

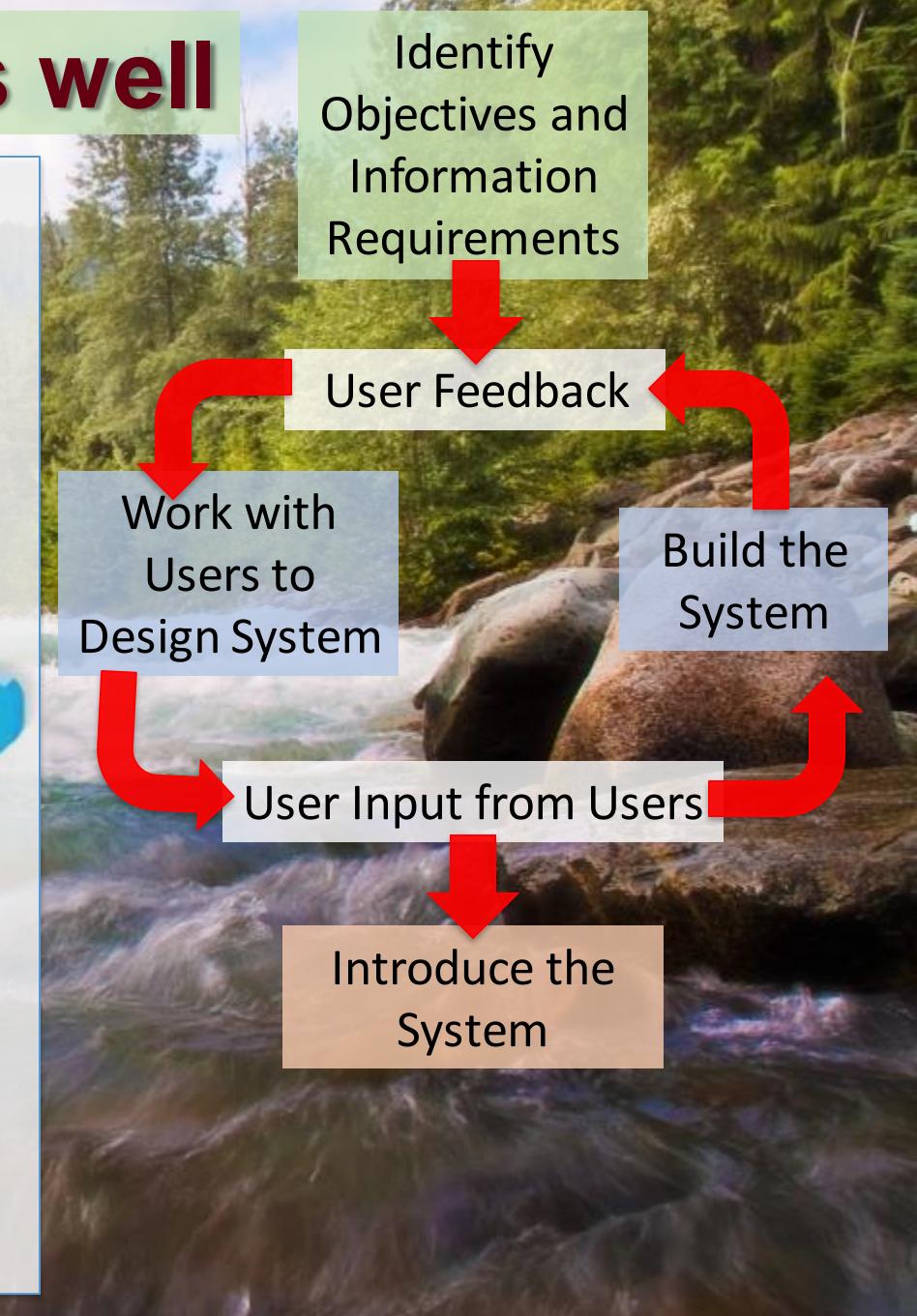
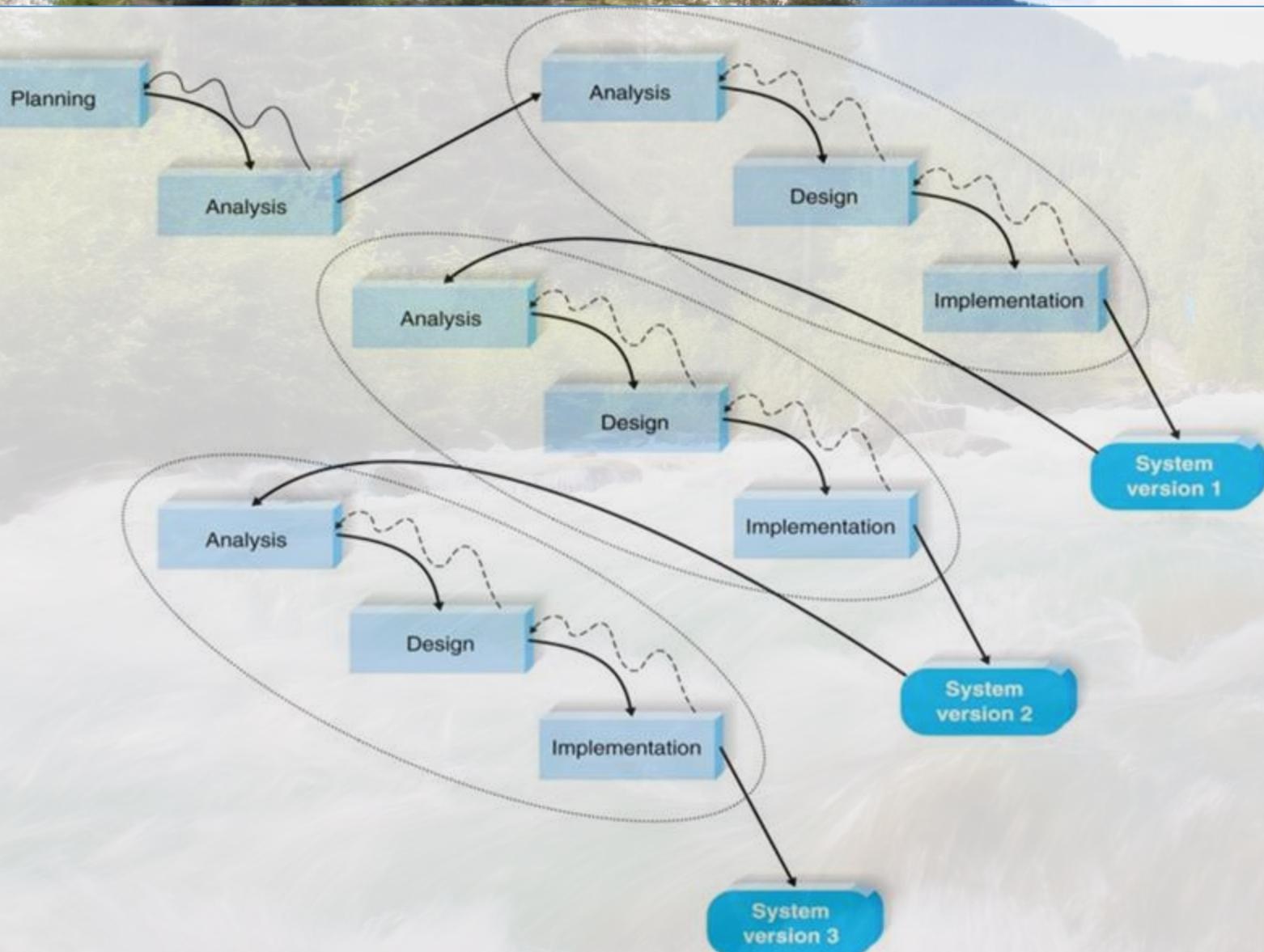


RAD Introduces Iteration

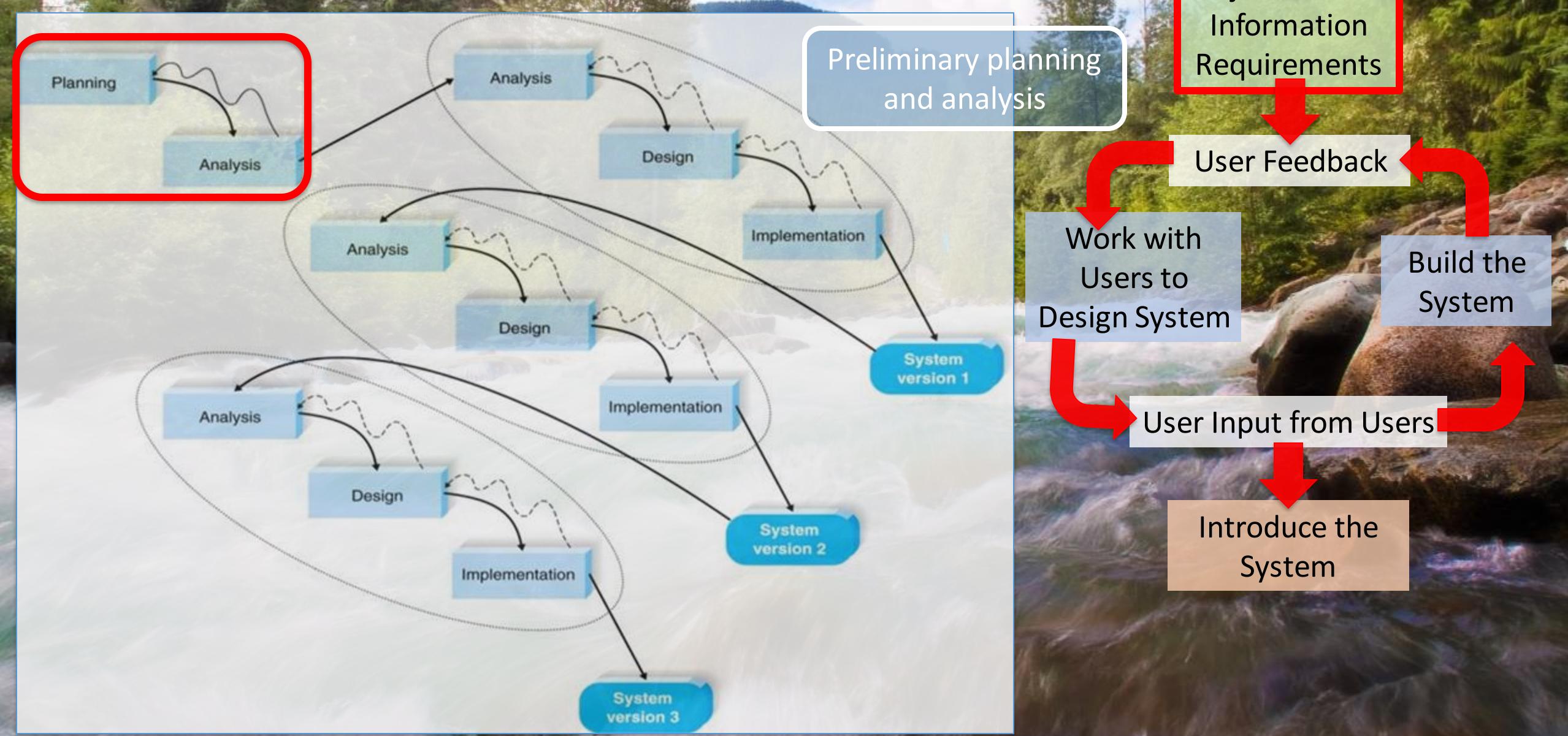
This cycle then continues to Version 3, Version 4 ...
With additional functionality at each version



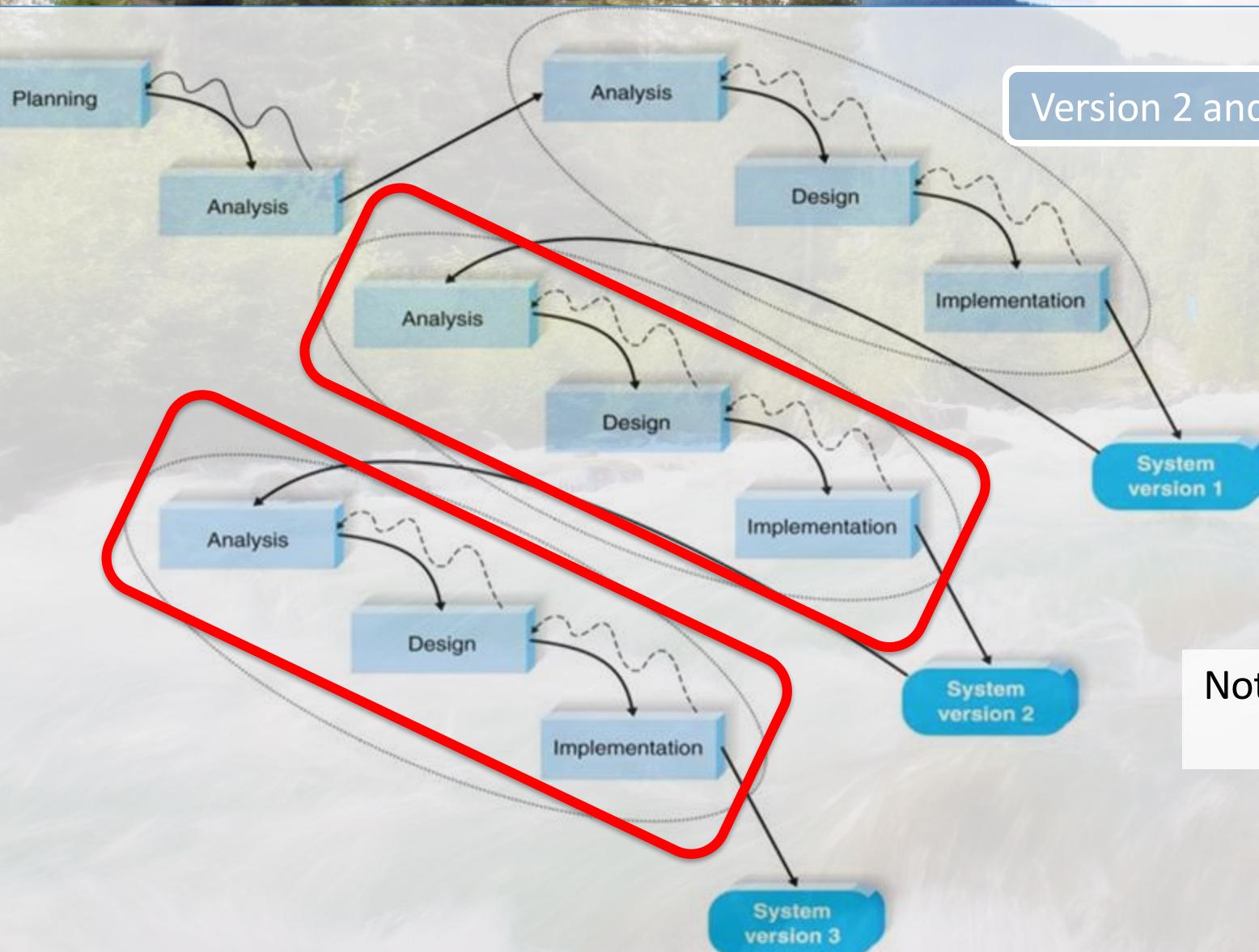
RAD: Your textbook illustrates this well



RAD: Your textbook illustrates this well



RAD: Your textbook illustrates this well



Identify
Objectives and
Information
Requirements

User Feedback

Work with
Users to
Design System

Build the
System

Not explicitly
shown

Introduce the
System

User Input from Users

Methodologies

Before 1985

- Ad hoc (no process)

1985 – now

- Waterfall methodology

1990 – now

- Super-charged waterfall

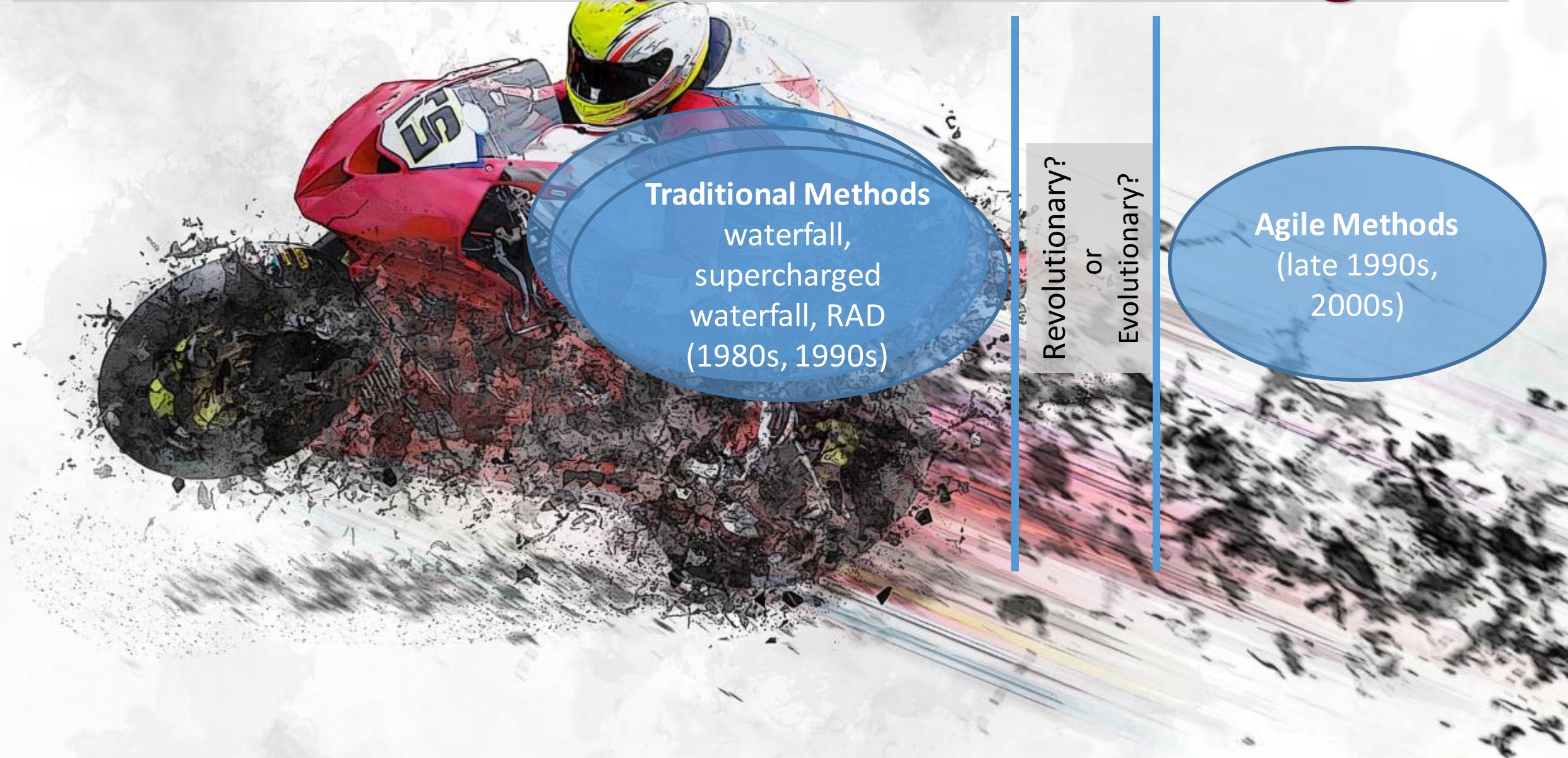
1995 - now

- RAD methodologies

2000 – now

- Agile methodologies

You need history to understand Agile



Traditional Methods
waterfall,
supercharged
waterfall, RAD
(1980s, 1990s)

Revolutionary?
or
Evolutionary?

Agile Methods
(late 1990s,
2000s)

Some consider Agile to be “revolutionary”

Values

MANIFESTO

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:
Individuals and interactions over **processes and tools**

Working software over **comprehensive documentation**

Customer collaboration over **contract negotiation**

Responding to change over **following a plan**

That is, while there is value in the items on the right, we value the items on the left more.

Agile Methods (late 1990s, 2000s)

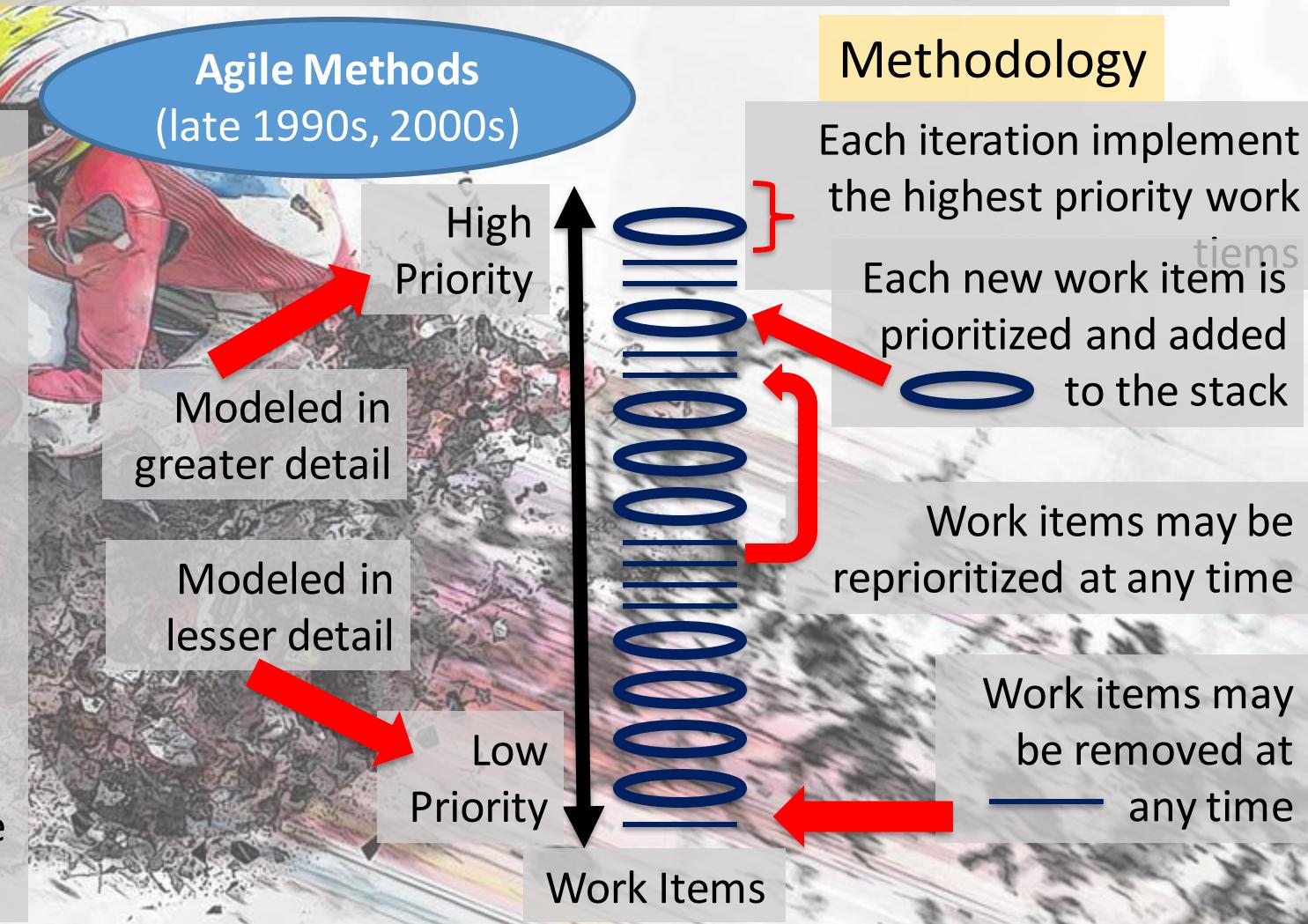
Methodology

Each iteration implement the highest priority work

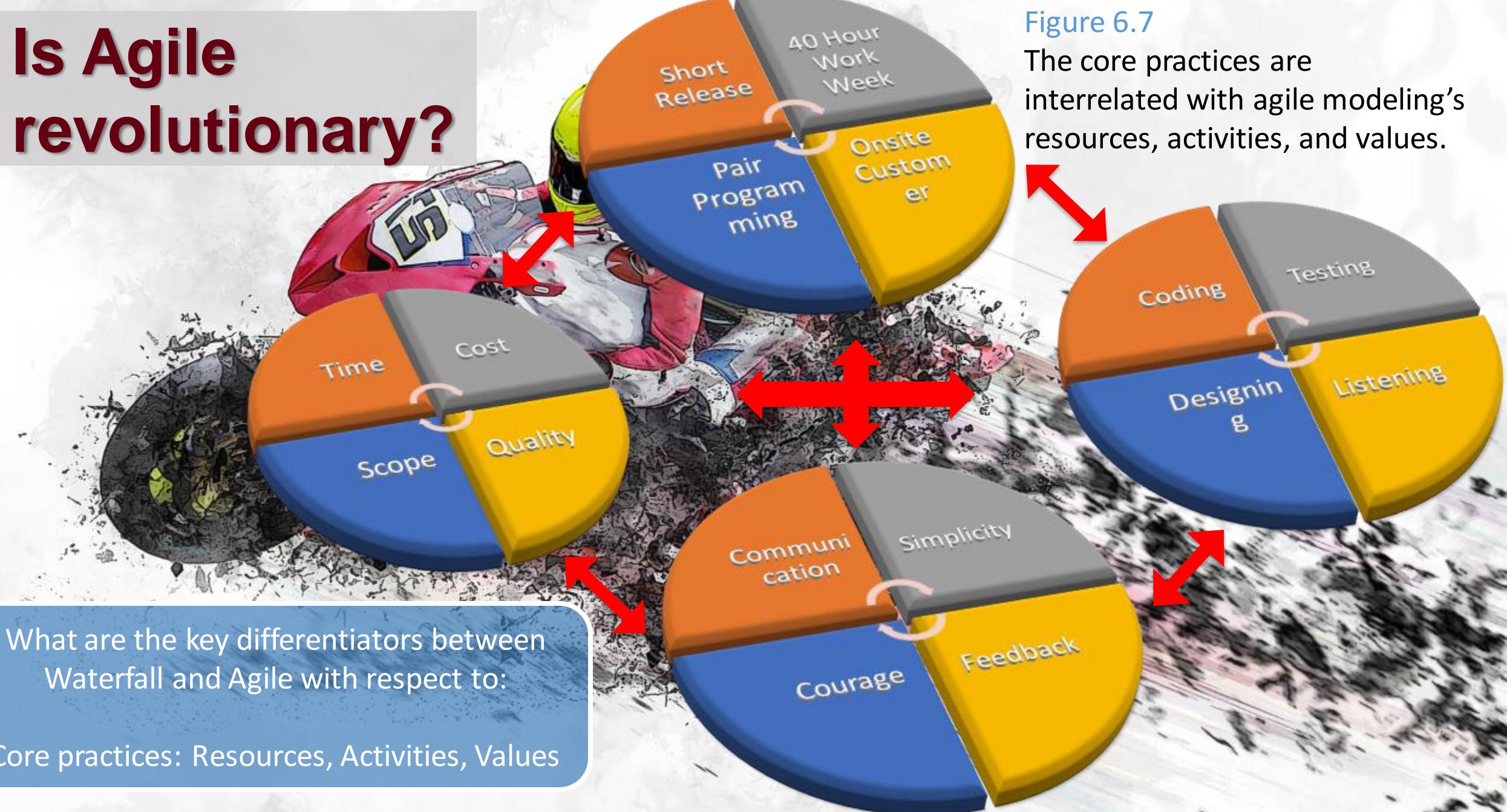
Each new work item is prioritized and added to the stack

Work items may be reprioritized at any time

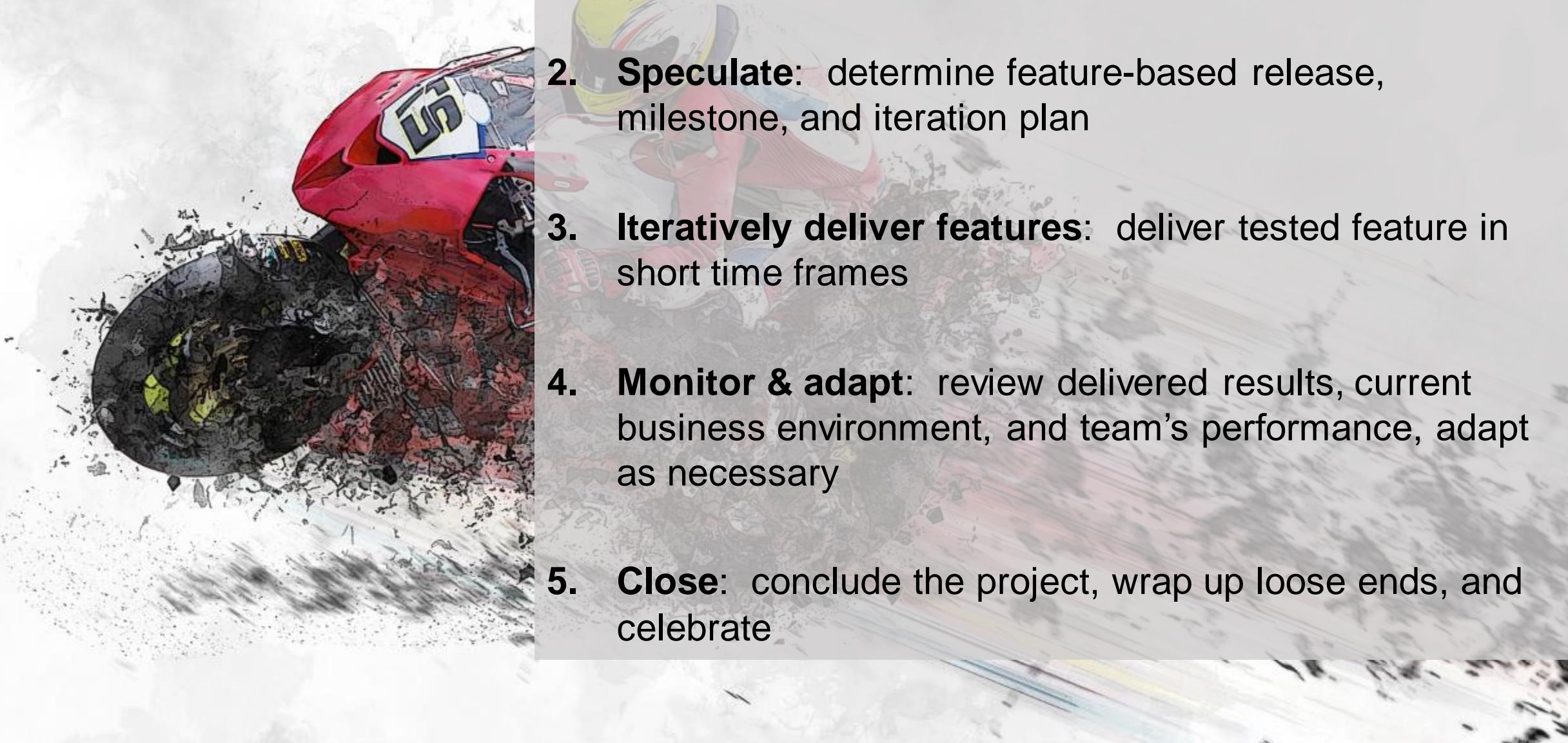
Work items may be removed at any time



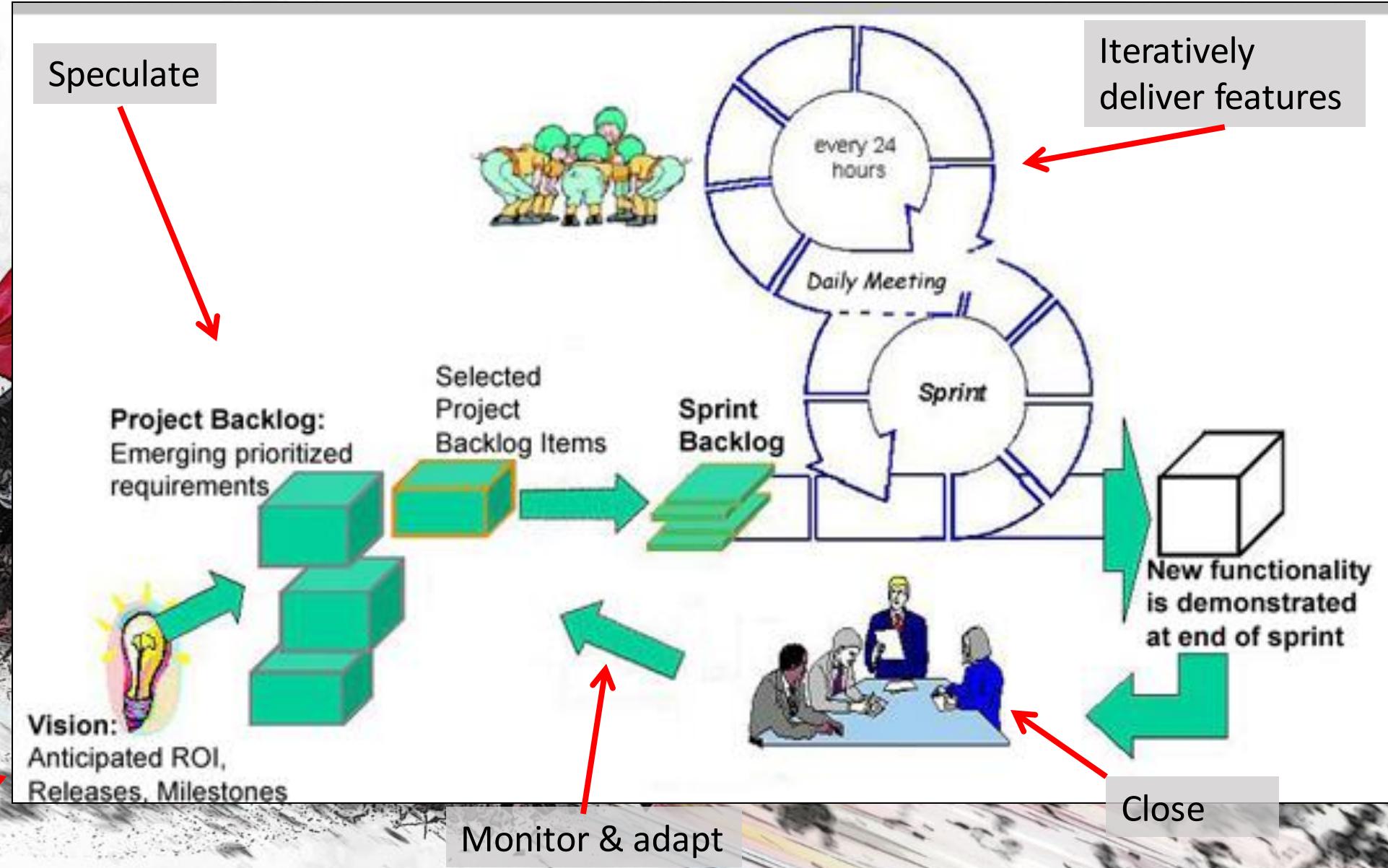
Is Agile revolutionary?



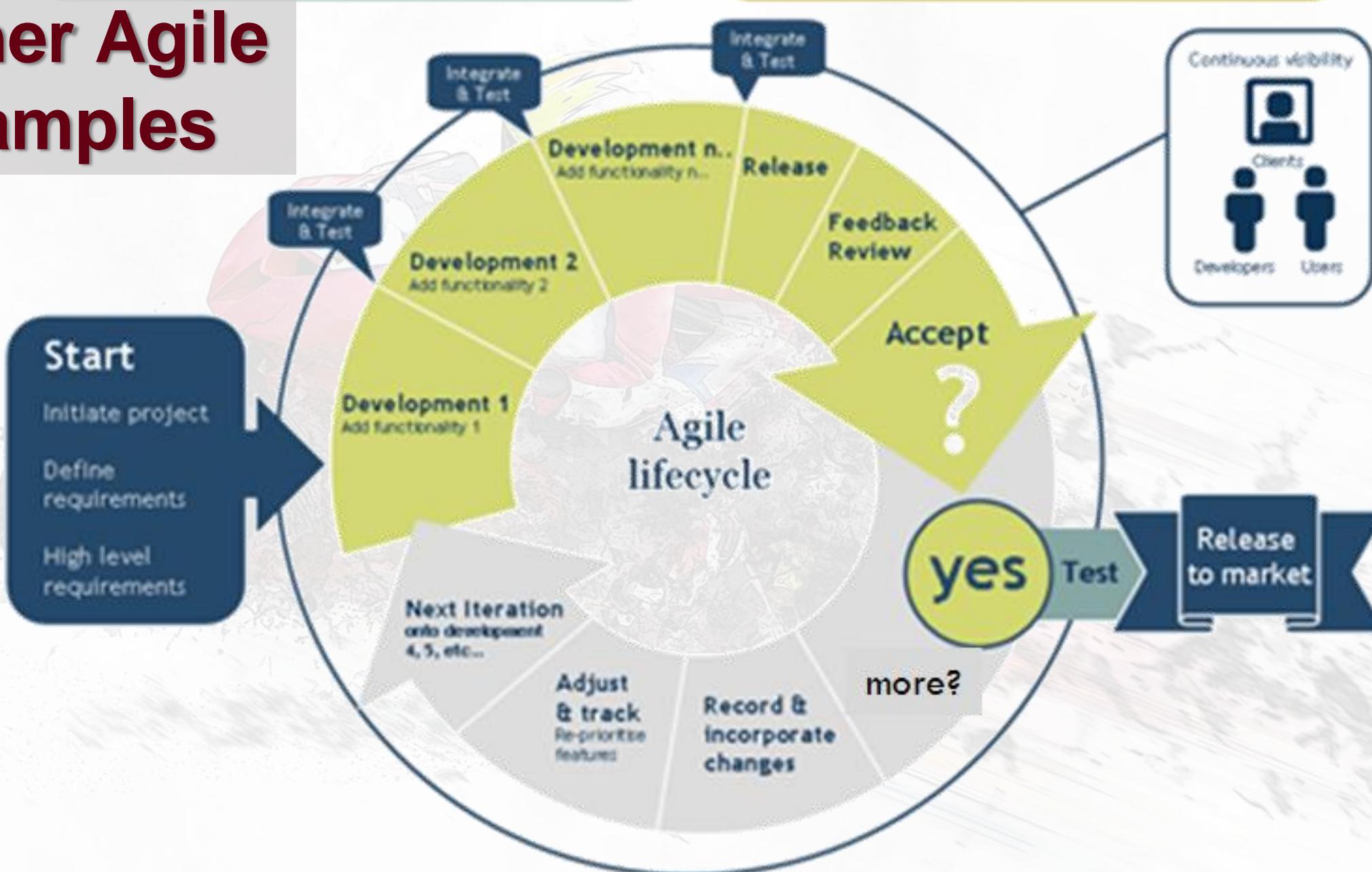
Agile Phases

- 
- A blurry, high-speed photograph of a motorcycle race. In the foreground, a red motorcycle with the number 16 is prominent, leaning into a turn. Other riders and motorcycles are visible in the background, creating a sense of motion.
1. **Envision:** determine the product vision, who is going to do the work, how team will work together
 2. **Speculate:** determine feature-based release, milestone, and iteration plan
 3. **Iteratively deliver features:** deliver tested feature in short time frames
 4. **Monitor & adapt:** review delivered results, current business environment, and team's performance, adapt as necessary
 5. **Close:** conclude the project, wrap up loose ends, and celebrate

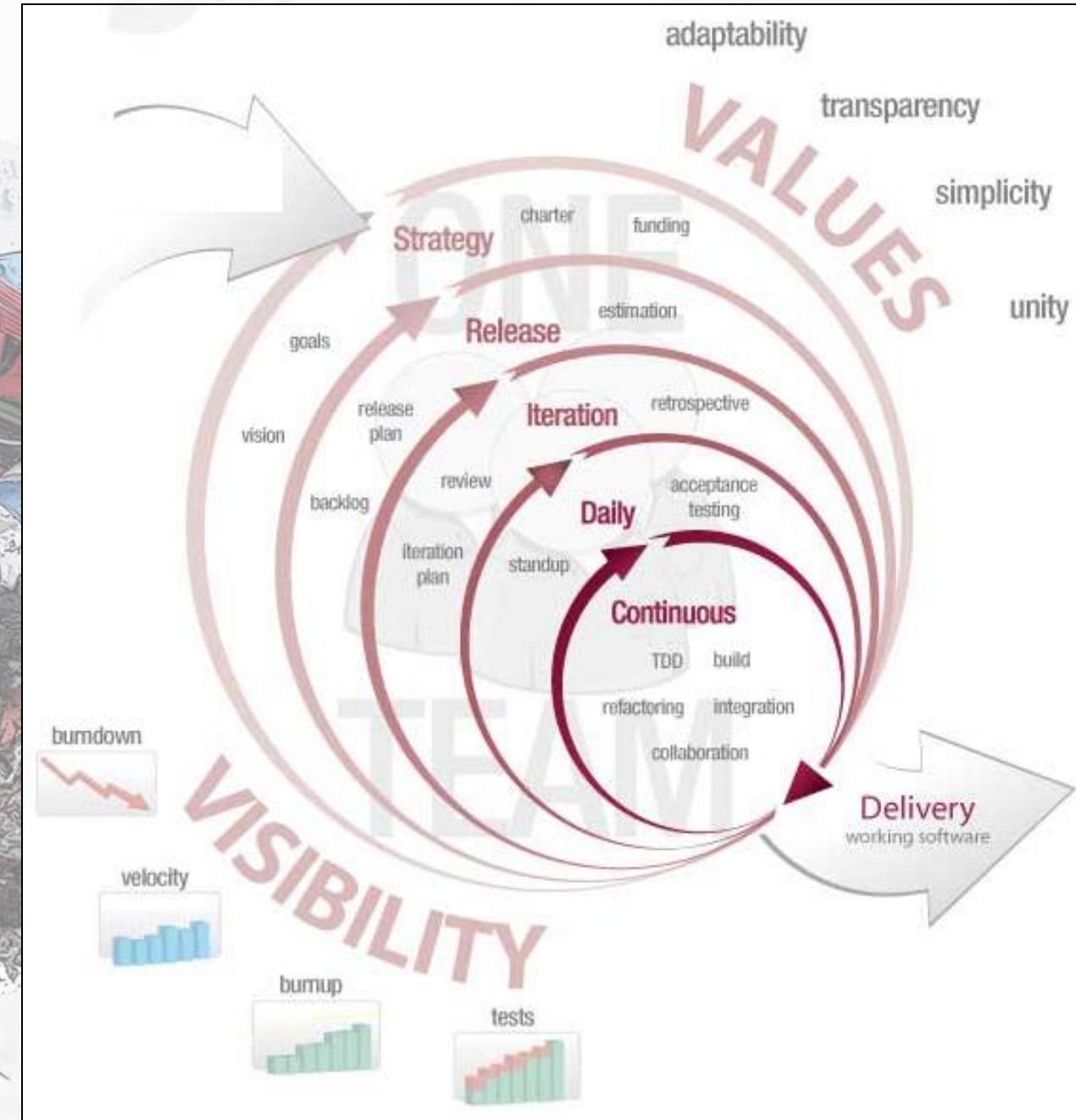
Agile Phases



Other Agile Examples



Other Agile Examples



Summary

Where would you like to work?

Project characteristics	Waterfall	RAD	Agile
Unclear & changing requirements	Poor	Good	Excellent
Unfamiliar technology	Poor	Good	Poor
Complex	Good	Good	Poor
Reliable	Good	Good	Good
Tight time constraints	Poor	Excellent	Excellent
Global teams	Excellent	Good	Poor
Fixed budget	Good	Poor	Good

On the other hand, agile typically has what we call poor process visibility. This means that upper management doesn't often get as much visibility into the project, the timeline, for example, how far along we are and when it will be done as we do with a more traditional process like waterfall. The other area where agile is still finding its place, so to speak, is in highly regulated environments. For example, FDA regulates the environment for medical devices, and agile processes don't often meet the documentation requirements of regulatory body like the FDA.



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA



The Role of the Business Analyst

Learning Objectives

After this lesson you will be able to...

- Describe the role of a business analyst (BA)
- Identify the phases of the SDLC in which the BA contributes
- Explain the nature and extent of the BA's contributions in each SDLC phase

The Role of BA is Key



The primary role of the business analyst is to manage and mediate communication.

- They document requirements because want to communicate those requirements effectively to developers.
- They document plans because they want to communicate those plans both to the people who have to implement it as well as to management.

Backgrounds of Business Analysts

- Management Information Systems degrees from business schools.
- Computer Science graduates who find that they enjoy working with the business a little bit more than simply writing code for example.
- Subject Matter Experts

The BA contributes in all phases of the systems development life cycle. We typically think of

- Skills & knowledge

- Characteristics

- Education & experience

You are writing a job posting for a BA.
What are your top requirements?

Role of the Business Analyst (BA):

- The BA contributes in all phases of the systems development life cycle.
- The BA's strongest contributions being in the analyze or analysis phase.
- The business analysts should be involved in the project from the very beginning.
- The business analysts are typically involved in change management or organizational readiness activities.

In summary, the business analyst bridges the gap of understanding between the business needs of the organization and the technology. The BA is in a sense a translator from the business world into the software developers or implementers who need to write or configure the software.

Where does the BA contribute?

Inefficient process

- Why?
- Id problem
- Calculate value
- Draft an approach
- Create the plan

Plan

Analyze

- What?
- Clarify requirements
- Understand current process
- Develop changes to business process

- How?
- Translate requirements into system specifications

Design

Implement

Technology enabled process

- Program and install automation
- Modify processes
- Ensure change is managed



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA



System Request



Learning Objectives



After this lesson, you will be able to...

- Describe the purpose of a **system request** in the context of the systems development lifecycle (SDLC)
- Determine when, and when not, to produce a system request
- List the important dimensions of a system request

Plan Phase Turns an Opportunity into a Project

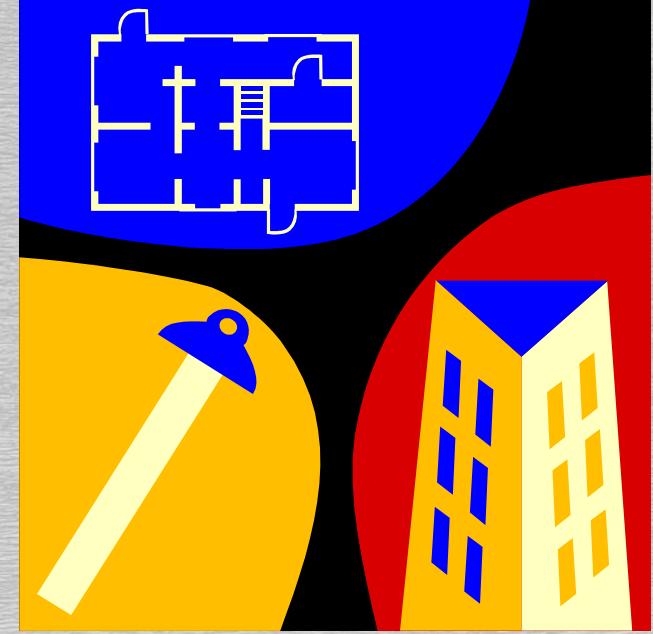
Triple Constraint

Scope

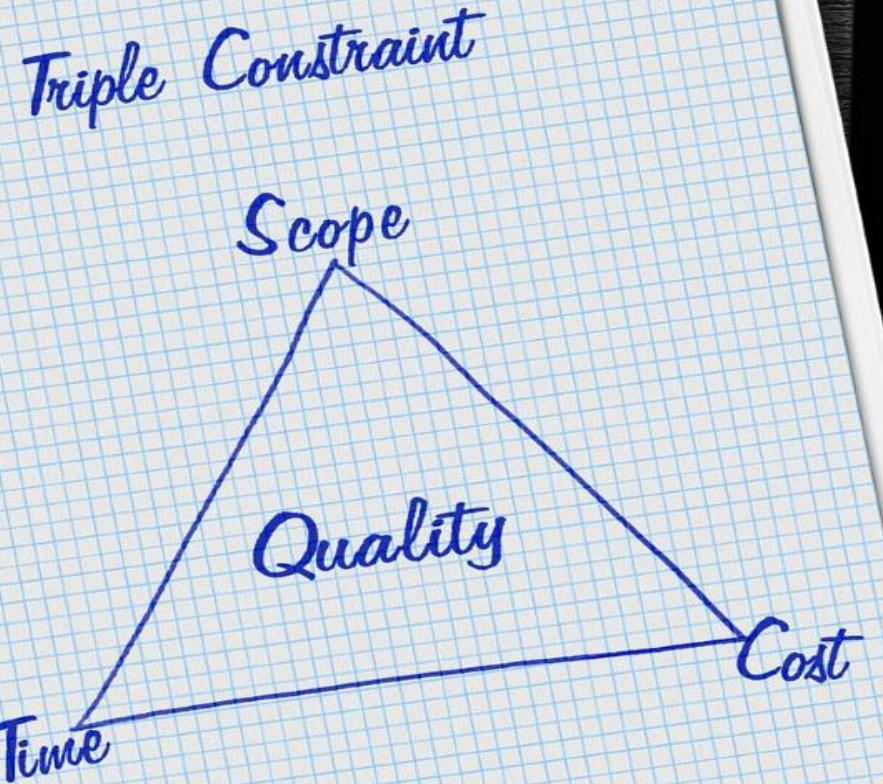
Quality

Time

1. Write system request
2. Analyze feasibility
3. Select project
4. Develop project plan



Use Deliverables Judiciously



- Build your toolbox; don't force fit each deliverable to every project
- Plan phase deliverables might include...
 - System Request
 - Feasibility Analysis
 - Project Plan
- Prepare those deliverables that add value and clarity to your task

Start with a System Request

A system request is also called a system proposal documents or other system initiation document.

- **System Request** is a general name for any document or system used to initiate the system development process.
 - Few organizations actually use the term “system request”
- Methods range from simple (spreadsheets) to complex (a formal submission and prioritization process)

Start with a System Request

- Think: How have the organizations you've worked for (or with) managed requests for new information systems and features?

Elements of the System Request

FIGURE 1-4
Elements of the System Request Form

Element	Description	Examples
Project Sponsor	The person who initiates the project and who serves as the primary point of contact for the project on the business side	Several members of the finance department Vice president of marketing IT manager Steering committee CIO CEO
Business Need	The business-related reason for initiating the system	Increase sales Improve market share Improve access to information Improve customer service Decrease product defects Supply chain acquisition processes
Business Requirements	The business capabilities that the system will provide	Provide online access to information Capture customer demographic information Include product search capabilities Produce management reports Include online user support
Business Value	The benefits that the system will create for the organization	3% increase in sales 1% increase in market share Reduction in headcount by 5 FTEs \$200,000 cost savings from decreased supply costs \$150,000 savings from removal of existing system
Special Issues or Constraints	Issues that are relevant to the implementation of the system that need to be known by the approval committee	Government-mandated deadline for Nov. 30 System needed in time for the Christmas holiday season Top-level security clearance needed by project team to work with data

* = Future equivalent

The System Request Initiates the SDLC

Sponsor
Business need
Business requirements
Value
Special Issues

Most importantly:
an approach

System Request—Digital Music Download Project

Project Sponsor: Carly Edwards, Assistant Vice President, Marketing

Business Need: This project has been initiated to create the capability of selling digital music downloads to customers through kiosks in our stores and over the Internet using our web site. Currently,

- Customers have many alternatives for downloading music and we need to provide this capability to retain our competitive position.
- Our music archive of rare and hard-to-find music is underutilized.

Business Requirements: Using this system over the Web or in-store kiosks, customers will be able to search for and purchase digital music downloads. The specific functionality that the system should have includes the following:

- Search for music in our digital music archive.
- Listen to music samples.
- Purchase individual downloads at a fixed fee per download.
- Establish a customer subscription account permitting unlimited downloads for a monthly fee.
- Purchase music download gift cards.

Business Value: We expect that Tune Source will increase sales by enabling existing customers to purchase specific digital music tracks and by reaching new customers who are interested in our unique archive of rare and hard-to-find music. We expect to gain a new revenue stream from customer subscriptions to our download services. We expect some increase in cross-selling, as customers who have downloaded a track or two of a CD decide to purchase the entire CD in a store or through our web site. We also expect a new revenue stream from the sale of music download gift cards.

Conservative estimates of tangible value to the company include the following:

- \$757,500 in sales from individual music downloads
- \$950,000 in sales from customer subscriptions
- \$205,000 in additional in-store or web site CD sales
- \$153,000 in sales from music download gift cards

Special Issues or Constraints:

- The marketing department views this as a strategic system. To prevent significant customer attrition, this project should be completed as soon as possible.

Plan Phase Turns an Opportunity into a Project

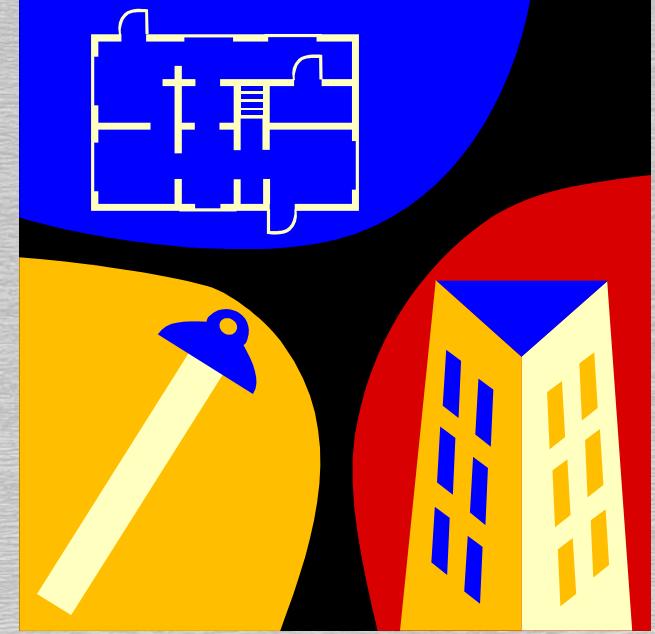
Triple Constraint

Scope

Quality

Time

- ✓ Write system request
- 2. Analyze feasibility
- 3. Select project
- 4. Develop project plan





CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA



Feasibility Analysis



Learning Objectives

A close-up photograph of a white and blue soccer ball that has just been kicked into a black mesh goal. The ball is positioned in the upper left quadrant of the frame, with the goal's net filling the lower right. The lighting creates strong highlights and shadows on the ball's surface.

After this lesson, you will be able to:

- Describe the 3 dimensions of a feasibility analysis
- Identify the primary considerations for technical feasibility
- Describe the 3 types of project financial benefits and which is most important
- Explain Total Cost of Ownership and its importance in process improvement projects

Before We Invest, We Evaluate!

- A **feasibility analysis** examines the feasibility of a particular project
- We break the concept of feasibility down into 3 dimensions:
 - Can we build it? (Technical Feasibility)
 - Should we build it? (Economic Feasibility)
 - Will people use it? (Organizational Feasibility)

Before We Invest, We Evaluate!

- The process of doing the feasibility analysis is more important than the final format
 - “The journey is its own reward!”

Can we build it?

- **Technical feasibility** addresses the organization's technical readiness to take on the project
- Consider:
 - Familiarity with the business application
 - Familiarity with the proposed technology
 - Project size
 - Integration requirements

Can we build it?

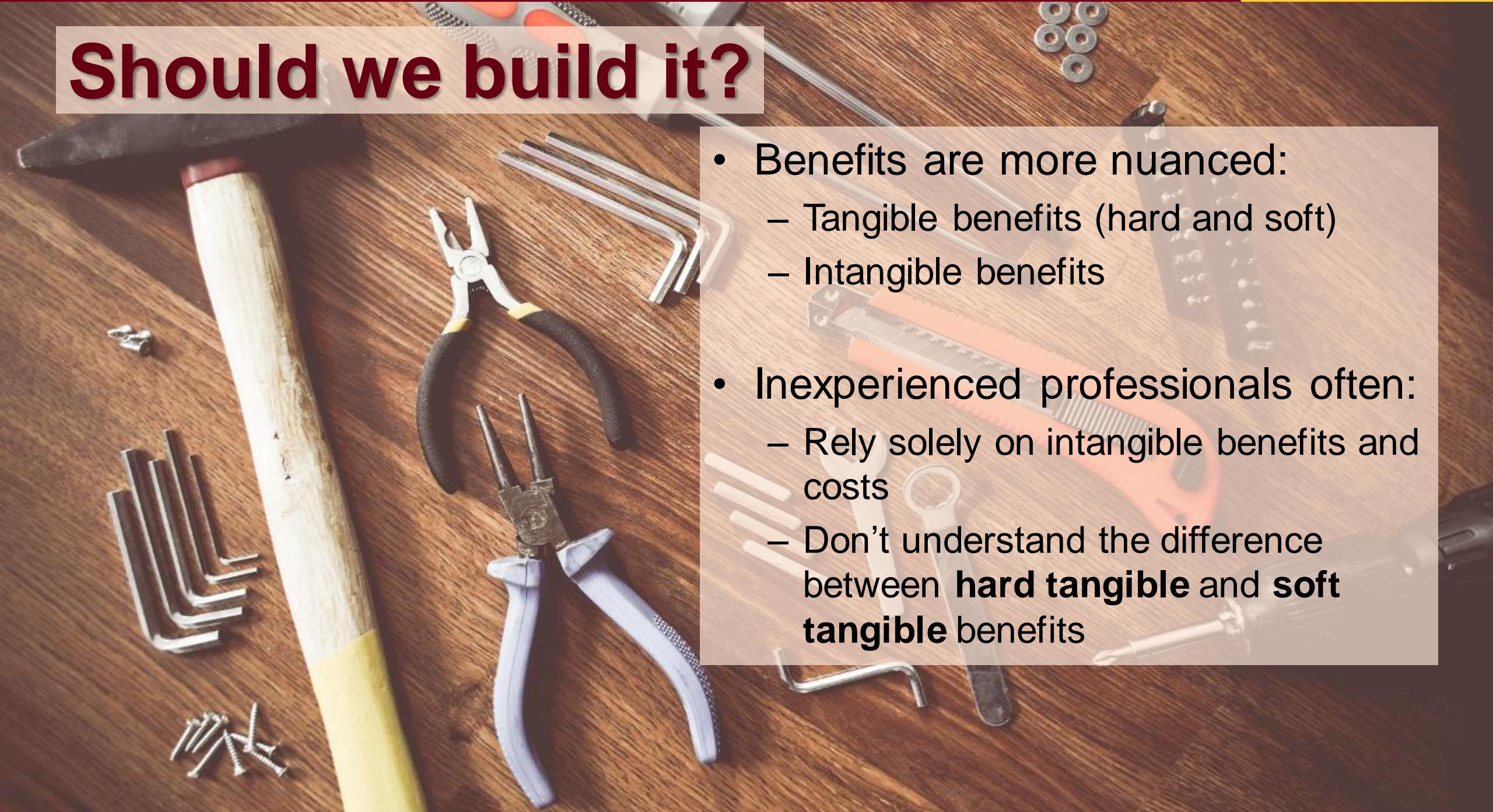
- How do these factors affect feasibility? Inexperienced professionals often:
 - State the technical aspects too clearly
 - Follow the recipe exactly

Should we build it?

- **Economic feasibility** examines whether it makes financial sense to proceed with a proposed improvement project
- You might think this is “not my job”
- Costs are relatively easy to understand

Should we build it?

- Benefits are more nuanced:
 - Tangible benefits (hard and soft)
 - Intangible benefits
- Inexperienced professionals often:
 - Rely solely on intangible benefits and costs
 - Don't understand the difference between **hard tangible** and **soft tangible** benefits



Tangible Benefits = a Number!

- **Tangible** economic benefits are generally:
 - A reduction in cost
 - An increase in revenue
 - Which is preferable?
- You can put a \$ number on tangible benefits, and they generally impact the overall cash flow of a project



Tangible Benefits = a Number!

- Examples include additional sales due to process improvement, or reduced staffing
- **Hard tangible** benefits impact the project cash flow or company P&L
 - Example: additional sales
- **Soft tangible** benefits are measurable, but do not impact the cash flow or P&L
 - Example: freeing up warehouse space



Intangible Benefits are “Nice to have”

- **Intangible** benefits do not have a precise dollar figure, but they are still important
- Examples:
 - Brand recognition
 - Corporate image
 - Job satisfaction
- How do you account for intangible benefits in the project cash flow or company P&L?



Should we build it?

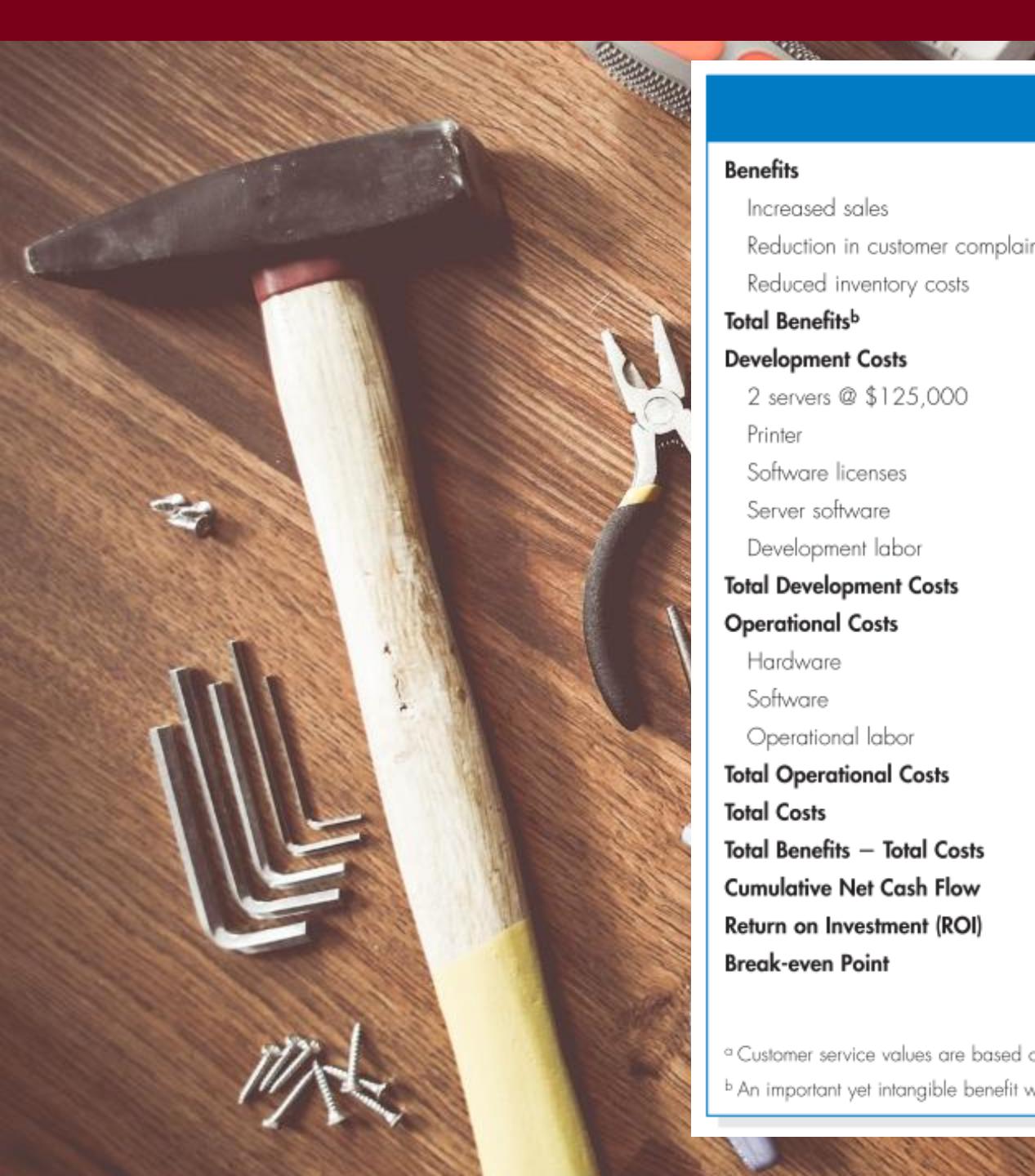
- We use a discounted cash flow model to determine the net present value (NPV) of a proposed process improvement:

FIGURE 1-8
Simple Cash Flow Projection

	Year 0	Year 1	Year 2	Year 3	Total
Total Benefits		45,000	50,000	57,000	152,000
Total Costs	100,000	10,000	12,000	16,000	138,000
Net Benefits (Total Benefits – Total Costs)	(100,000)	35,000	38,000	41,000	14,000
Cumulative Net Cash Flow	(100,000)	(65,000)	(27,000)	14,000	

FIGURE 1-9
Discounted Cash Flow Projection

	Year 0	Year 1	Year 2	Year 3	Total
Total Benefits		45,000	50,000	55,000	
PV of Total Benefits		40,909	41,322	42,825	125,056
Total Costs	100,000	10,000	12,000	16,000	
PV of Total Costs	100,000	9,091	9,917	12,021	131,029



	2012	2013	2014	2015	2016	Total
Benefits						
Increased sales		500,000	530,000	561,800	595,508	2,187,308
Reduction in customer complaint calls ^a		70,000	70,000	70,000	70,000	280,000
Reduced inventory costs		68,000	68,000	68,000	68,000	272,000
Total Benefits^b	638,000	668,000	699,800	733,508	2,739,308	
Development Costs						
2 servers @ \$125,000	250,000	0	0	0	0	250,000
Printer	100,000	0	0	0	0	100,000
Software licenses	34,825	0	0	0	0	34,825
Server software	10,945	0	0	0	0	10,945
Development labor	1,236,525	0	0	0	0	1,236,525
Total Development Costs	1,632,295	0	0	0	0	1,632,295
Operational Costs						
Hardware		50,000	50,000	50,000	50,000	200,000
Software		20,000	20,000	20,000	20,000	80,000
Operational labor		115,000	119,600	124,384	129,359	488,343
Total Operational Costs		185,000	189,600	194,384	199,359	768,343
Total Costs	1,632,295	185,000	189,600	194,384	199,359	2,400,638
Total Benefits – Total Costs	(1,632,295)	453,000	478,400	505,416	534,149	338,670
Cumulative Net Cash Flow	(1,632,295)	(1,179,295)	(700,895)	(195,479)	338,670	
Return on Investment (ROI)	14.1%	(338,670/2,400,638)				
Break-even Point	3.37 years	[3 years of negative cumulative cash flow + [534,149 – 338,670]/534,149 = .37]				

TCO is the Bottom Line!

- **Total Cost of Ownership (TCO)** refers to the entire cost of building, maintaining, and eventually retiring a software system
- According to Gartner, initial project cost is only 8% of TCO!
- For every \$1 of initial cost you spend \$.77 per year in maintenance and support

TCO is the Bottom Line!

- Thus, \$12.50 in TCO for each \$1 of project cost
- Consider the implications: \$10M system with \$5M in hard benefits annually.
 - Without TCO: 70% ROI
 - With TCO: 1% ROI



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Organizational Readiness



Learning Objectives

A close-up photograph of a white and blue soccer ball that has just been kicked into a black and yellow soccer net. The ball is positioned on the left side of the frame, with the net filling the background.

After this lesson, you will be able to:

- Describe the concept of organizational feasibility
- Define organizational readiness and its relationship to organizational feasibility
- List strategies for enhancing organizational feasibility

Before We Invest, We Evaluate!

- A **feasibility analysis** examines the feasibility of a particular project
- We break the concept of feasibility down into 3 dimensions:
 - Can we build it? (Technical Feasibility)
 - Should we build it? (Economic Feasibility)
 - **Will people use it? (Organizational Feasibility)**

Before We Invest, We Evaluate!

- The process of doing the feasibility analysis is more important than the final format
 - “The journey is its own reward!”

Will people use it?

- **Organizational feasibility** examines whether the organization is ready for the improved system or process
- Most of the time, it's not *whether* users will resist the new system but more about identifying *who* will resist the new system!

Will people use it?

- Examine support from:
 - Top-down
 - Bottom-up
- Inexperienced professionals might:
 - Assume everyone will be happy with the new change

If We Build It, Will People Use It?

Internet Order Feasibility Analysis Executive Summary

Margaret Mooney and Alec Adams created the following feasibility analysis for the CD Selections Internet Order System Project. The System Proposal is attached, along with the detailed feasibility study. The highlights of the feasibility analysis are:

Technical Feasibility

The Internet Order System is feasible technically, although there is some risk.

Organizational Feasibility

From an organizational perspective, this project has low risk. The objective of the system, which is to increase sales, is aligned well with the senior management's goal of increasing sales for the company. The move to the Internet also aligns with Marketing's goal to become more savvy in Internet marketing and sales.

The project has a project champion, Margaret Mooney, Vice President of Marketing. Margaret is well positioned to sponsor this project and to educate the rest of the senior management team when necessary. To date, much of senior management is aware of and supports the initiative.

The users of the system, Internet consumers, are expected to appreciate the benefits of CD Selections' Web presence. And, management in the retail stores should be willing to accept the system, given the possibility of increased sales at the store level.

Additional Comments:

- The Marketing Department views this as a strategic system. This Internet system will add value to our current business model, and it also will serve as a proof of concept for future Internet endeavors. For example, in the future, CD Selections may want to sell products directly over the Internet.
- We should consider hiring a consultant with expertise in similar applications to assist with the project.
- We will need to hire new staff to operate the new system, from both the technical and business operations aspects.

Leverage Your Stakeholders

	Role	To Enhance Organizational Feasibility
Champion	A champion: <ul style="list-style-type: none">• Initiates the project• Promotes the project• Allocates his or her time to the project• Provides resources	<ul style="list-style-type: none">• Make a presentation about the objectives of the project and the proposed benefits to those executives who will benefit directly from the system.• Create a prototype of the system to demonstrate its potential value.• Make a presentation to management about the objectives of the project and the proposed benefits.• Market the benefits of the system, using memos and organizational newsletters.• Encourage the champion to talk about the project with his or her peers.• Assign users official roles on the project team.• Assign users specific tasks to perform, with clear deadlines.• Ask for feedback from users regularly (e.g., at weekly meetings).
Organizational Management	Organizational managers: <ul style="list-style-type: none">• Know about the project• Budget enough money for the project• Encourage users to accept and use the system	
System Users	Users: <ul style="list-style-type: none">• Make decisions that influence the project• Perform hands-on activities for the project• Ultimately determine whether the project is successful by using or not using the system	

FIGURE 1-14

Important Stakeholders for Organizational Feasibility

Strategy for Success: Organizational Readiness

Gartner.
G00173029

**Beyond Change Management: A Guide to
Organizational Readiness for BPM Projects**

Published: 11 December 2009

Are change management and
organizational readiness synonyms?

Textbook Approaches are Narrow!

Inefficient process

- Why?
- Id problem
- Calculate value
- Draft an approach
- Create the plan

Plan

Analyze

- What?
- Clarify requirements
- Understand current process
- Develop changes to business process

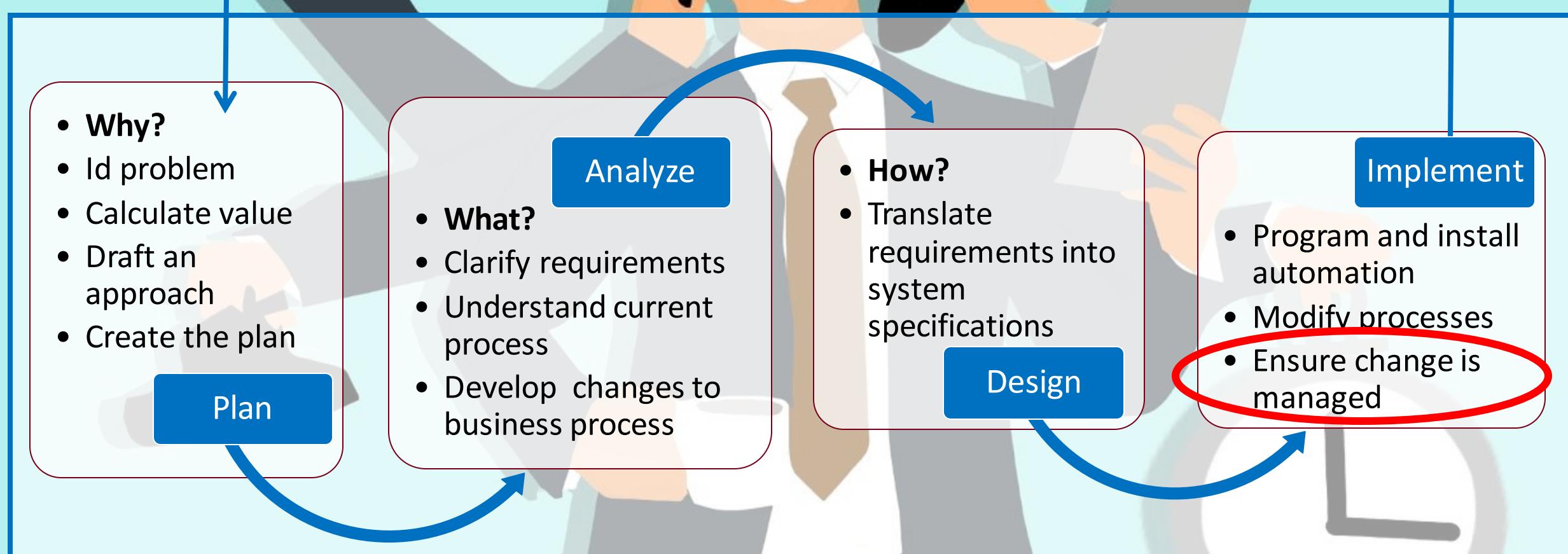
- How?
- Translate requirements into system specifications

Design

Implement

- Program and install automation
- Modify processes
- Ensure change is managed

Technology enabled process



Change Management Historically Occurs in Implementation Phase

Time for Change

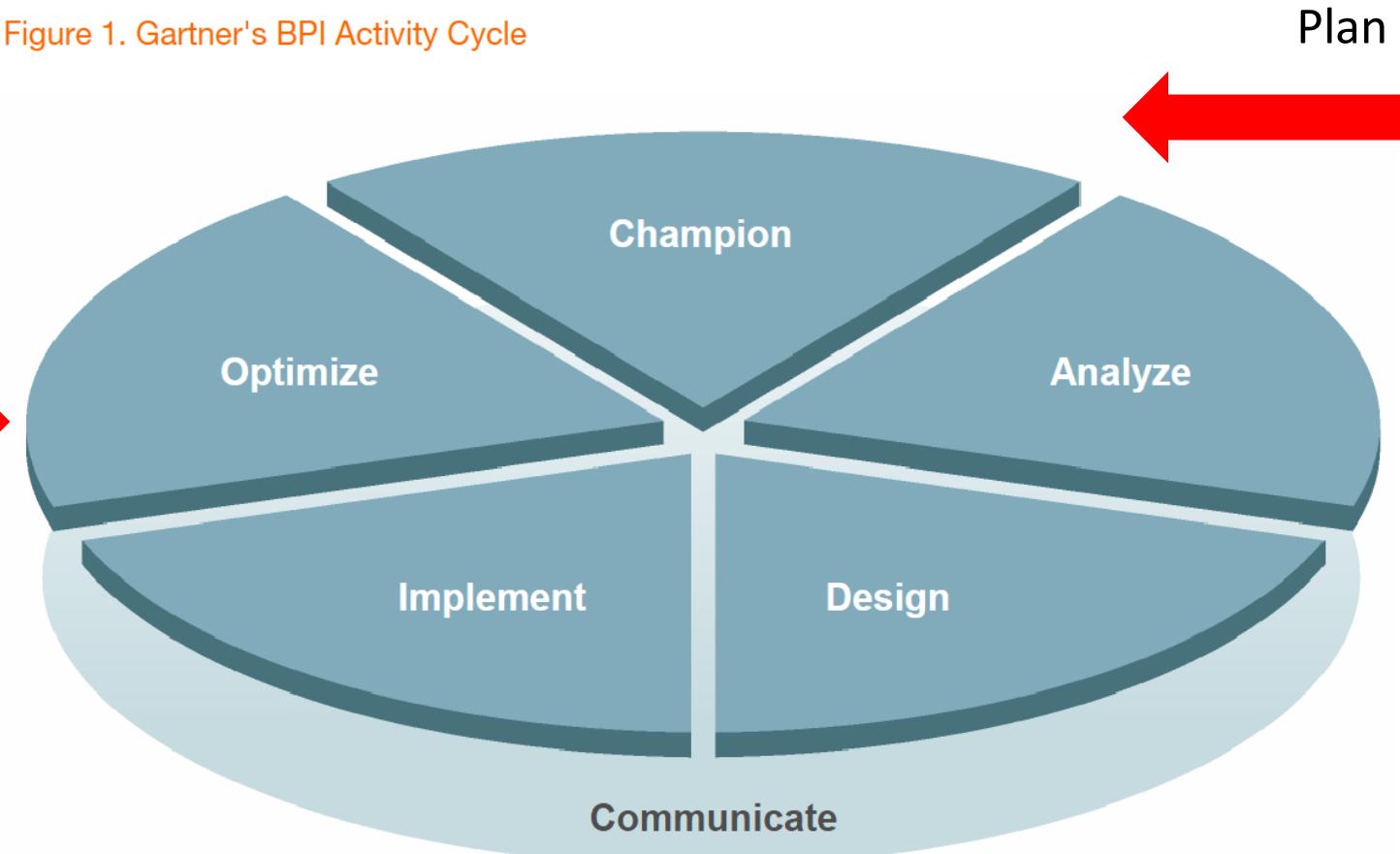
- Preparing people for the new system
- Revising management policies
- Assessing costs and benefits
- Motivating adoption
- Enabling adoption: training

These subtopics in most textbooks address change management in Implementation phase

Organizational Readiness Occurs in All SDLC

Post-project

Figure 1. Gartner's BPI Activity Cycle

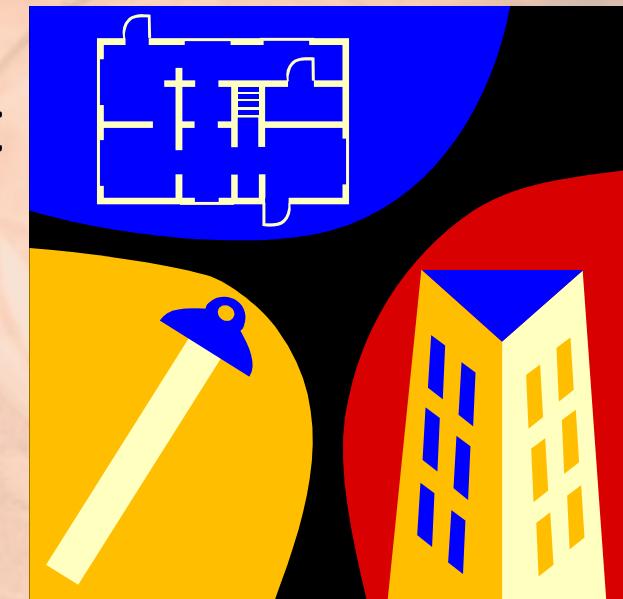


Source: Gartner (December 2009)

The remainder of the article
describes organizational readiness
activities that occur in each phase

Plan Phase Turns an Idea into a Project

- ✓ Write system request
- ✓ Analyze feasibility
- 3. Select project
- 4. Develop project plan





CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA



System Proposal

Learning Objectives



After this lesson, you will be able to:

- Identify artifacts from plan phase that we revised in analysis phase
- Describe new artifacts created during analysis phase
- List at least one diagram type that we used to understand the current process

Starting Analysis Phase

Inefficient process

System Proposal

Technology enabled process

- Why?
- Id problem
- Calculate value
- Draft an approach
- Create the plan

Plan

- What?
- Clarify requirements
- Understand current process
- Develop changes to business process

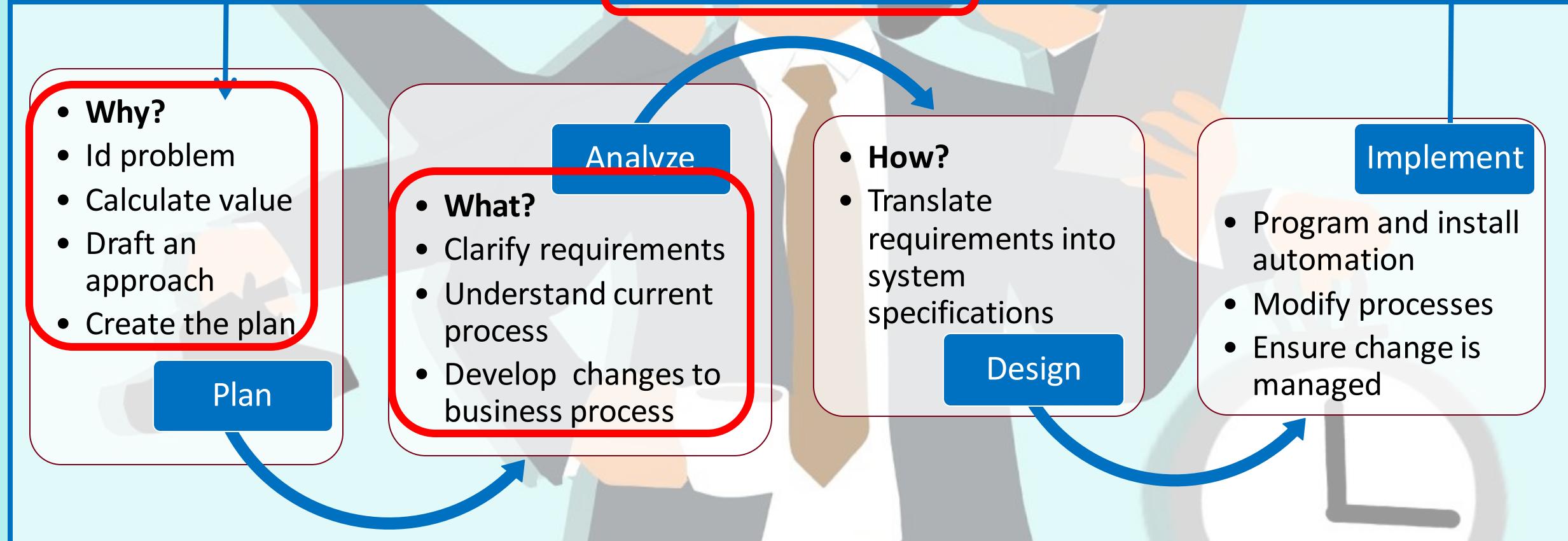
Analyze

- How?
- Translate requirements into system specifications

Design

Implement

- Program and install automation
- Modify processes
- Ensure change is managed



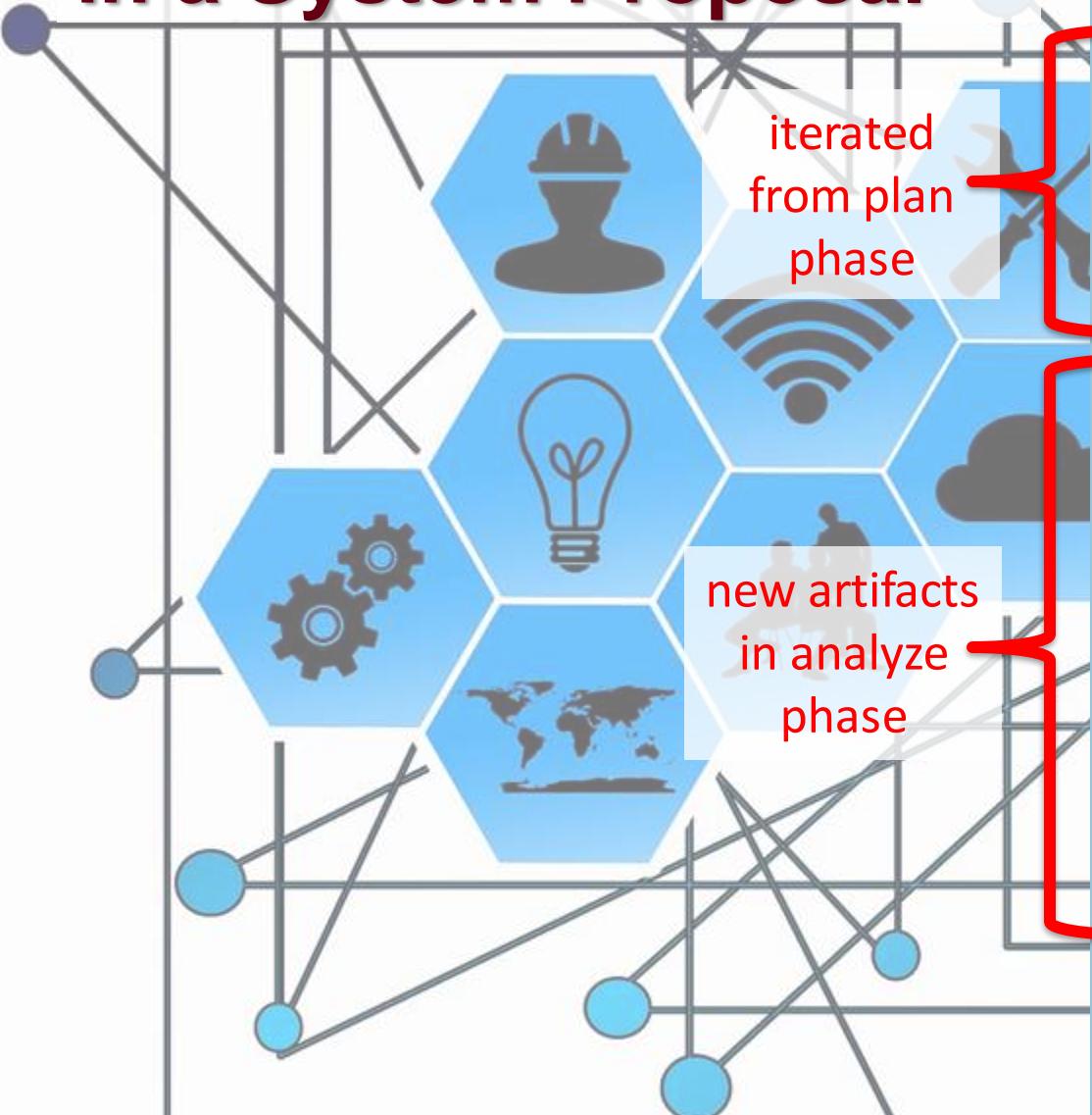
Begin With the End in Mind



The analysis phase answers the questions of who will use the system, what the system will do, and where and when it will be used.

All of the deliverables are combined into a system proposal, which is presented to management, who decides whether the project should continue to move forward.

Analysis Phase Results in a System Proposal



1. Table of Contents

2. Executive Summary

A summary of all the essential information in the proposal so that a busy executive can read it quickly and decide what parts of the plan to read in more depth.

3. System Request

The revised system request form. (See Chapter 1.)

4. Work plan

The original work plan, revised after having completed the analysis phase. (See Chapter 2.)

5. Feasibility Analysis

A revised feasibility analysis, using the information from the analysis phase. (See Chapter 1.)

6. Requirements Definition

A list of the functional and nonfunctional business requirements for the system (this chapter).

7. Use Cases

A set of use cases that illustrate the basic processes that the system needs to support. (See Chapter 4.)

8. Process Model

A set of process models and descriptions for the to-be system. (See Chapter 5.) This may include process models of the current as-is system that will be replaced.

9. Data Model

A set of data models and descriptions for the to-be system. (See Chapter 6.) This may include data models of the as-is system that will be replaced.

Appendices

These contain additional material relevant to the proposal, often used to support the recommended system. This might include results of a questionnaire survey or interviews, industry reports and statistics, etc.

What is missing?

Developing Requirements is a Multi-Step Process

Understand as-is system

Define requirements

Resolve priorities & inconsistencies

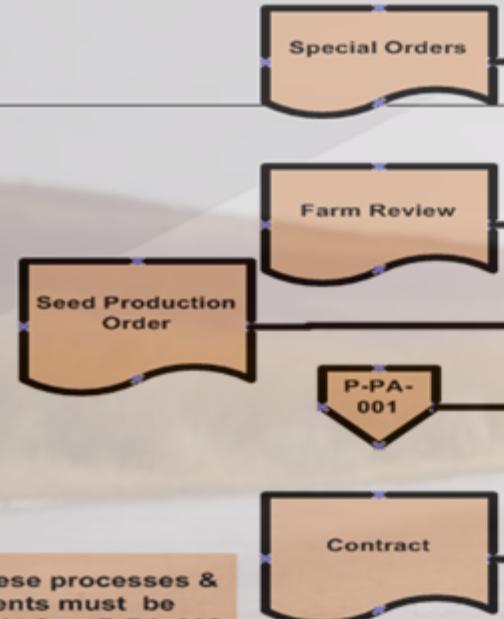
Identify improvements

Develop test cases

Crops Planning (Process P-PA-002)

LOSGR^{OB}O
AGROPECUARIA

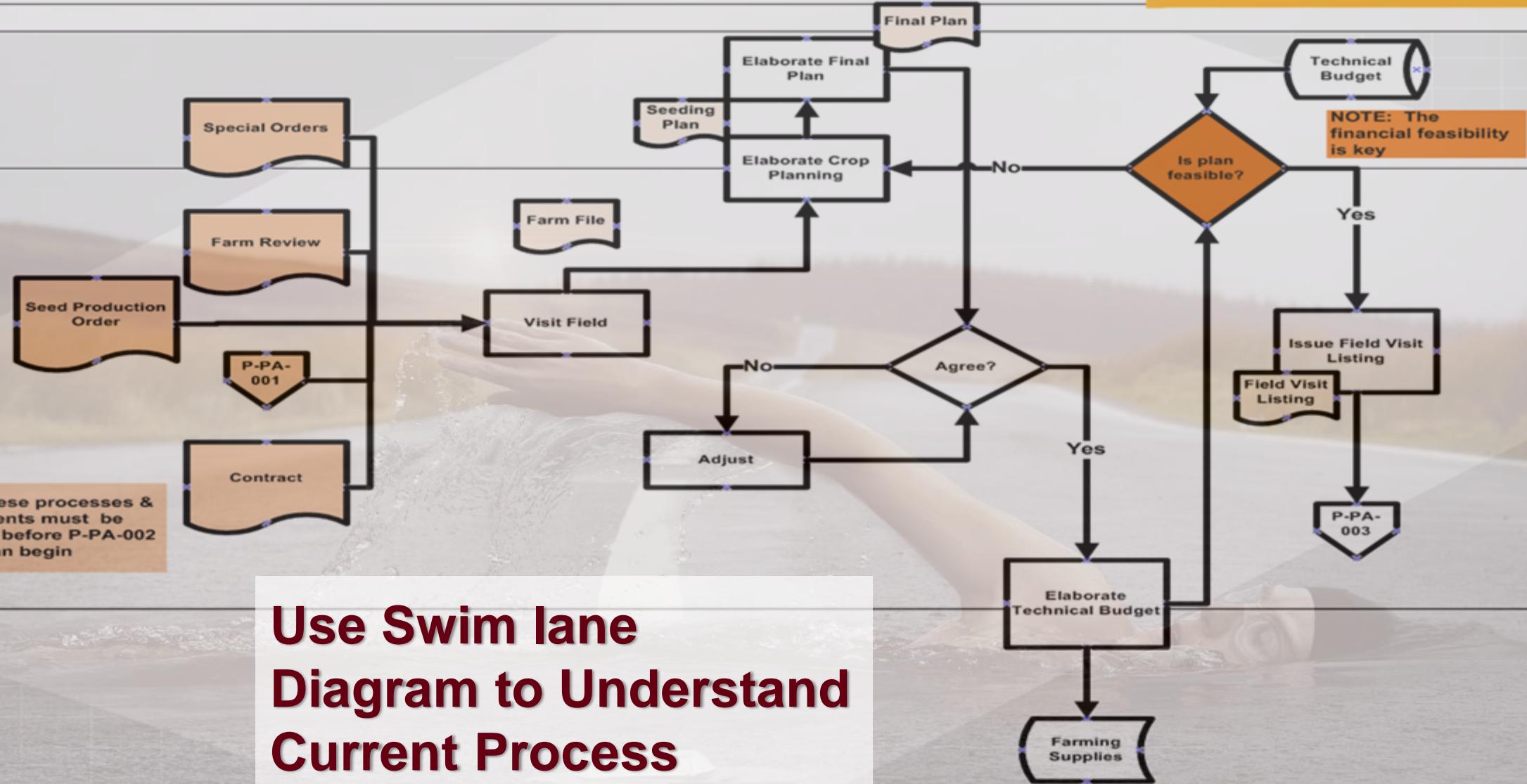
Production Manager



NOTE: These processes & documents must be completed before P-PA-002 can begin

Agronomist

Agronomist



Use Swim Lane Diagram to Understand Current Process



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA



Well-Formed Requirements



Learning Objectives

After this lesson, you will be able to:

- Describe the purpose of writing requirements
- List the qualities of a well-formed requirement

Developing Requirements is a Multi-Step Process

Understand as-is system

Define requirements

Resolve priorities & inconsistencies

Identify improvements

Develop test cases

Requirements State What the System Must Do

- The purpose of writing requirements is **communication**
 - To gain approval from users/stakeholders
 - To tell developers what to build
 - To tell testers what to test
 - To help marketers market
 - To help technical writers write
 - ...and on and on.
- Requirements should not state “how” the system will meet the requirement

Well-formed Requirements...

(from ISO/IEC/IEEE 29148:2011)

...can be **verified**.

...has to be met or possessed by a
system to solve a stakeholder problem
or to achieve a stakeholder objective.

Well-formed Requirements...

...is qualified by **measurable** conditions and bounded by constraints.

...defines the **performance** of the system when used by a specific stakeholder or the corresponding **capability** of the system, but not a capability of the user, operator, or other stakeholder.

Each Individual Requirement Should...

(from ISO/IEC/IEEE 29148:2011)

- ...be uniquely **identified**.
- ...include **priority**. (1 – 5, H/M/L, etc)
- ...identify **dependencies**.

Each Individual Requirement Should...

- ...be **traceable** to the source.
- ...include **rationale**.
- ...describe the **difficulty** of implementation.
- ...be **categorized**. (functional / non-functional, etc)



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA



Individual Requirements



Learning Objectives

After this lesson, you will be able to:

- List the desirable qualities of good requirements
- Define each quality

Developing Requirements is a Multi-Step Process

Understand as-is system

Define requirements

Resolve priorities & inconsistencies

Identify improvements

Develop test cases

Individual Requirements

Not Optional

(from ISO/IEC/IEEE 29148:2011)

- **Necessary.** The requirement defines an essential capability, characteristic, constraint, and/or quality factor. If it is removed or deleted, a deficiency will exist, which cannot be fulfilled by other capabilities of the product or process. The requirement is currently applicable and has not been made obsolete by the passage of time. Requirements with planned expiration dates or applicability dates are clearly identified.

Individual Requirements

(from ISO/IEC/IEEE 29148:2011)

- **Implementation Free.** The requirement, while addressing what is necessary and sufficient in the system, avoids placing unnecessary constraints on the architectural design. The objective is to be implementation independent. The requirement states what is required, not how the requirement should be met.

Individual Requirements

(from ISO/IEC/IEEE 29148:2011)

- **Unambiguous.** The requirement is stated in such a way so that it can be interpreted in only one way. The requirement is stated simply and is easy to understand.

Individual Requirements

(from ISO/IEC/IEEE 29148:2011)

- **Consistent.** The requirement is free of conflicts with other requirements.

Individual Requirements

(from ISO/IEC/IEEE 29148:2011)

- **Complete.** The stated requirement needs no further amplification because it is measurable and sufficiently describes the capability and characteristics to meet the stakeholder's need.

Individual Requirements

(from ISO/IEC/IEEE 29148:2011)

- **Singular.** The requirement statement includes only one requirement with no use of conjunctions.

Individual Requirements

(from ISO/IEC/IEEE 29148:2011)

- **Feasible.** The requirement is technically achievable, does not require major technology advances, and fits within system constraints (e.g., cost, schedule, technical, legal, regulatory) with acceptable risk.

Individual Requirements

(from ISO/IEC/IEEE 29148:2011)

- **Traceable.** The requirement is upwards traceable to specific documented stakeholder statement(s) of need, higher tier requirement, or other source (e.g., a trade or design study). The requirement is also downwards traceable to the specific requirements in the lower tier requirements specification or other system definition artefacts. That is, all parent-child relationships for the requirement are identified in tracing such that the requirement traces to its source and implementation.

Individual Requirements

(from ISO/IEC/IEEE 29148:2011)

- **Verifiable.** The requirement has the means to prove that the system satisfies the specified requirement. Evidence may be collected that proves that the system can satisfy the specified requirement. Verifiability is enhanced when the requirement is measurable.

Requirements in Real Life

From: 30 key requirements for an ERP replacement

	Primary Process	Requirement	Source	Definition
19	After Sales Support	Recall Management	Interview 9/5/04	Ability to track and report on all aspects of product genealogy and distribution history, in order to quickly and efficiently manage product recall.
20	Demand-to-Supply Proposal-to-Revenue	Supply Chain Visibility	S2STh	Ability to manage worldwide inventory (at supplier; in field sales; on plant floor; on customer consignment, and in distribution network) through: <ul style="list-style-type: none">• visibility to cross-site inventory; supplier inventory; and supplier capabilities• ability to understand the impact of product expiration• ability to quickly respond to shortages and unplanned demand with a view of time-phased supply-and-demand at each supply chain node• tools that may allow tracking of customer-owned inventory, either through integrations to hospital systems or calculations of 'shipped product – implanted product.'

How'd they do? one good thing/one potential improvement



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA



Requirements Language



Learning Objectives



After this lesson, you will be able to:

- Describe the basic language elements used to construct a good requirement
- List common pitfalls in writing requirements and how to avoid them
- Use 3 sample syntaxes to write good requirements

Developing Requirements is a Multi-Step Process

Understand as-is system

Define requirements

Resolve priorities & inconsistencies

Identify improvements

Develop test cases

Use Consistent Language

(from ISO/IEC/IEEE 29148:2011)

- Requirements are mandatory binding provisions and use **shall**.
- Preferences or goals are desired, non-mandatory, non-binding provisions and use **should**.
- Suggestions or allowances are non-mandatory, non-binding provisions and use **may**.

Use Consistent Language

(from ISO/IEC/IEEE 29148:2011)

- Non-requirements, such as descriptive text, use verbs such as 'are', 'is', and 'was'. It is best to avoid using the term 'must', due to potential misinterpretation as a requirement.
- Use **positive statements** and avoid negative requirements such as 'shall not'.
- Use **active voice**: avoid using passive voice, such as 'shall be able to select'.

Avoid...

SVEIKI
SERVIUS
Cześć
HELO
HALOO
TIENS
ALOHA
HEI
ZDRAV
SALUT

(from ISO/IEC/IEEE 29148:2011)

- **Superlatives** (such as 'best', 'most')
- **Subjective language** (such as 'user friendly', 'easy to use', 'cost effective')
- **Vague pronouns** (such as 'it', 'this', 'that')
- **Ambiguous** adverbs and adjectives (such as 'almost always', 'significant', 'minimal')

Avoid...



(from ISO/IEC/IEEE 29148:2011)

- **Open-ended**, non-verifiable terms (such as 'provide support', 'but not limited to', 'as a minimum')
- **Comparative phrases** (such as 'better than', 'higher quality')
- **Loopholes** (such as 'if possible', 'as appropriate', 'as applicable')

Avoid...



(from ISO/IEC/IEEE 29148:2011)

- **Incomplete references** (not specifying the reference with its date and version number; not specifying just the applicable parts of the reference to restrict verification work)
- **Negative statements** (such as statements of system capability not to be provided)

Use a Formal Syntax (Example)

Dzień dobry

SZIA

(from ISO/IEC/IEEE 29148:2011)

SVEIKI

SERVUS

Cześć

BUNNIZUA

SELAM

HEICAN

HEJ

GUTEN TAG

JOUR

SZERVUSZ

BONGHJORNU

DAR FIADA

OLÁ

Allô

TIENS

ALOHA

HEI

ZDRAV

SALUT

VER

[Condition] [Subject] [Action] [Object]
[Constraint]

EXAMPLE: When signal x is received
[Condition], the system [Subject] shall
set [Action] the signal x received bit
[Object] within 2 seconds [Constraint].

Use a Formal Syntax (Example 2)

Dzień dobry

SZIA

(from ISO/IEC/IEEE 29148:2011)

SVEIKI

SERVUS

Cześć

SELAM

HEIAN

[Condition] [Action or Constraint]
[Value]

HELLO

HALOO

Hi

EXAMPLE: At sea state 1 [Condition],
the Radar System shall detect targets at
ranges out to [Action or Constraint]
100 nautical miles [Value].

TIENS

ALOHA

HEI

ZDRAV

SALUT

ZDR

VER

Use a Formal Syntax (Example 3)

Dzień dobry

SZIA

(from ISO/IEC/IEEE 29148:2011)

[Subject] [Action] [Value]

EXAMPLE: The Invoice System
[Subject], shall display pending
customer invoices [Action] in ascending
order [Value] in which invoices are to be
paid.



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Functional vs. Non-functional Requirements



Learning Objectives

A close-up photograph of a white and blue soccer ball that has been kicked into a dark-colored net. The ball is positioned on the left side of the frame, with its black pentagonal panels and white缝合 lines clearly visible. The net's mesh is draped behind it, creating a textured background.

After this lesson, you will be able to:

- Describe the purpose of separating requirements into functional vs. nonfunctional categories
- List the characteristics of both functional and nonfunctional requirements
- Categorize requirements as functional or nonfunctional

Developing Requirements is a Multi-Step Process

Understand as-is system

Define requirements

Resolve priorities & inconsistencies

Identify improvements

Develop test cases

Separate Your Requirements

- Separate your requirements into **functional** and **non-functional** categories
- We use each type of requirement in a different way as we proceed with our system
 - Functional requirements relate to the process the system has to perform
 - Analysis Phase
 - Non-functional requirements relate to constraints on the system
 - Design Phase

Functional Requirements



Functional Requirements

1. New Vehicle Management

- 1.1 The system will allow managers to view the current new vehicle inventory.
- 1.2 The system will allow the new vehicle manager to place orders for new vehicles.
- 1.3 The system will record the addition of new vehicles to inventory when they are received from the manufacturers.

2. Vehicle Sales Management

- 2.1 The system will enable salespersons to create a customer offer.
- 2.2 The system will allow salespeople to know whether an offer is pending on a specific vehicle.
- 2.3 The system will enable managers to record approval of a customer offer.
- 2.4 The system will prepare a sales contract.
- 2.5 The system will prepare a shop work order based on customer requested dealer options.
- 2.6 The system will record a customer deposit.
- 2.7 The system will record a customer payment.
- 2.8 The system will create a record of the customer's vehicle purchase.

3. Used Vehicle Management

- 3.1 The system will record information on a customer trade-in vehicle ... etc.

Nonfunctional Requirements

Nonfunctional Requirements

1. Operational

- 1.1 The system should run on tablet PCs to be used by salespeople.
- 1.2 The system should interface with the shop management system.
- 1.3 The system should connect to printers wirelessly.

2. Performance

- 2.1 The system should support a sales staff of 15 salespeople.
- 2.2 The system should be updated with pending offers on vehicles every 15 minutes.

3. Security

- 3.1 No salesperson can access any other salesperson's customer contacts.
- 3.2 Only the owner and sales manager may approve customer offers.
- 3.3 Use of each tablet PC should be restricted to the salesperson to whom it is assigned.

4. Cultural and Political

- 4.1 Company policy says that all computer equipment is purchased from Dell.



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Requirements Gathering Methods



Learning Objectives

After this lesson, you will be able to:

- List the primary requirements gathering methods
- Construct a well-defined business problem
- Suggestion questions for a user requirements interview

Developing Requirements is a Multi-Step Process

Understand as-is system

Define requirements

Resolve priorities & inconsistencies

Identify improvements

Develop test cases

4 Requirements Gathering Methods

- Interviews
- Prototypes – covered in more detail later
- Document Analysis
- Ethnography



Interviews



A Structured Requirements Interview...

- ...should gather information about the **interview subject**.
- ...should clearly identify the **business problem**.
- ...should clarify the **user environment**.
- ...should test **potential solutions**.
- ... should identify **non-functional requirements**.

Inquiring About the Interviewee

- Name
- Title / role
- Reporting structure
- Barriers / issues in performing job responsibilities



Identifying the Business Problem

- “A problem well-defined is half-solved.”
- Consider these two:
 - “We don’t have a system to take orders online.”
 - “We can’t take orders at the rate that customers are placing them.”
- Follow up (what else is a problem?). Always dig one level deeper.

Identifying the User Environment

- Who will use the system?
- What is the users' level of training and education?
- What platforms do the users use today?
- What are your expectations for time on task?
- What do you expect for training?

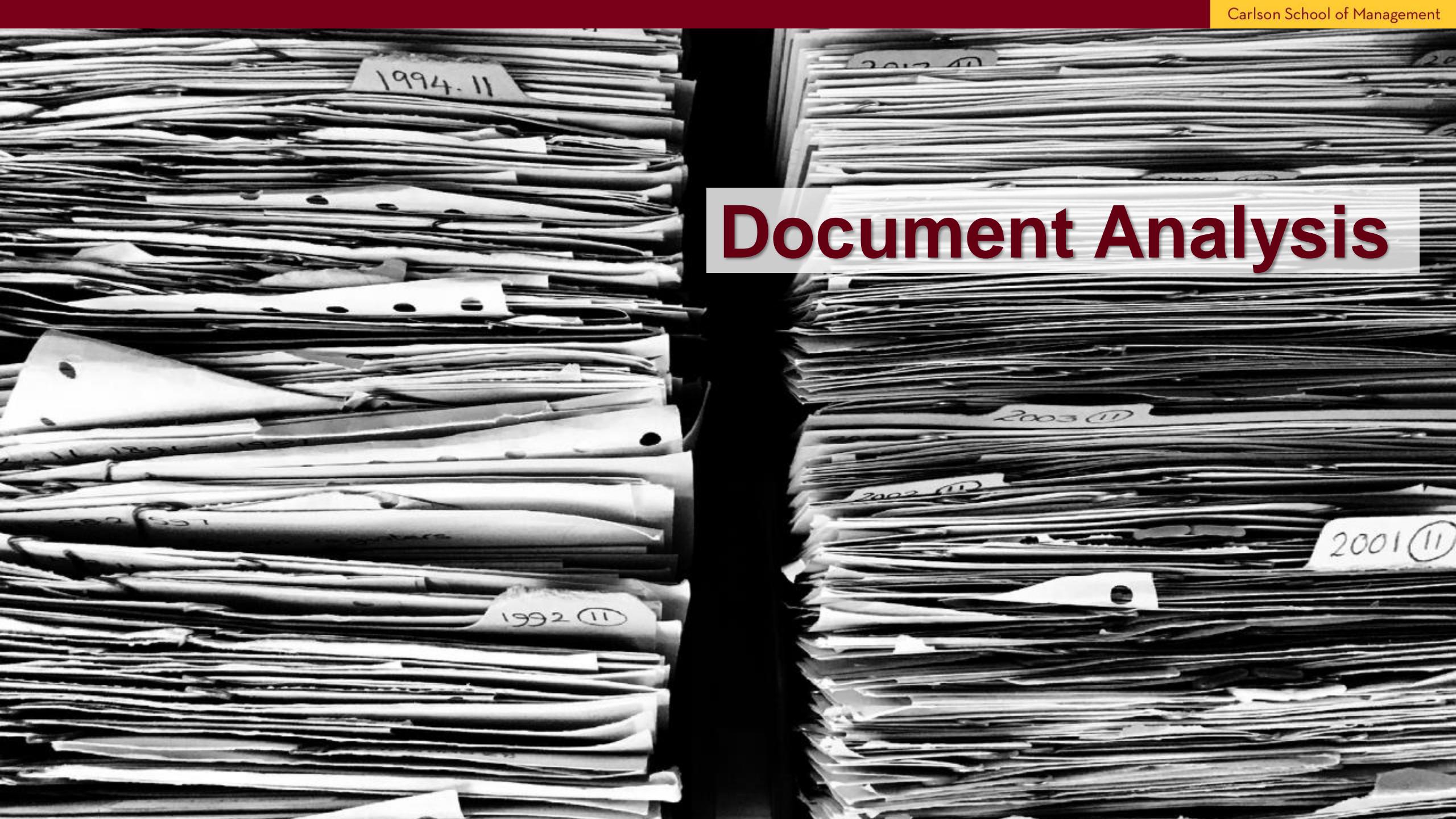
MIND THE GAP

Propose Potential Solutions

- Proposing potential solutions can engage users early in the process
- However, deciding on **the one** solution before gathering all of the requirements can be dangerous
- Summarize the problem and share ideas with the user about potential solutions
 - Emphasize that the final solution is yet to come

Non-functional Requirements

- Availability
- Performance
- Language support
- Security
- Regulatory



Document Analysis

Document Analysis Gives Clues

- Users often poorly approximate processes – even processes they perform every day!
- Document analysis provides an objective look at a process
 - In particular, the information requirements of the process
- Document analysis often informs the other requirements gathering techniques:
 - Interviews
 - Prototypes
 - Ethnography

Ethnography

Ethnography is essentially observing people doing their work.



```
        "timestamp": "2017-06-03T18:42:18.018", "deltaStartMillis": 0, "method": "GET", "sizeChars": "5022", "message": "Duration Log", "durationMillis": 0, "webURL": "/app/page/analyze", "webParams": "null", "class": "com.orgmanager.handlers.RequestHandler", "durationInMillis": 0, "sessionID": "8249868e-afd8-46ac-9745-839146a20f09", "sessionId": "8249868e-afd8-46ac-9745-839146a20f09", "sizeChars": "36"}, {"timestamp": "2017-06-03T18:43:335.030", "deltaStartMillis": 0, "method": "POST", "sizeChars": "10190", "message": "Duration Log", "durationMillis": 0, "webURL": "/app/rest/json/file", "webParams": "file=chartdata_new.json", "class": "com.orgmanager.handlers.RequestHandler", "durationInMillis": 0, "sessionID": "14402n620jm9trnd3s3n7wg0k", "sessionId": "14402n620jm9trnd3s3n7wg0k", "sizeChars": "0"}, {"timestamp": "2017-06-03T18:46:921.000", "deltaStartMillis": 0, "method": "INFO", "sizeChars": "48455", "message": "Duration Log", "durationMillis": 0, "webURL": "/app/page/analyze", "webParams": "file=chartdata_new.json", "class": "com.orgmanager.handlers.RequestHandler", "durationInMillis": 0, "sessionID": "789d89cb-bfa8-4e7d-8047-498454af885d", "sessionId": "789d89cb-bfa8-4e7d-8047-498454af885d", "sizeChars": "7"}, {"timestamp": "2017-06-03T18:46:921.000", "deltaStartMillis": 0, "method": "INFO", "sizeChars": "10190", "message": "Duration Log", "durationMillis": 0, "webURL": "/app/rest/json/file", "webParams": "file=chartdata_new.json", "class": "com.orgmanager.handlers.RequestHandler", "durationInMillis": 0, "sessionID": "7ac6ce95-19e2-4a60-88d7-6ead86e273d1", "sessionId": "7ac6ce95-19e2-4a60-88d7-6ead86e273d1", "sizeChars": "23"}, {"timestamp": "2017-06-03T18:42:18.018", "deltaStartMillis": 0, "method": "GET", "sizeChars": "5022", "message": "Duration Log", "durationMillis": 0, "webURL": "/app/page/analyze", "webParams": "null", "class": "com.orgmanager.handlers.RequestHandler", "durationInMillis": 0, "sessionID": "8249868e-afd8-46ac-9745-839146a20f09", "sessionId": "8249868e-afd8-46ac-9745-839146a20f09", "sizeChars": "36"}, {"timestamp": "2017-06-03T18:43:335.030", "deltaStartMillis": 0, "method": "POST", "sizeChars": "10190", "message": "Duration Log", "durationMillis": 0, "webURL": "/app/rest/json/file", "webParams": "file=chartdata_new.json", "class": "com.orgmanager.handlers.RequestHandler", "durationInMillis": 0, "sessionID": "14402n620jm9trnd3s3n7wg0k", "sessionId": "14402n620jm9trnd3s3n7wg0k", "sizeChars": "0"}, {"timestamp": "2017-06-03T18:46:921.000", "deltaStartMillis": 0, "method": "INFO", "sizeChars": "48455", "message": "Duration Log", "durationMillis": 0, "webURL": "/app/page/analyze", "webParams": "file=chartdata_new.json", "class": "com.orgmanager.handlers.RequestHandler", "durationInMillis": 0, "sessionID": "789d89cb-bfa8-4e7d-8047-498454af885d", "sessionId": "789d89cb-bfa8-4e7d-8047-498454af885d", "sizeChars": "7"}, {"timestamp": "2017-06-03T18:46:921.000", "deltaStartMillis": 0, "method": "INFO", "sizeChars": "10190", "message": "Duration Log", "durationMillis": 0, "webURL": "/app/rest/json/file", "webParams": "file=chartdata_new.json", "class": "com.orgmanager.handlers.RequestHandler", "durationInMillis": 0, "sessionID": "7ac6ce95-19e2-4a60-88d7-6ead86e273d1", "sessionId": "7ac6ce95-19e2-4a60-88d7-6ead86e273d1", "sizeChars": "23"}]
```



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA



Traceability

Learning Objectives



After this lesson, you will be able to:

- Define traceability as it relates to system requirements
- Describe at least one approach to documenting traceability
- List the two types of traceability and why each is important

Developing Requirements is a Multi-Step Process

Understand as-is system

Define requirements

Resolve priorities & inconsistencies

Identify improvements

Develop test cases

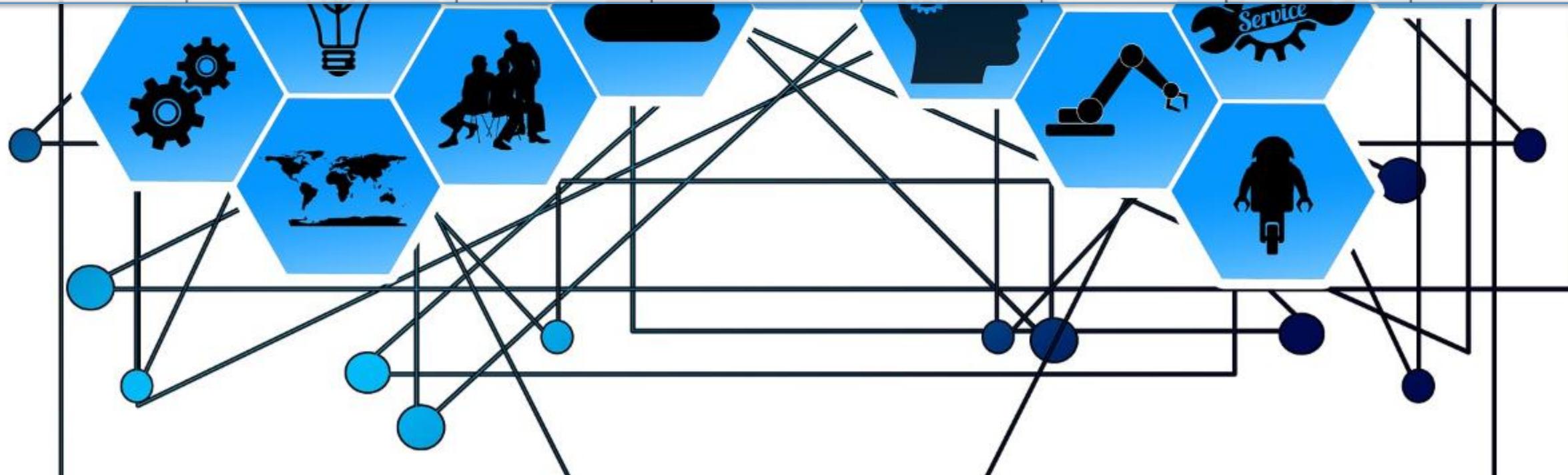
Traceability



Traceability Ties Artifacts Together

- Traceability is a technique that ties all project artifacts together
 - It is a **required** practice in most regulated industries
 - It is a **good idea** for everyone else!
- Most enterprise CASE tools support traceability in some form
- There is a variety of other methods to handle traceability

Requirement Source	Product Requirements	HLD Section #	LLD Section #	Code Unit	UTS Case #	STS Case #	User Manual
Use Case #132 step 6	R00230 Read Gas Flow	7.2.2 Gas Flow Meter Interface	7.2.2 Read Gas Flow Indicator	Read_Gas_Flow.c	UT 7.2.043 UT 7.2.044	ST 230.002 ST 230.003	Section 21.1.2
	R00231 Calculate Gas Price	7.3 Calculate Gas price	7.3 Calculate Gas price	Cal_Gas_Price.c	UT 7.3.005 UT 7.3.006 UT 7.3.007	ST 231.001 ST 231.002 ST 231.003	Section 21.1.3



Requirements Traceability Matrix (RDD Attachment B)

REQ #	Related Use Case #s	Requirement Text	Requirements Categorization					Related ELIS Proviso					
			Requirement Type	Requirement Sub-Type 1	Requirement Sub-Type 2	Priority	Applies To	ELIS Phase	Related Test Cases	Attendance Management	EFS Replacement	Financial Management	Provider Payments
REQ-1	UC-ELC-19	The system shall provide the ability to sort the attendance roster by user-configurable criteria (e.g., child name, care level, classroom).	Functional	Attendance Management	Advanced Search	High	ELC	Phase - 1	TC-ELC-19	✓			
REQ-2	UC-ELC-21	The system shall provide the ability for prior period attendance adjustment transactions to be queried through a variety of criteria (e.g., child attendance information, SR / VPK Payment information, date of attendance adjustment).	Functional	Attendance Management	Advanced Search	High	ELC	Phase - 1	TC-ELC-21	✓			
REQ-3	UC-ELC-19	The system shall provide the ability to generate a notification to Early Learning Coalition staff indicating an attendance roster has been submitted.	Functional	Attendance Management	Communications / Notifications	High	ELC	Phase - 1	TC-ELC-19	✓			
REQ-4	UC-ELC-19	The system shall provide the ability to generate an attendance roster based on a user-defined schedule (with various submission frequencies).	Functional	Attendance Management	Events	High	ELC	Phase - 1	TC-ELC-19	✓			
REQ-5	UC-ELC-19	The system shall provide the ability to enter child information during the attendance submission (e.g., day attended, hours attended).	Functional	Attendance Management	N/A	High	ELC	Phase - 1	TC-ELC-19	✓			
REQ-6	UC-ELC-19	The system shall provide the ability to enter the termination of service for a child during attendance entry.	Functional	Attendance Management	N/A	High	ELC	Phase - 1	TC-ELC-19	✓			
REQ-7	UC-ELC-19	The system shall prevent the addition of children to the attendance roster through attendance processing.	Functional	Attendance Management	N/A	High	ELC	Phase - 1	TC-ELC-19	✓			
REQ-8	UC-ELC-19	The system shall provide the ability to generate the attendance roster based on the child information in the system (e.g., enrollment date, billing group, eligibility group, care level, child's calendar).	Functional	Attendance Management	N/A	High	ELC	Phase - 1	TC-ELC-19	✓			
REQ-9	UC-ELC-19	The system shall provide the ability to enter attendance for multiple care levels for the same child on the same attendance roster.	Functional	Attendance Management	N/A	High	ELC	Phase - 1	TC-ELC-19	✓			
REQ-10	UC-ELC-19	The system shall provide the ability to calculate the amount due to an SR / VPK Provider for the approved attendance roster.	Functional	Attendance Management	N/A	High	ELC	Phase - 1	TC-ELC-19	✓			
REQ-11	UC-ELC-19	The system shall provide the ability to identify attendance errors by severity (e.g., severe, moderate, trivial).	Functional	Attendance Management	N/A	High	ELC	Phase - 1	TC-ELC-19	✓			
REQ-12	UC-ELC-19	The system shall provide the ability to create financial transactions at the time of attendance roster approval (e.g., appropriate allocations to billing groups, eligibility groups, other cost accumulators, pre-payment reduction).	Functional	Attendance Management	N/A	High	ELC	Phase - 1	TC-ELC-19	✓			
REQ-13	UC-ELC-19	The system shall prevent the user approving the pending attendance submission to be the same user who created the pending attendance submission.	Functional	Attendance Management	N/A	High	ELC	Phase - 1	TC-ELC-19	✓			
REQ-14	UC-ELC-19	The system shall provide the ability to create financial entries (e.g., payment, fund allocation, other cost accumulator codes, service pre-payment allocation) based on the prior period attendance adjustment.	Functional	Attendance Management	N/A	High	ELC	Phase - 1	TC-ELC-19	✓			
REQ-15	UC-ELC-19	The system shall provide the ability to maintain user-defined business rules for the types of absences (e.g., number of days of vacation allowed per service, un-excused absences for at-risk children).	Functional	Attendance Management	N/A	High	ELC	Phase - 1	TC-ELC-19	✓			
REQ-16	UC-ELC-19	The system shall provide the ability to reconcile match funds incurred prior to performing the prior period attendance adjustment to the match funds incurred following the approval of the prior period attendance adjustment.	Functional	Attendance Management	N/A	High	ELC	Phase - 1	TC-ELC-19	✓			
REQ-17	UC-ELC-19	The system shall provide the ability to search and present any existing information stored in the system associated with tracking and analyzing the developmental assessment process and results.	Functional	Attendance Management	N/A	High	ELC	Phase - 1	TC-ELC-19	✓			

Other Ways to Handle Traceability

19	After Sales Support	Recall Management	Interview 9/5/04	Ability to track and report on all aspects of product genealogy and distribution history, in order to quickly and efficiently manage product recall.
20	Demand-to-Supply Proposal-to-Revenue	Supply Chain Visibility	S2STh	Ability to manage worldwide inventory (at supplier; in field sales; on plant floor; on customer consignment, and in distribution network) through: <ul style="list-style-type: none">visibility to cross-site inventory; supplier inventory; and supplier capabilitiesability to understand the impact of product expirationability to quickly respond to shortages and unplanned demand with a view of time-phased supply-and-demand at each supply chain nodetools that may allow tracking of customer-owned inventory, either through integrations to hospital systems or calculations of 'shipped product – implanted product.'

Traceability Throughout the SDLC

- Id problem
- Calculate value
- Draft an approach
- Create the plan

Plan

Analyze

- Clarify requirements
- Understand current process
- Develop changes to business process

- Translate requirements into system specifications

Design

Implement

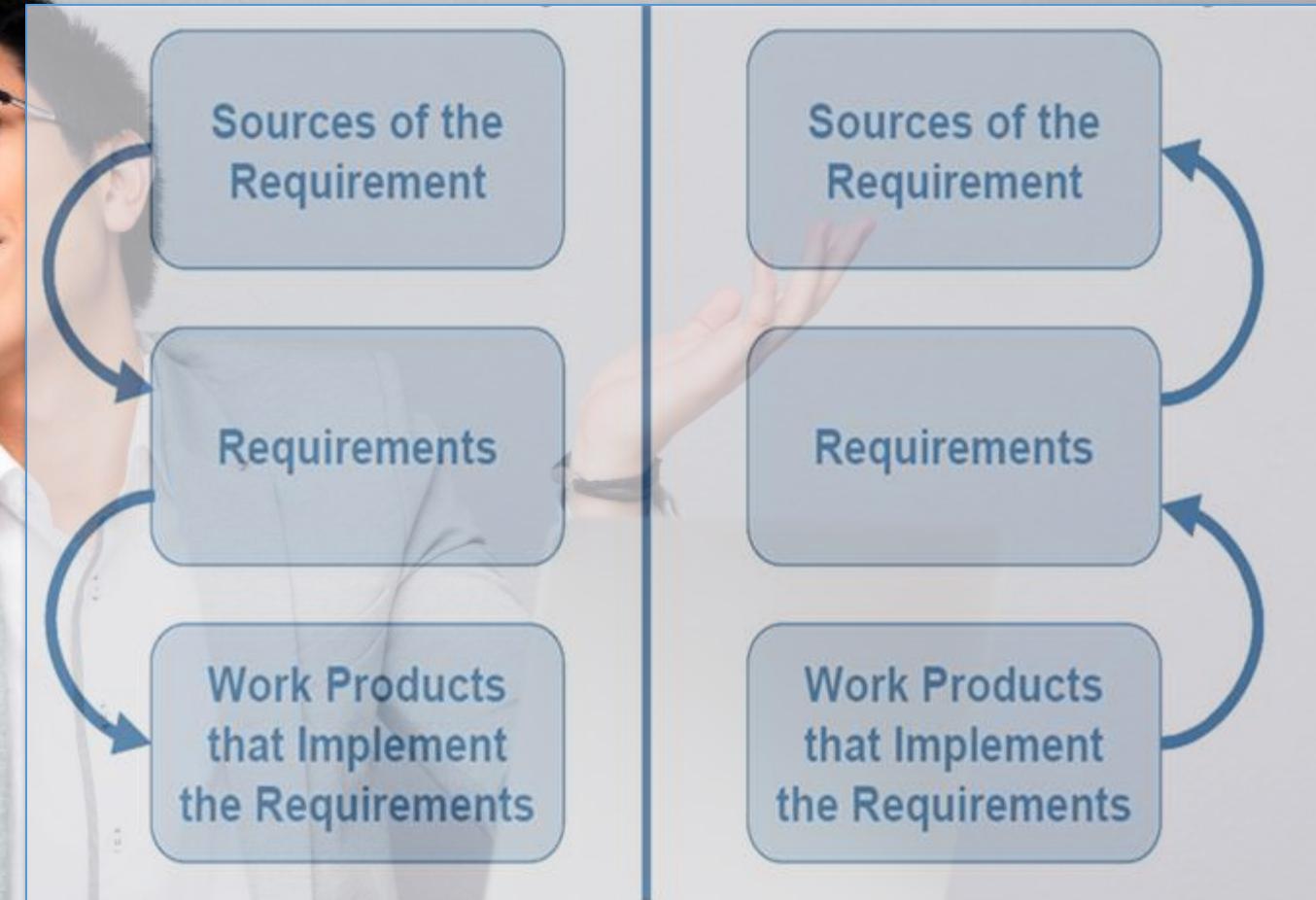
- Program and install software
- Modify processes
- Ensure change is managed



Why bother?

Forward traceability moves forward through the systems development life cycle. Starting with the source of the requirement, making sure that it appears in the requirements documents and then further making sure that it appears in test cases, then it's finally tested and in the released product.

Backward traceability starts at the end of the system and its development life cycle, and moves backwards. It starts with an end product, and then moves backwards to make sure that we can find evidence of those features in the test cases that the test cases are tied to requirements, and that the requirements actually came from a business user.





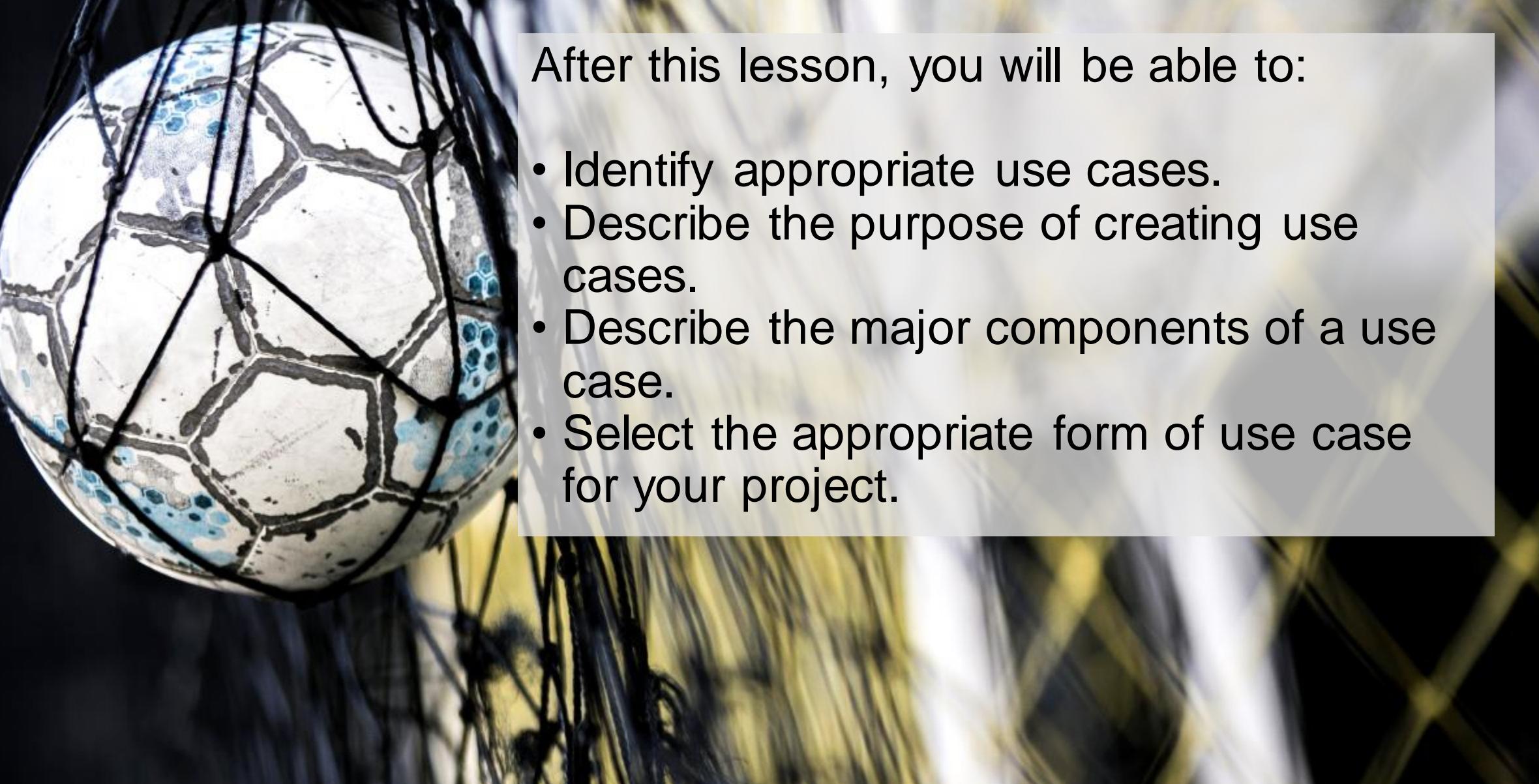
CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA



Use Cases

Learning Objectives



After this lesson, you will be able to:

- Identify appropriate use cases.
- Describe the purpose of creating use cases.
- Describe the major components of a use case.
- Select the appropriate form of use case for your project.

Analysis Phase Results in a System Proposal

The purpose of use cases is to validate and improve your requirements. In other words, we write use cases so that we can refine the requirements that we wrote in our earlier step.

A secondary purpose of writing use cases is for the testing team. Use cases help the testers understand how the system will be used in the context of its own process. They can then use this to plan their testing of the system to make sure that it meets the requirements.

new artifacts
in analyze phase

1. Table of Contents

2. Executive Summary

A summary of all the essential information in the proposal so that a busy executive can read it quickly and decide what parts of the plan to read in more depth.

3. System Request

The revised system request form. (See Chapter 1.)

4. Work plan

The original work plan, revised after having completed the analysis phase. (See Chapter 2.)

5. Feasibility Analysis

A revised feasibility analysis, using the information from the analysis phase. (See Chapter 1.)

6. Requirements Definition

A list of the functional and nonfunctional business requirements for the system (this chapter).

7. Use Cases

A set of use cases that illustrate the basic processes that the system needs to support. (See Chapter 4.)

8. Process Model

A set of process models and descriptions for the to-be system. (See Chapter 5.) This may include process models of the current as-is system that will be replaced.

9. Data Model

A set of data models and descriptions for the to-be system. (See Chapter 6.) This may include data models of the as-is system that will be replaced.

Appendices

These contain additional material relevant to the proposal, often used to support the recommended system. This might include results of a questionnaire survey or interviews, industry reports and statistics, etc.

Use Cases Validate and Improve Requirements

Validate & improve functional requirements

Prepare for testing

Initial Functional Requirements for Creating a Customer Offer (from Figure 3-3)

- The system will enable salespersons to create a customer offer (2.1).
- The system will allow salespeople to know whether an offer is pending on a specific vehicle (2.2).

Revised Functional Requirements for Creating a Customer Offer (based on UC-3, Figure 4-11)

- The system shall obtain the offer vehicle from the salesperson.
- The system shall search all Pending Offers to determine if the offer vehicle has a Pending Offer.
- The system shall notify the salesperson if a pending offer found for the offer vehicle, and the process terminates.
- The system shall use the salesperson's entry of "new offer" or "revised offer" to create a new offer with vehicle details supplied from the Vehicle datastore or will fill the offer with the previous offer details obtained from the Rejected Offers datastore.
- The system shall allow the salesperson to complete and/or modify information on the offer.
- The system shall display a complete summary of the offer before it is confirmed by the customer.
- The system allows the offer to be confirmed by the customer or cancelled.
- The system shall store new confirmed offers as a new Pending Offer in the Pending Offers datastore.
- The system shall enable copies of the Pending Offer to be printed.
- The system shall send a notice of a new Pending Offer to the Sales Manager.

What is a Use Case?

The use case diagram is like a table of contents or an index into the use cases for our system.

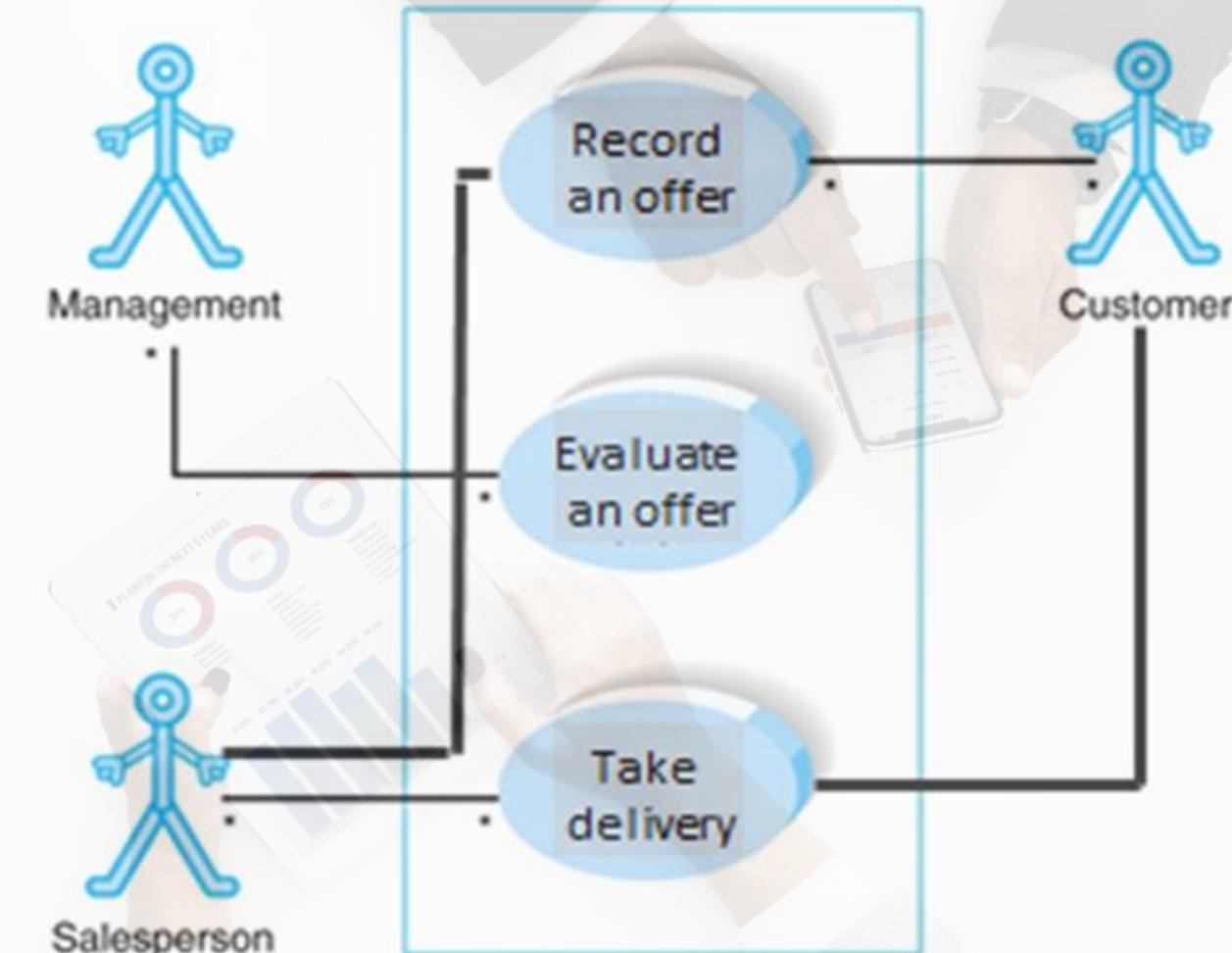
A Use Case Diagram does not describe, for example, the individual steps of a use case, instead it describes the relationship of several use cases to each other.

Three use cases to record an offer, evaluate an offer, and take delivery.

The salesperson, management, and the customer in this case are all actors in the use case, in other words, they cause action to happen.

Use Case Diagram

Vehicle Sales System



Use Case Process



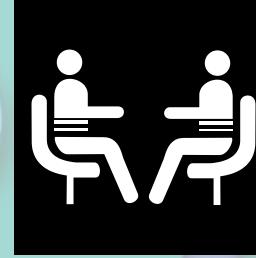
+ draw UC diagram

Step	Activities
1. Identify the use cases.	Start a use case report form for each use case by filling in the name, description and trigger. If there are more than nine use cases, group them into packages.
2. Identify the major steps within each use case.	For each use case, fill in the major steps needed to complete the task.
3. Identify elements within steps.	For each step, identify its triggers and its inputs and outputs.
4. Confirm the use case.	For each use case, validate that it is correct and complete.

Identifying Use Cases



Review requirements



Listen!

Event	Response
1) New vehicles are needed for inventory.	Purchase order is placed with vehicle manufacturer. Vehicle information is recorded on new vehicle record.
2) New vehicles arrive from manufacturer.	
3) Customer makes an offer on a new vehicle.	Details of offer are recorded and presented to owner for acceptance decision.
4) Customer offer is accepted.	Details of accepted offer are recorded on a sales contract, and customer provides a deposit.
5) Customer takes delivery on new vehicle.	Customer pays for vehicle once offer is accepted, takes possession of the vehicle, and turns in trade-in. Details of the entire purchase are saved.
6) Trade-in is added to used vehicle inventory.	Used vehicle information is recorded on used vehicle form.

Jot down events (if needed)

1. Record an offer
2. Accept an offer
3. Take delivery

Identify which events are worthy of use cases



Use Case Name: Request a chemical	Activity of company and Data and prognosis of activity	600	ID: UC-2	Priority: High
Actor: Lawn Chemical Applicator (LCA)				
Description: The Lawn Chemical Applicator (LCA) specifies the lawn chemical needed for a job by entering its name or ID number. The system satisfies the request by reserving the quantity requested or the quantity available and notifying the Chemical Supply Warehouse of the pick-up.				
Trigger: A Lawn Chemical Applicator (LCA) needs a chemical for a job.				Actor:
Type: <input checked="" type="checkbox"/> External <input type="checkbox"/> Temporal				Importance:
Preconditions:				Trigger
1. The LCA identity is authenticated. 2. The LCA has necessary training and credentials on file. 3. The Chemical Supply datastore is up-to-date and on-line.				Use Case Name: Functional Goal

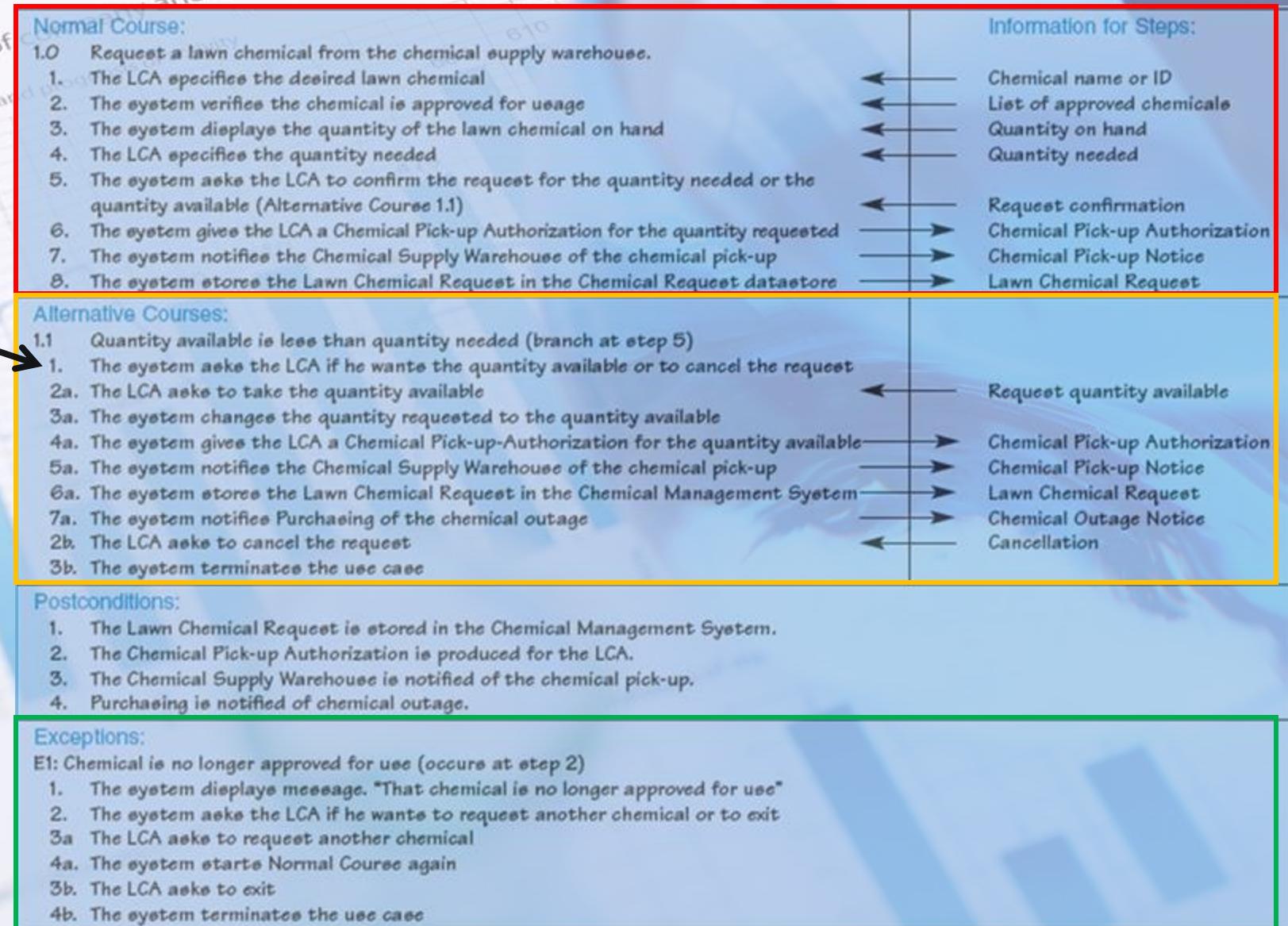
The use case names should essentially start with a verb, and it should be a strong phrase.

Components of individual use cases:

- Actors
- Description
- Trigger
- Trigger Type
- Preconditions

Steps

- Alternative courses
- Exceptions
- Conditional step



Alternative Courses:

- 1.1 Quantity available is less than quantity needed (branch at step 5)
1. The system asks the LCA if he wants the quantity available or to cancel the request
 - 2a. The LCA asks to take the quantity available
 - 3a. The system changes the quantity requested to the quantity available
 - 4a. The system gives the LCA a Chemical Pick-up-Authorization for the quantity available
 - 5a. The system notifies the Chemical Supply Warehouse of the chemical pick-up
 - 6a. The system stores the Lawn Chemical Request in the Chemical Management System
 - 7a. The system notifies Purchasing of the chemical outage
 - 2b. The LCA asks to cancel the request
 - 3b. The system terminates the use case

Request quantity available

Chemical Pick-up Authorization

Chemical Pick-up Notice

Lawn Chemical Request

Chemical Outage Notice

Cancellation

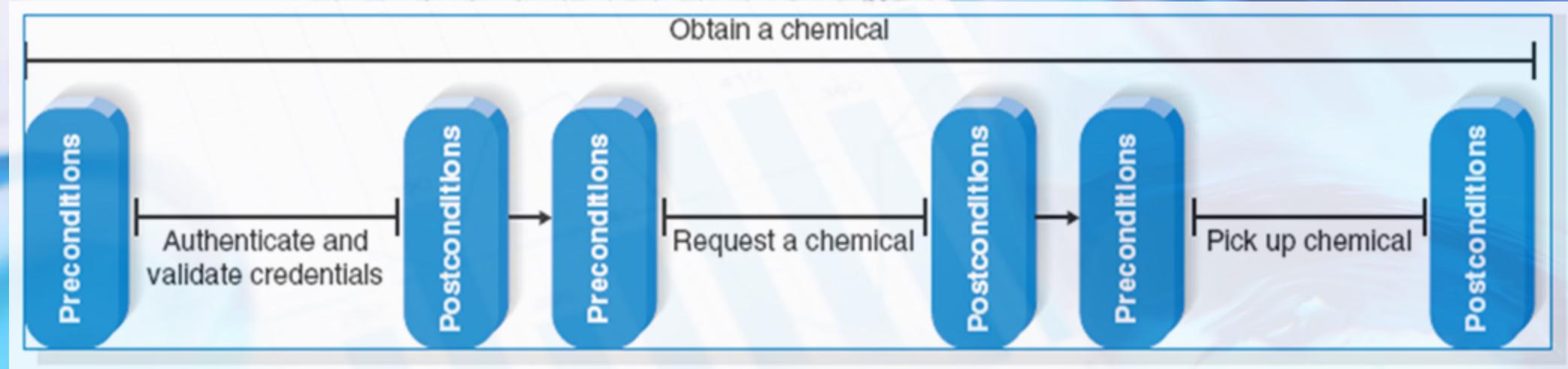
Data required for step

Summary Inputs	Source	Outputs	Destination
Chemical name or ID	LCA	Chemical Pick-up Authorization	LCA
List of approved chemicals	Lawn Chemicals Supply datastore	Chemical Pick-up Notice	Chemical Supply Warehouse
Chemical quantity on hand	Lawn Chemicals Supply datastore	Lawn Chemical Request	Chemical Request datastore
Quantity needed	LCA		Purchasing
Request confirmation	LCA		
Request quantity available or cancellation	LCA	Chemical Outage Notice	

Summary Inputs & outputs

At the bottom of our example use case format, you have a summary. These are the inputs and outputs of the system. This can be used to get a quick summary of the use case and the data needed in order to perform it, and this is useful when designing the system.

Use Cases Might Include Pre- and Post- Conditions



The pre- and post-conditions describe things that must be true before the use case can be initiated, or post-conditions describe things that will be true after the use case is completed.

The pre-conditions above are:

- 1- Authenticate and validate credentials
- 2- Requests a chemical
- 3- Pickup chemical

How does the previous Use Case fit into this flow?

Why not write one very large Use Case?

Reuse of Use Cases



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

• Preconditions

• Post conditions

Use Case Name:	Request a chemical	ID:	UC-2	Priority:	High
Actor:	Lawn Chemical Applicator (LCA)				
Description:	The Lawn Chemical Applicator (LCA) specifies the lawn chemical needed for a job by entering its name or ID number. The system satisfies the request by reserving the quantity requested or the quantity available and notifying the Chemical Supply Warehouse of the pick-up.				
Trigger:	A Lawn Chemical Applicator (LCA) needs a chemical for a job.				
Type:	<input checked="" type="checkbox"/> External <input type="checkbox"/> Temporal				
Preconditions:	<ol style="list-style-type: none"> 1. The LCA identity is authenticated. 2. The LCA has necessary training and credentials on file. 3. The Chemical Supply datastore is up-to-date and on-line. 				
Normal Course:	<ol style="list-style-type: none"> 1.0 Request a lawn chemical from the chemical supply warehouse. 1. The LCA specifies the desired lawn chemical 2. The system verifies the chemical is approved for usage 3. The system displays the quantity of the lawn chemical on hand 4. The LCA specifies the quantity needed 5. The system asks the LCA to confirm the request for the quantity needed or the quantity available (Alternative Course 1.1) 6. The system gives the LCA a Chemical Pick-up Authorization for the quantity requested 7. The system notifies the Chemical Supply Warehouse of the chemical pick-up 8. The system stores the Lawn Chemical Request in the Chemical Request datastore 				Information for Steps:
		←	Chemical name or ID		
		←	List of approved chemicals		
		←	Quantity on hand		
		←	Quantity needed		
		→	Request confirmation		
		→	Chemical Pick-up Authorization		
		→	Chemical Pick-up Notice		
		→	Lawn Chemical Request		
Alternative Courses:	<ol style="list-style-type: none"> 1.1 Quantity available is less than quantity needed (branch at step 5) <ol style="list-style-type: none"> 1. The system asks the LCA if he wants the quantity available or to cancel the request 2a. The LCA asks to take the quantity available 3a. The system changes the quantity requested to the quantity available 4a. The system gives the LCA a Chemical Pick-up-Authorization for the quantity available 5a. The system notifies the Chemical Supply Warehouse of the chemical pick-up 6a. The system stores the Lawn Chemical Request in the Chemical Management System 7a. The system notifies Purchasing of the chemical outage 2b. The LCA asks to cancel the request 3b. The system terminates the use case 				Request quantity available
		→	Chemical Pick-up Authorization		
		→	Chemical Pick-up Notice		
		→	Lawn Chemical Request		
		→	Chemical Outage Notice		
		←	Cancellation		
Postconditions:	<ol style="list-style-type: none"> 1. The Lawn Chemical Request is stored in the Chemical Management System. 2. The Chemical Pick-up Authorization is produced for the LCA. 3. The Chemical Supply Warehouse is notified of the chemical pick-up. 4. Purchasing is notified of chemical outage. 				
Exceptions:	E1: Chemical is no longer approved for use (occurs at step 2)				
	1. The system displays message, "That chemical is no longer approved for use"				
	2. The system asks the LCA if he wants to request another chemical or to exit				
	3a. The LCA asks to request another chemical				
	4a. The system starts Normal Course again				
	3b. The LCA asks to exit				
	4b. The system terminates the use case				
Summary					
Inputs		Source		Outputs	
Chemical name or ID List of approved chemicals Chemical quantity on hand Quantity needed Request confirmation Request quantity available or cancellation	LCA Lawn Chemical Supply datastore Lawn Chemical Supply datastore LCA LCA LCA		Chemical Pick-up Authorization Chemical Pick-up Notice Lawn Chemical Request Chemical Outage Notice	Chemical Pick-up Authorization Chemical Pick-up Notice Lawn Chemical Request Chemical Outage Notice	LCA Chemical Supply Warehouse Chemical Request datastore Purchasing

Use Case Richness



The level of detail largely depends on the culture of the organization, along with the market that organization might be operating in.

Highly regulated environments, like medical devices, typically have requirements for exhaustive use cases, fully dressed.

On the other hand, organizations that might have a more agile approach might rely on brief use cases.

- **Fully-dressed** (exhaustive, numbered with data flows)
- **Casual** (approximate, numbered steps or paragraphs)
- **Briefs** (a few sentences)

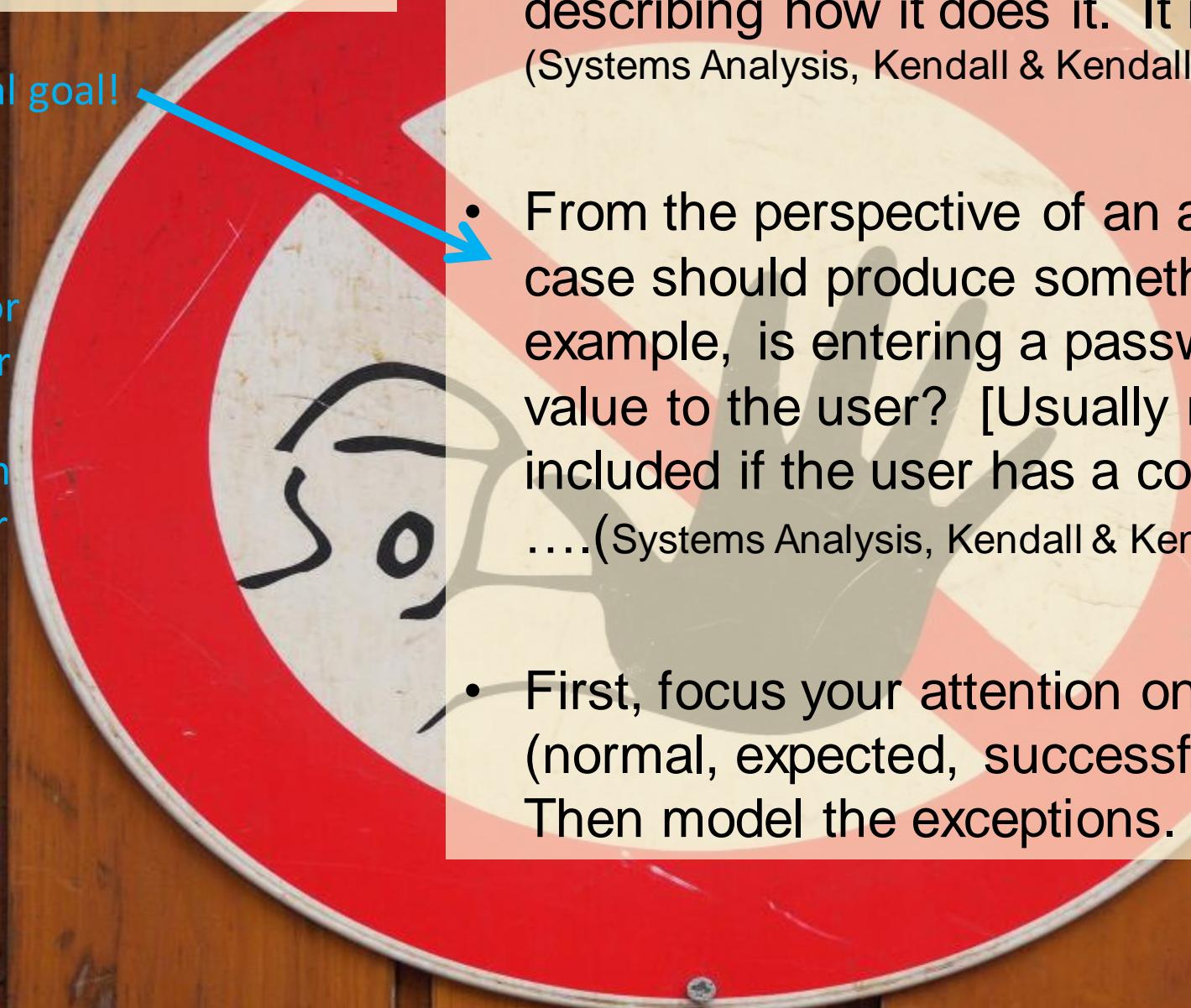
A Brief Just Lays Out The Key Steps

- **Use Case #2:** Request a chemical
- **Steps**
 1. Specify the chemical and quantity
 2. If sufficient inventory isn't on hand, order more
 3. Give the LCA a pick-up authorization slip

Watch Out For...

Concept of functional goal!

A use case for an actor to log in to a particular system doesn't necessarily accomplish something of value for the user unless that system happens to be operating in an area where security is ultimately very, very important.



- Use cases describe what a system does without describing how it does it. It is a logical model.... (Systems Analysis, Kendall & Kendall)
- From the perspective of an actor (or user), a use case should produce something of value. For example, is entering a password something of value to the user? [Usually not, but] it may be included if the user has a concern about security(Systems Analysis, Kendall & Kendall)
- First, focus your attention on the **happy path** (normal, expected, successful completion). Then model the exceptions.



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA



Use Case Diagrams

Learning Objectives

After this video, you will be able to:

- Describe the role of a use case diagram.
- Define the two major types of use case reuse.
- Organize your use cases using packages.

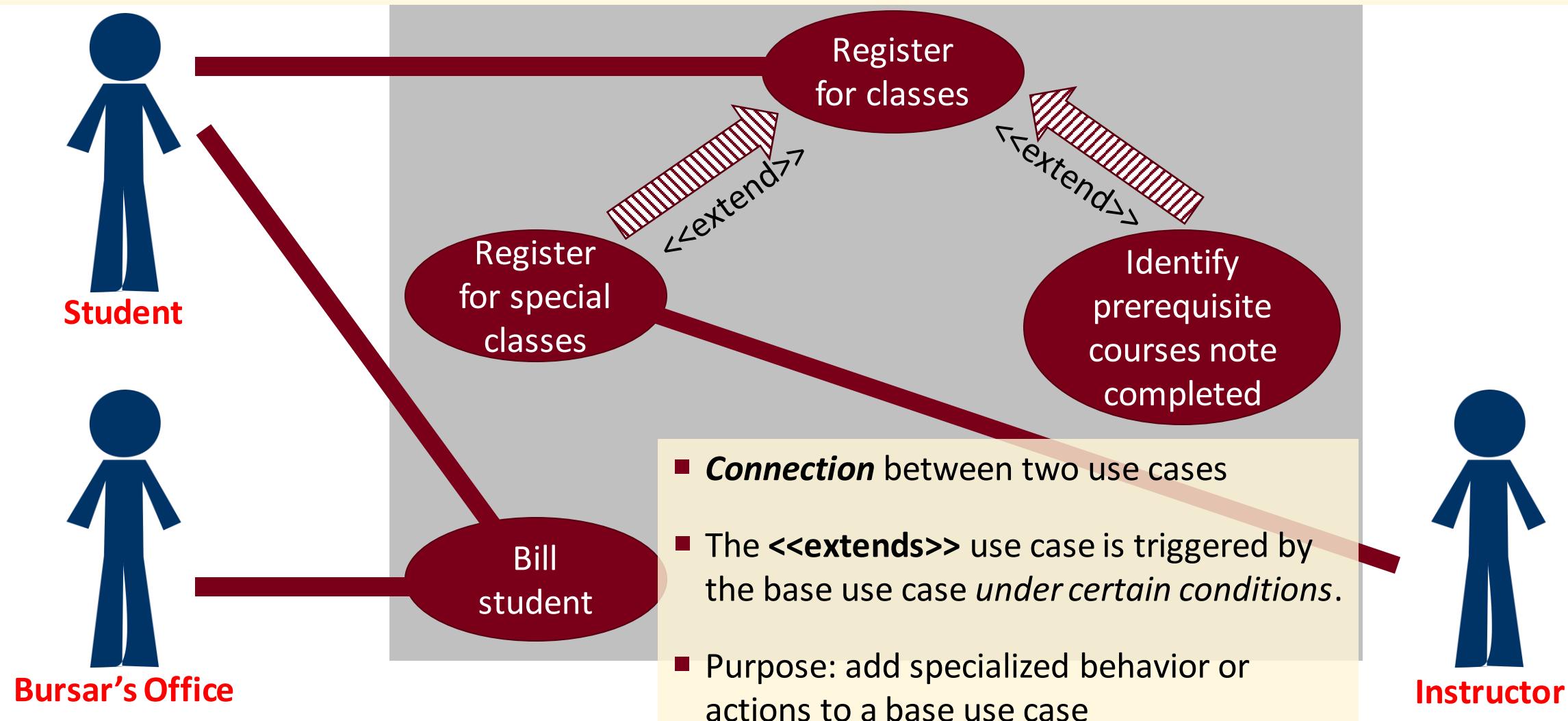
Use Case Diagrams Show Relationships

- A use case diagram shows the relationship among use cases
- It is a “table of contents” to a set of use cases, which may number in the hundreds!
- Relationships among use cases can be complex. UML use case diagram syntax attempts to model these complexities.
- What tools help us draw use case diagrams?

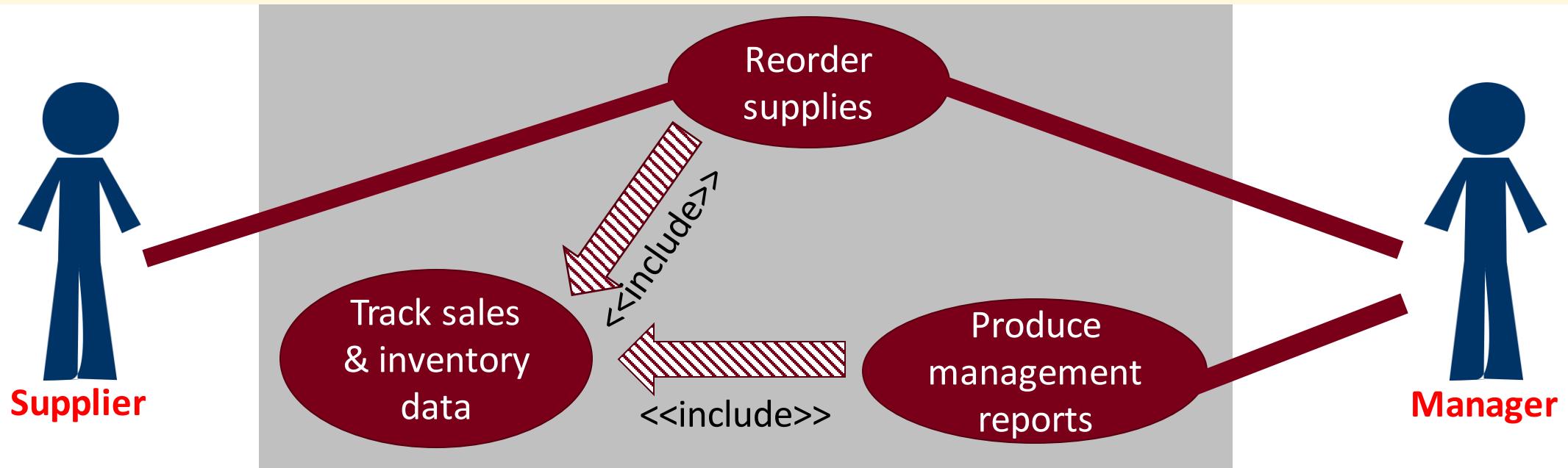
Use Case Diagrams Contain Additional Features

- What if use cases are related to each other?
- What if actors are related to each other?
- What if there are too many use cases?

<<extend>> is triggered *under certain conditions*

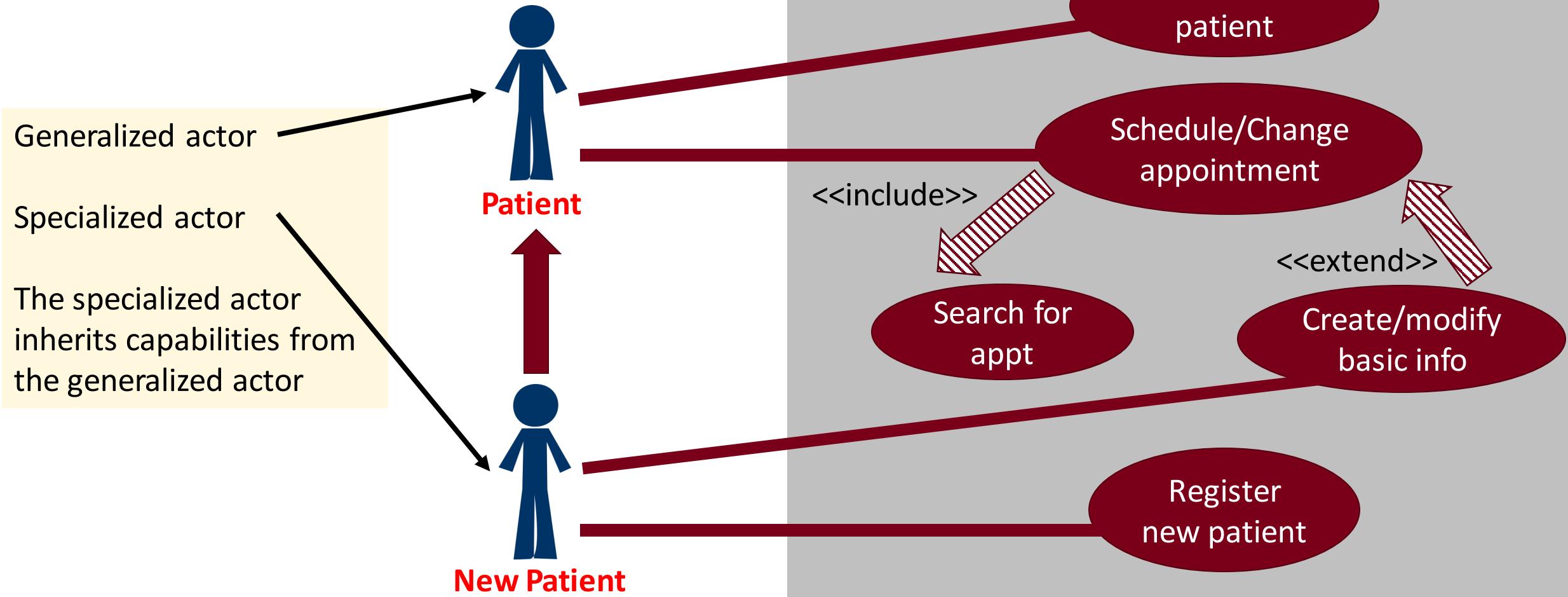


<<include>> is triggered *under all conditions*

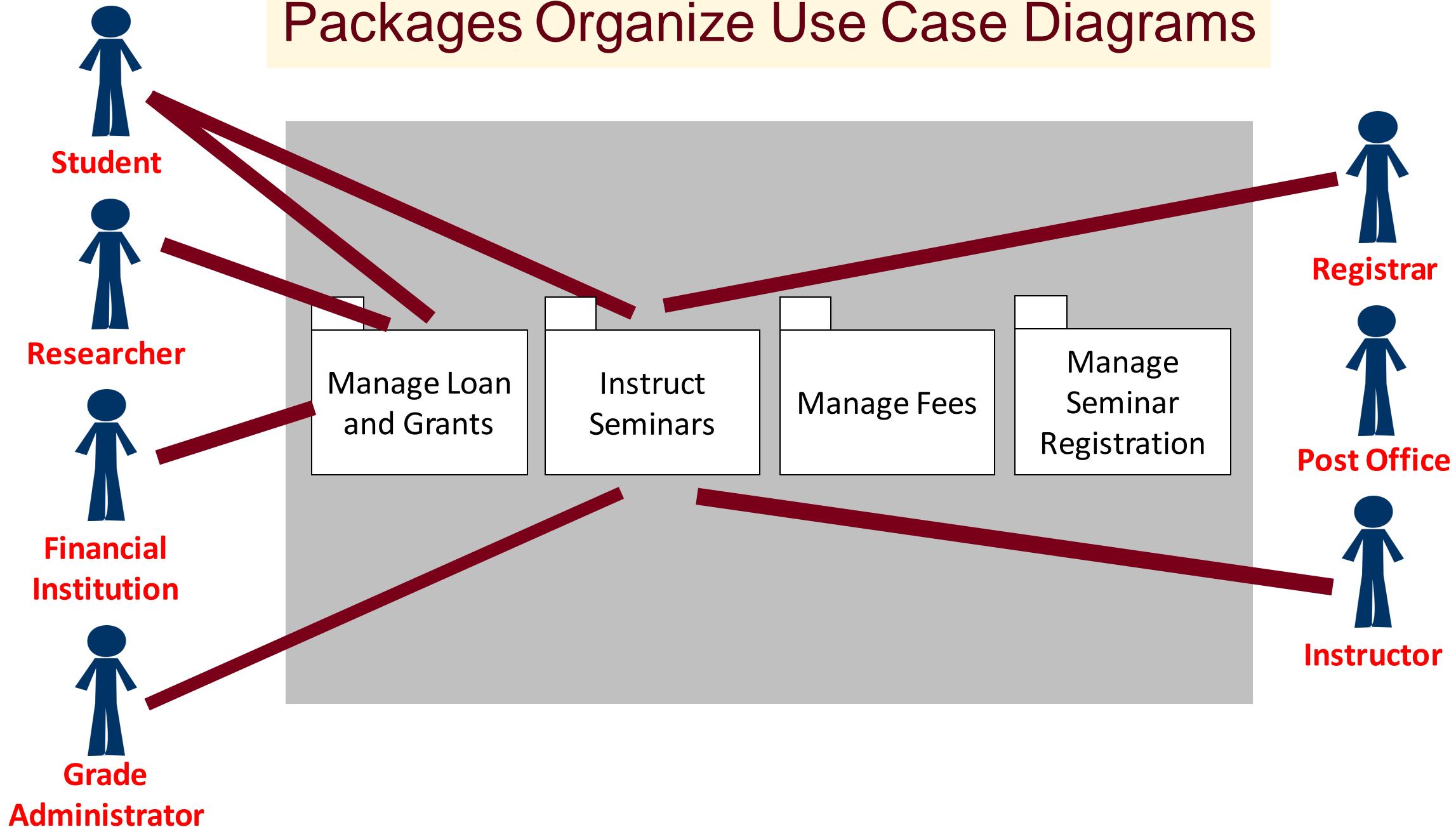


- **Connection** between two use cases
- The <<includes>> use case is triggered by the base use case *under all conditions*.
- Purpose
 - Simplify the writing of the base use case (by carving out a new use case for complex logic), and/or
 - Allowing a specific function to be used by many use cases (simplifying diagram)
- [Note: <<include>> is called <<use>> in Visio]

A Specialized Actor Inherits Capabilities



Packages Organize Use Case Diagrams





CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA



Test Cases



Learning Objectives

After this video, you will be able to:

- Describe how test uses fit into the system development life cycle.
- Define a test case.
- Describe the key components of a good test case.

Developing Requirements is a Multi-Step Process

Understand as-is system

Define requirements

Resolve priorities & inconsistencies

Identify improvements

Develop test cases

Developing Requirements is a Multi-Step Process

Understand as-is system

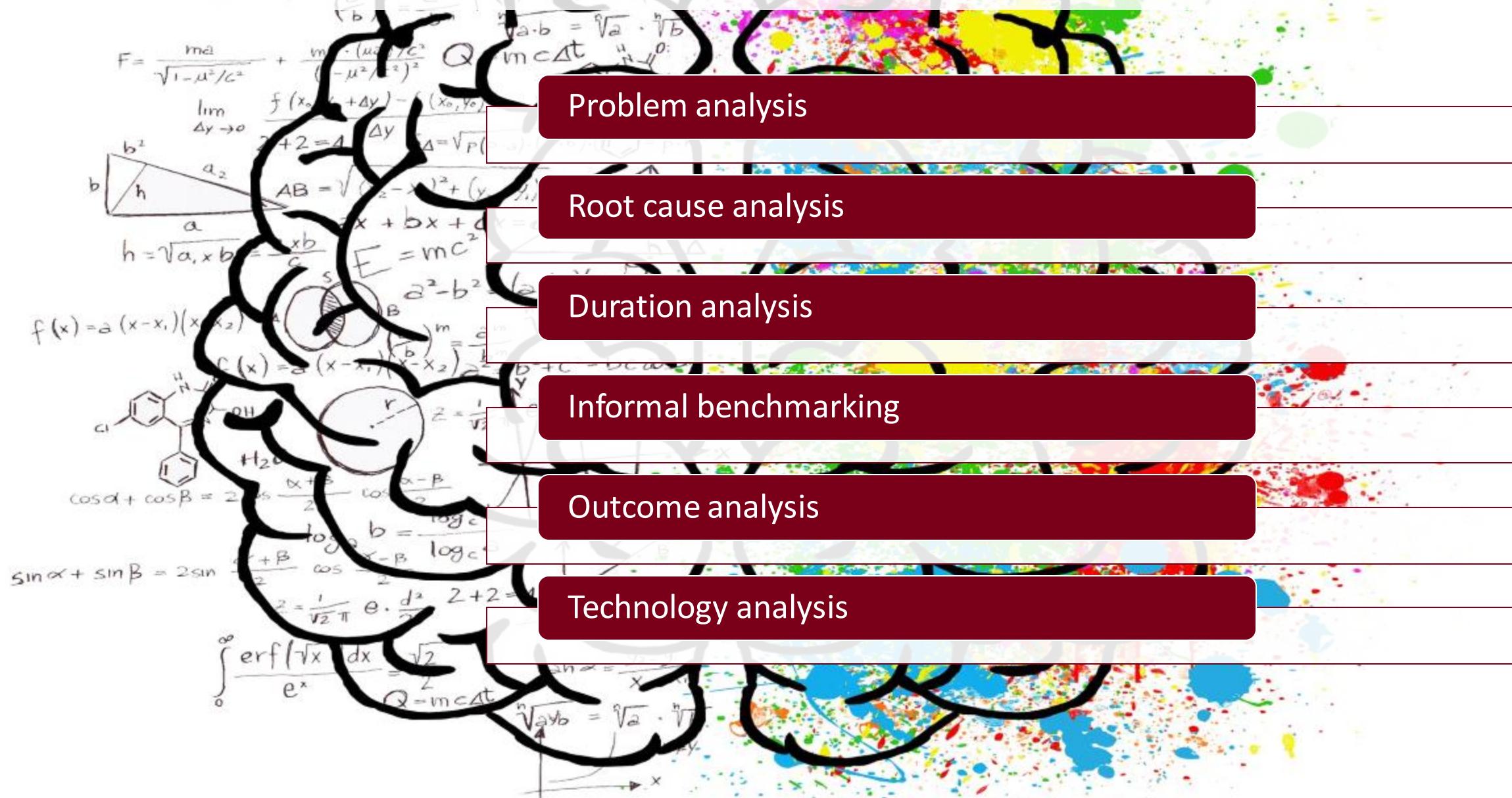
Define requirements

Resolve priorities & inconsistencies

Identify improvements

Develop test cases

Identifying Improvements is a Creative Process



Developing Requirements is a Multi-Step Process

Understand as-is system

Define requirements

Resolve priorities & inconsistencies

Identify improvements

Develop test cases

What is a test case?

IEEE Standard 610 defines test case as follows:

- “(1) A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.
- “(2) (IEEE Std 829-1983) Documentation specifying inputs, predicted results, and a set of execution conditions for a test item.”

According to Ron Patton:

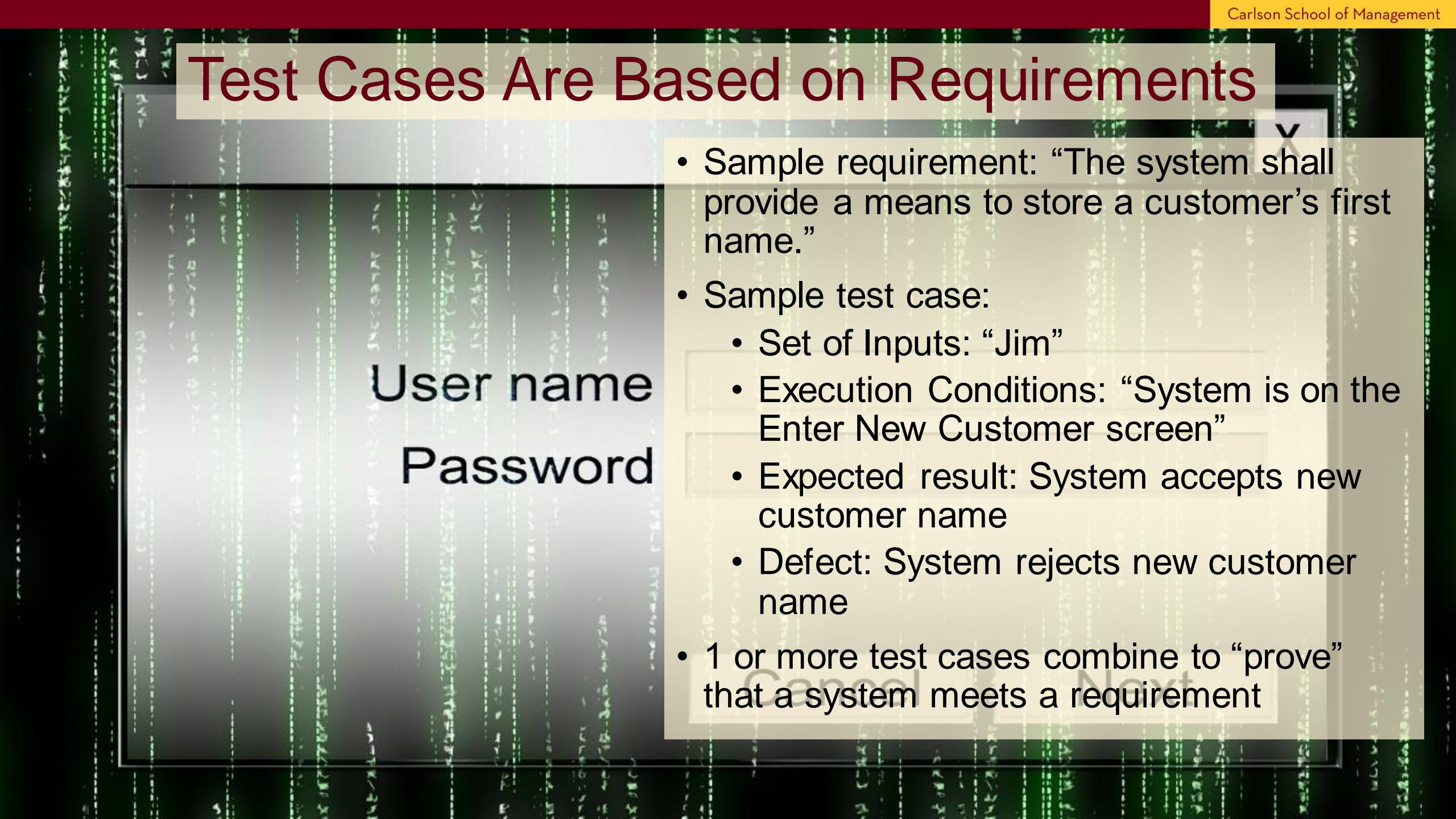
“Test cases are the specific inputs that you'll try and the procedures that you'll follow when you test the software.”

Boris Beizer defines a test as:

“A sequence of one or more subtests executed as a sequence because the outcome and/or final state of one subtest is the input and/or initial state of the next. The word ‘test’ is used to include subtests, tests proper, and test suites.”

(excerpted from “Writing Good Test Cases”)

Test Cases Are Based on Requirements



User name
Password

- Sample requirement: “The system shall provide a means to store a customer’s first name.”
- Sample test case:
 - Set of Inputs: “Jim”
 - Execution Conditions: “System is on the Enter New Customer screen”
 - Expected result: System accepts new customer name
 - Defect: System rejects new customer name
- 1 or more test cases combine to “prove” that a system meets a requirement

Test Cases Can Be Simple or Complex

Test Case Template						
Project Name:						
Test Case ID: Fun_10		Test Designed by: <Name>				
Test Priority (Low/Medium/High): Med		Test Designed date: <Date>				
Module Name: Google login screen		Test Executed by: <Name>				
Test Title: Verify login with valid username and password		Test Execution date: <Date>				
Description: Test the Google login page.						
Pre-conditions: User has valid username and password						
Dependencies:						
Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to login page	User: example@gmail.com	User should be able to login	User is navigated to login	Pass	
2	Provide valid username	Password: 1234				
(From Step 1)	Provide valid password	testinghelp.com				
4	Click on Login button					
Post-conditions:						
User is validated with database and successfully login to account. The account session details are logged in database.						

Developing Test Cases

2. Vehicle Sales Management

- 2.1 The system will enable salespersons to create a customer offer.
- 2.2 The system will allow salespeople to know whether an offer is pending on a specific vehicle.
- 2.3 The system will enable managers to record approval of a customer offer.

Using salesperson login

- 2.1a Add a customer offer with complete information
- 2.1b Add a customer offer with missing address
- 2.1c Add a customer offer without a first name
- 2.1d Add a duplicate offer customer

Using non-salesperson login

- 2.1e Add a customer with complete information

5 test cases for 1 requirement!
Sometimes, it can be much more.

Each should include specific inputs,
execution conditions, and expected
results.

From Microsoft Excel to HP Quality Center,
solutions abound for test case
management.



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA



Return to Analysis Phase



Learning Objectives



After this lesson, you will be able to:

- Describe the purpose of a data flow diagram.
- List the components of a data flow diagram.
- Describe the relationship between a use case diagram and data flow diagram.

SDLC Roadmap

Inefficient process

Technology enabled process

- Id problem
- Calculate value
- Draft an approach
- Create the plan

Plan

Analyze

- Clarify requirements
- Understand current process
- Develop changes to business process

System Proposal

- Translate requirements into system specifications

Design

Implement

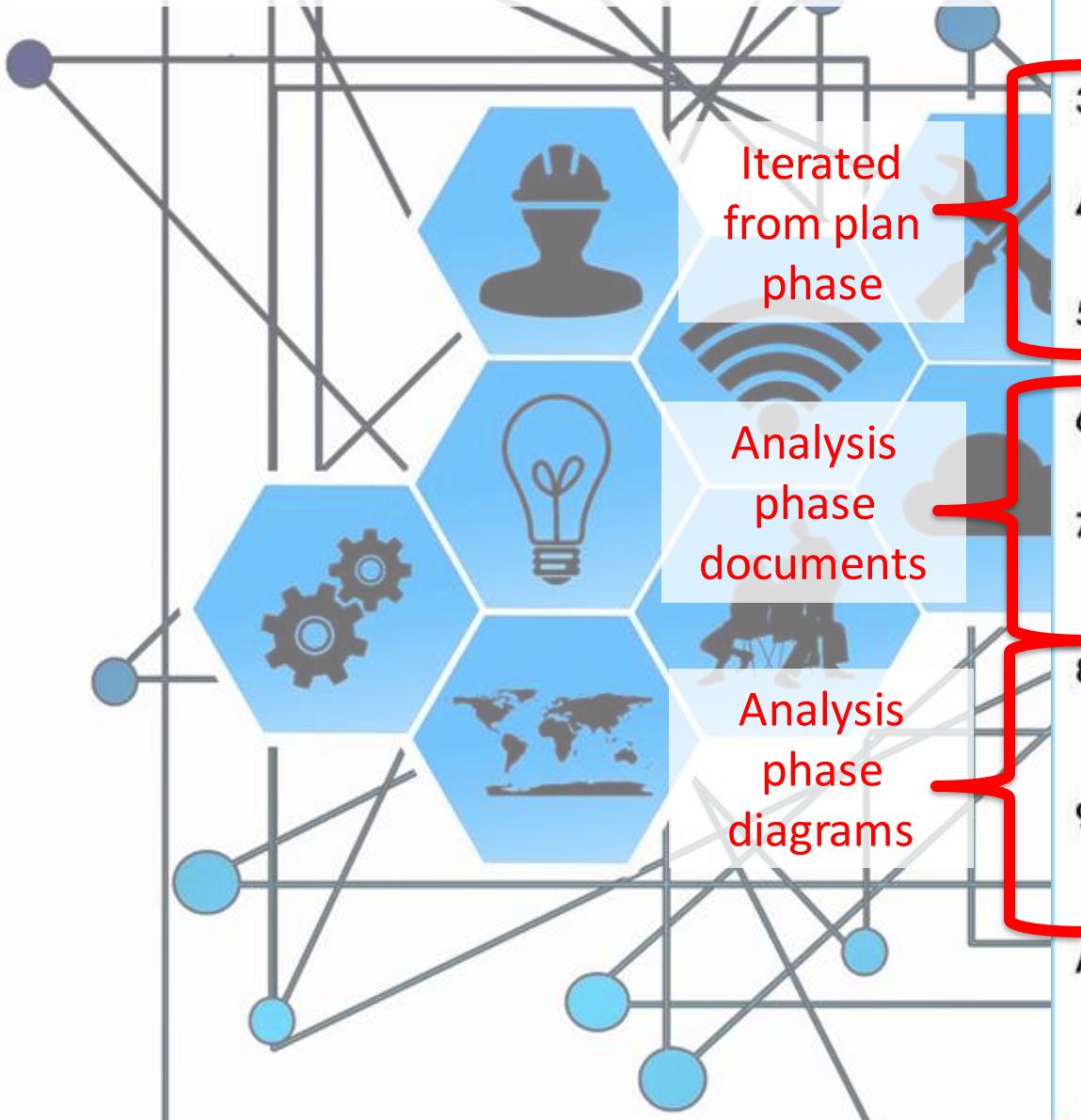
- Program and install automation
- Modify processes
- Ensure change is managed

Project Plan

System Specification

System Specification

Analysis Phase Roadmap



1. Table of Contents

2. Executive Summary

A summary of all the essential information in the proposal so that a busy executive can read it quickly and decide what parts of the plan to read in more depth.

3. System Request

The revised system request form. (See Chapter 1.)

4. Work plan

The original work plan, revised after having completed the analysis phase. (See Chapter 2.)

5. Feasibility Analysis

A revised feasibility analysis, using the information from the analysis phase. (See Chapter 1.)

6. Requirements Definition

A list of the functional and nonfunctional business requirements for the system (this chapter).

7. Use Cases

A set of use cases that illustrate the basic processes that the system needs to support. (See Chapter 4.)

8. Process Model Data Flow Diagrams

A set of process models and descriptions for the to-be system. (See Chapter 5.) This may include process models of the current as-is system that will be replaced.

9. Data Model Entity Relationship Diagrams

A set of data models and descriptions for the to-be system. (See Chapter 6.) This may include data models of the as-is system that will be replaced.

Appendices

These contain additional material relevant to the proposal, often used to support the recommended system. This might include results of a questionnaire survey or interviews, industry reports and statistics, etc.

Data Flow Diagrams



From our text...

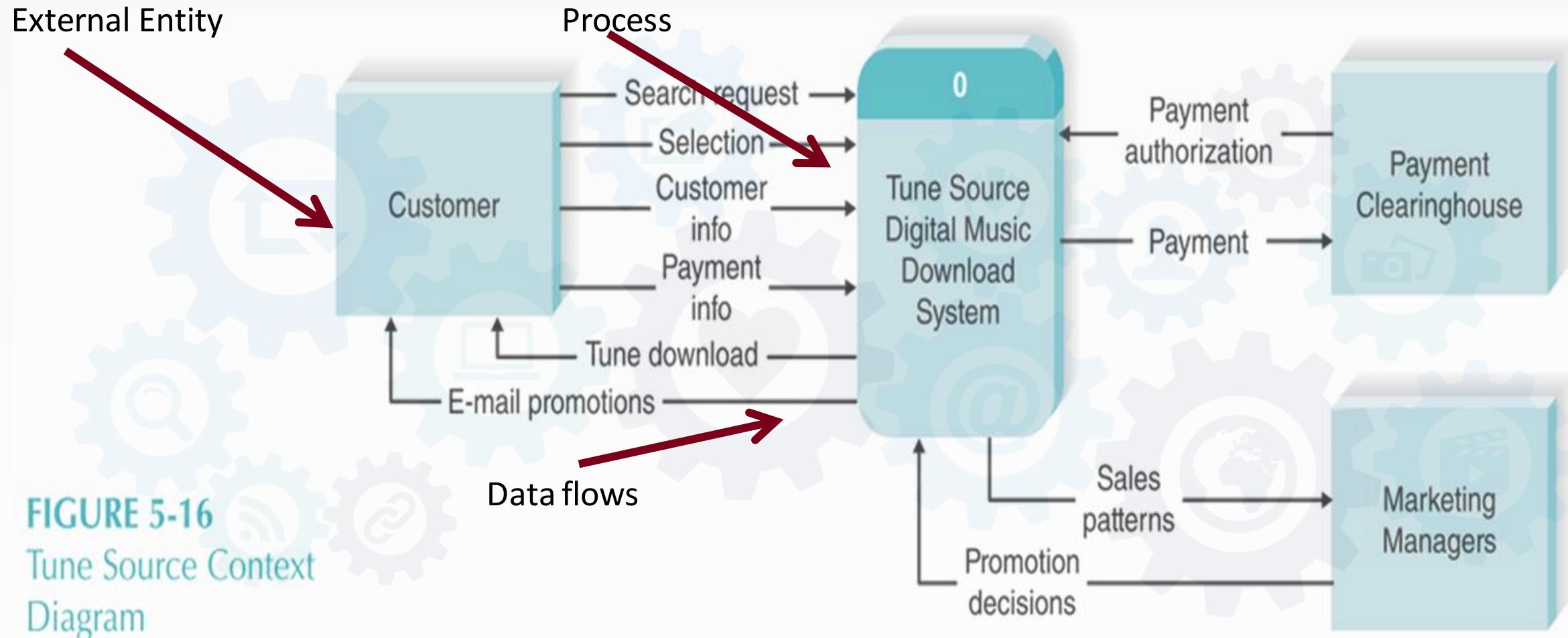


FIGURE 5-16
Tune Source Context
Diagram

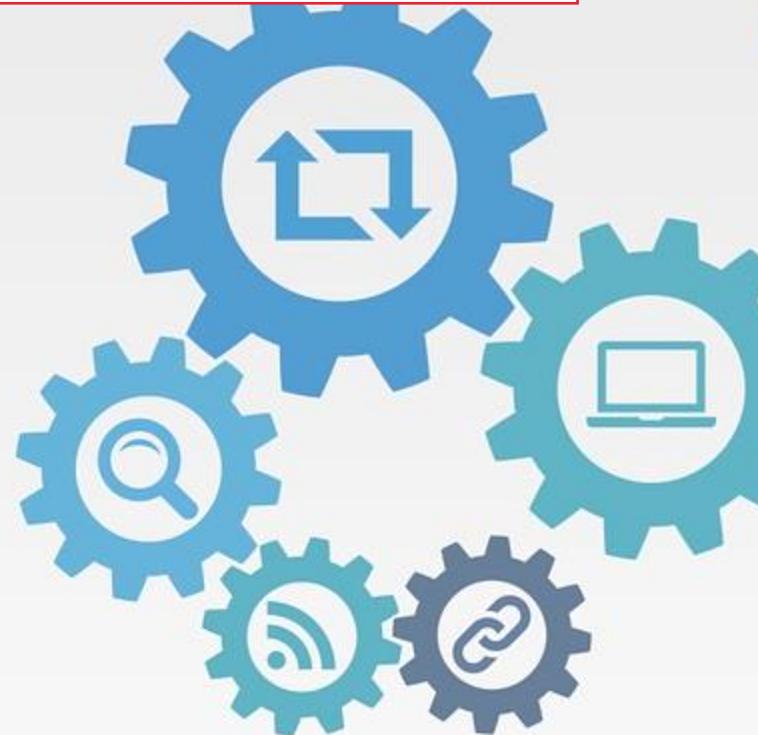
A context diagram is a special subset of a data flow diagram. You can think of it as the top level data flow diagram that shows the relationship of a system to its external entities. Major components in a data flow diagram are:

- A process. Processes in a data flow diagram typically appear in a box with rounded corners. Process name is The Source Digital Music Download and its number is 0.
- External Entities such as people or other systems. They typically appear in a box with sharp corners.
- Data Flow between external entities and processes or in between processes themselves.
- Data Store such as a database

Components

Notation Types:

- 1- Gane and Sarson notation
- 2- DeMarco Yourdon notation

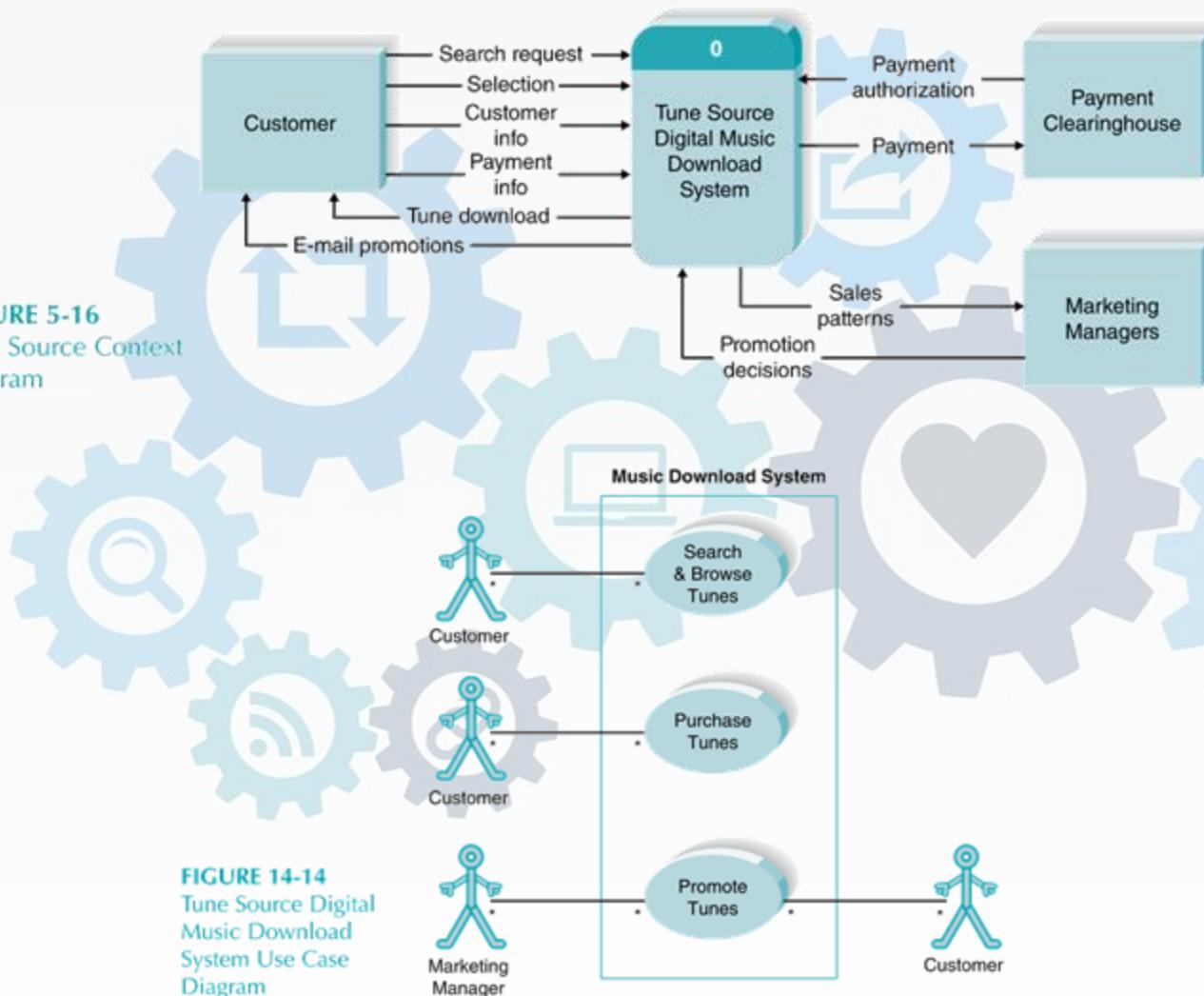


Data stores include computerized databases, down to a filing cabinet or even just a stack of papers on your desk.

Data Flow Diagram Element	Typical Computer-Aided Software Engineering Fields	Gane and Sarson Symbol	DeMarco and Yourdon Symbol
Every <i>process</i> has a number a name (verb phase) a description at least one output data flow at least one input data flow	Label (name) Type (process) Description (what is it) Process number Process description (structured English) Notes		
Every <i>data flow</i> has a name (a noun) a description one or more connections to a process	Label (name) Type (flow) Description Alias (another name) Composition (description of data elements) Notes		
Every <i>data store</i> has a number a name (a noun) a description one or more input data flows one or more output data flows	Label (name) Type (store) Description Alias (another name) Composition (description of data elements) Notes		
Every <i>external entity</i> has a name (a noun) a description	Label (name) Type (entity) Description Alias (another name) Entity description Notes		

Use Case Diagram vs. DFD

FIGURE 5-16
Tune Source Context
Diagram



Usually identical

- Actor & external entity
- Title

Similar

- Process names
 - Focus on user's functional goal (Use case diagram)
 - Focus on corporation (DFD)

Different

- Labeled data flows (Context diagram)
- Next level of detail (Use case diagram)

Use Case Diagram vs. DFD

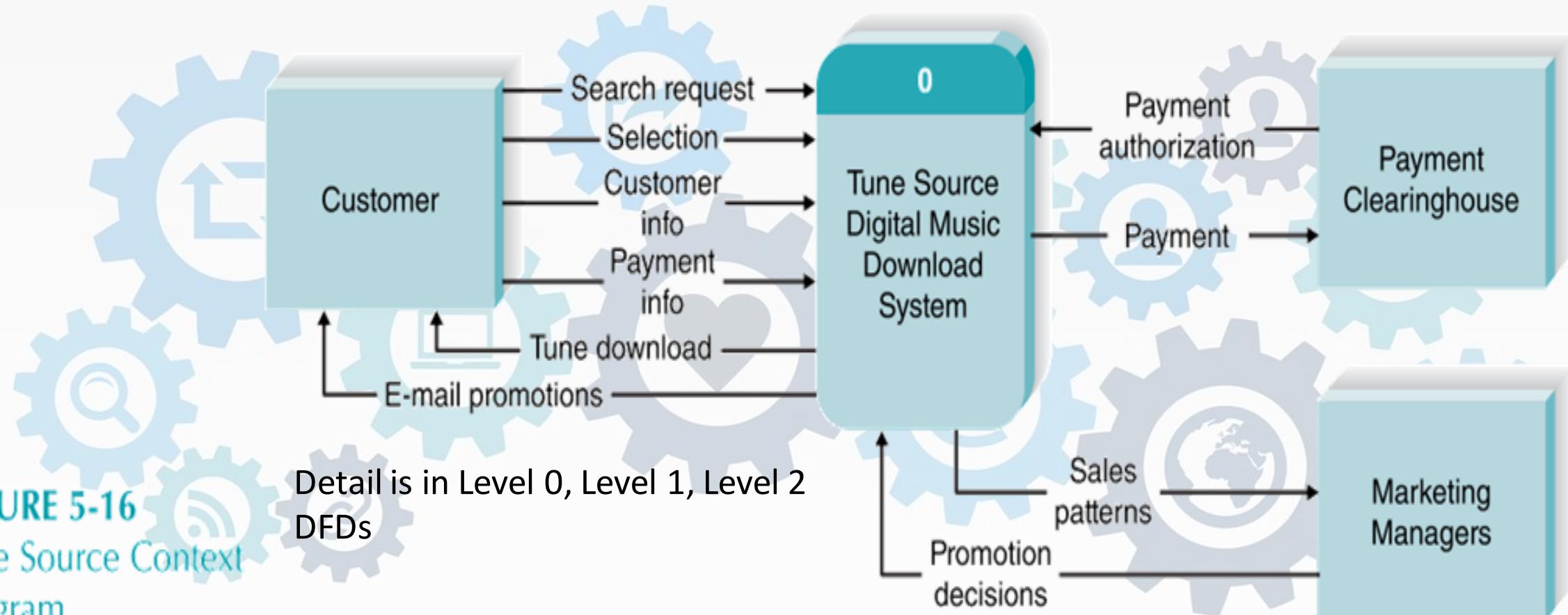


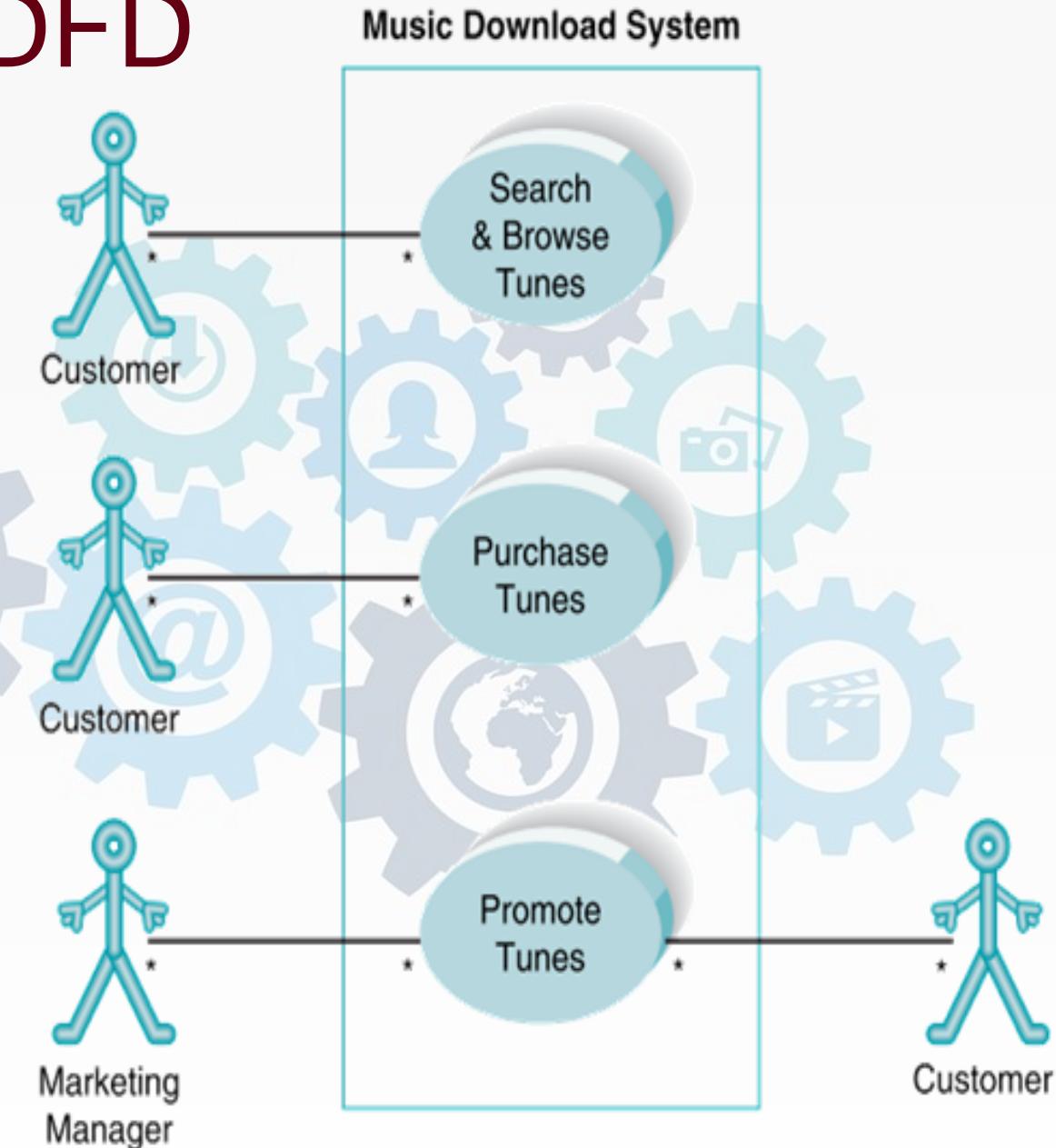
FIGURE 5-16
Tune Source Context
Diagram

Use Case Diagram vs. DFD



Detail in use cases
and steps

FIGURE 14-14
Tune Source Digital
Music Download
System Use Case
Diagram





CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Data Flow Diagrams: Levels



Learning Objectives

After this lesson, you will be able to:

- Describe the purpose of higher level DFDs.
- Evaluate whether a DFD is balanced.
- Decompose a system into processes.

Context Diagram → Level 0 DFD

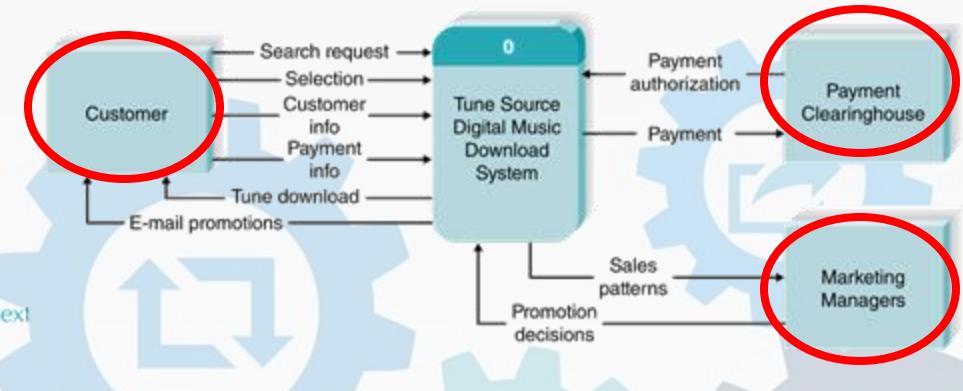


FIGURE 5-16
Tune Source Context
Diagram

The context diagram can be decomposed into what's called a level zero data flow diagram.
 The context diagram shows the overall system with its relationship to the external entities.
 The level zero data flow diagram shows the major processes that make up the system.

Steps

- 1- Take the external entities from our context diagram and effectively replicate them on our level zero data flow diagram.
- 2- Decompose the system from the context diagram into a set of constituent processes. These are called the level zero processes.

The level zero processes in our DFD roughly correspond to the use cases from our use case diagram.

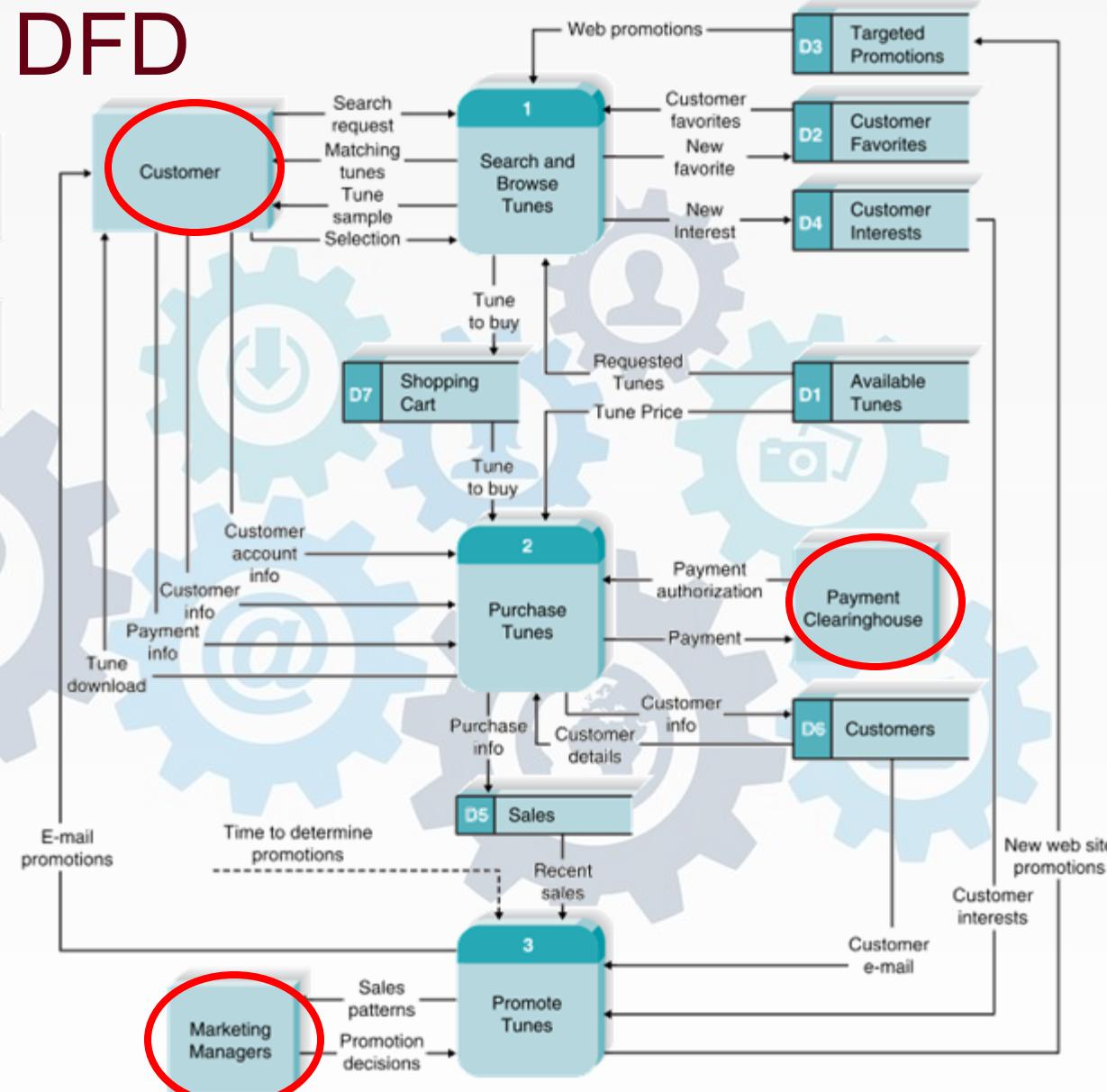
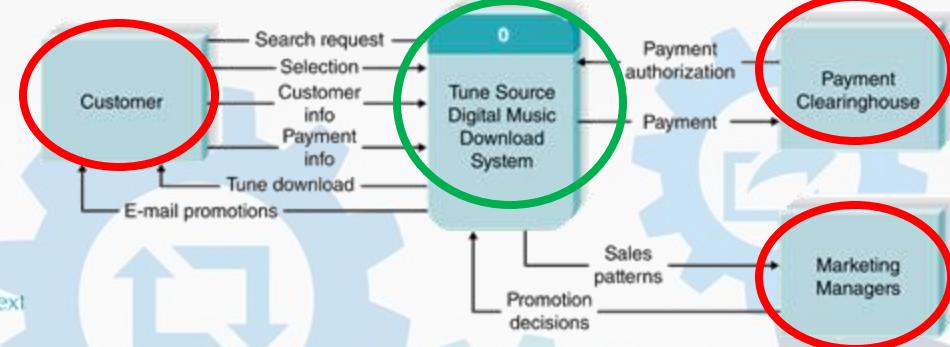


FIGURE 5-18 Tune Source Level 0 DFD

Context Diagram → Level 0 DFD

FIGURE 5-16
Tune Source Context Diagram



Steps

- 1- Take the external entities from our context diagram and effectively replicate them on our level zero data flow diagram.
- 2- Decompose the system from the context diagram into a set of constituent processes. These are called the level zero processes.

The level zero processes in our DFD roughly correspond to the use cases from our use case diagram.

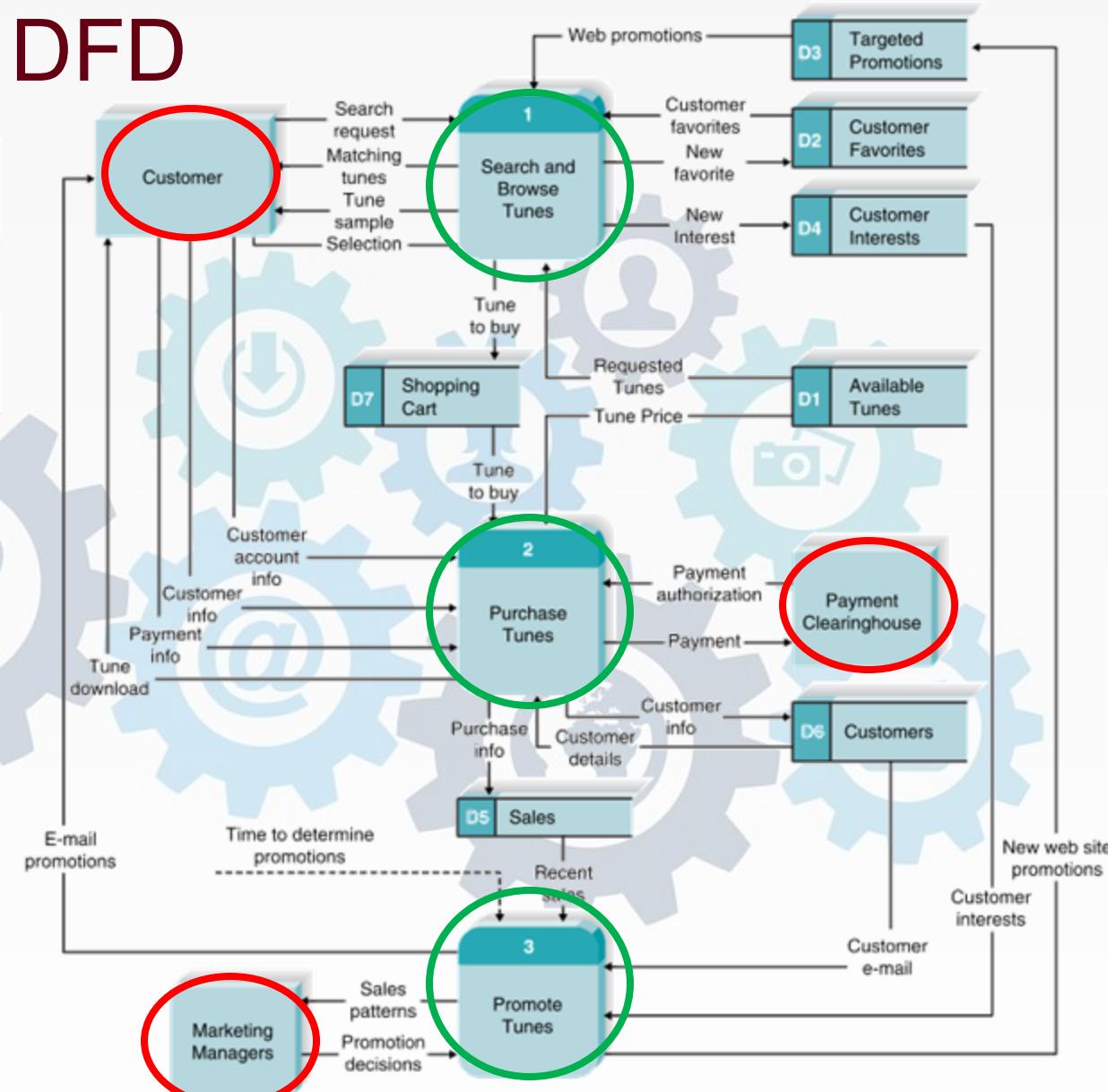
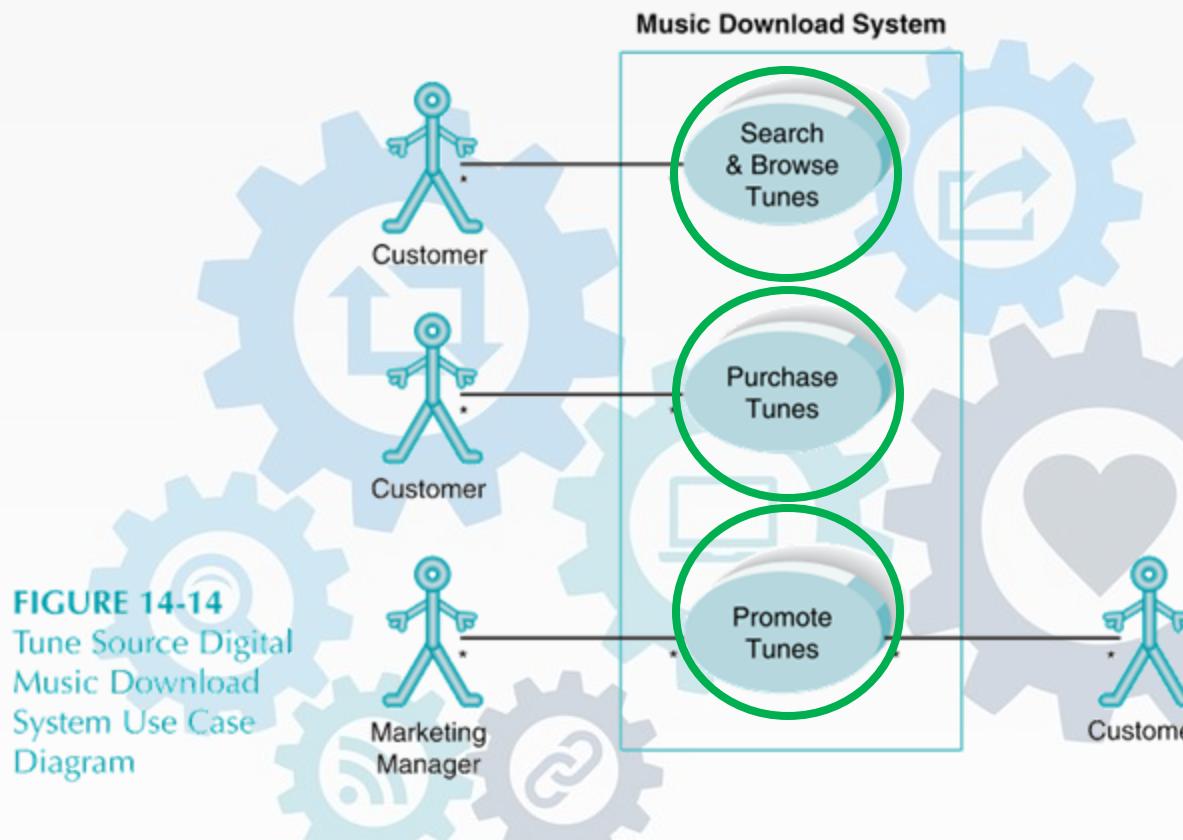


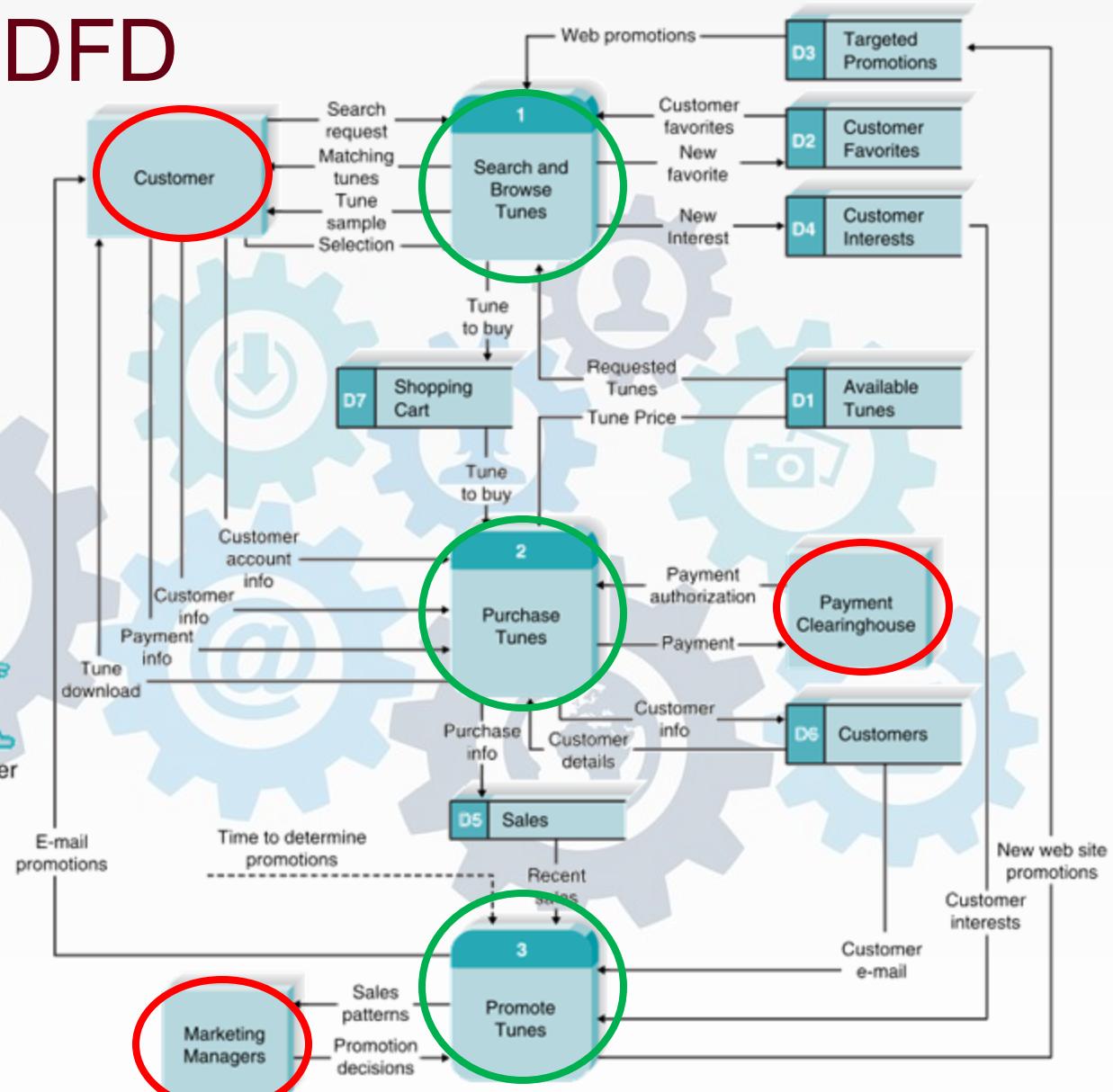
FIGURE 5-18 Tune Source Level 0 DFD

Context Diagram → Level 0 DFD



The level zero processes in our DFD roughly correspond to the use cases from our use case diagram.

Go to the use case diagram, pull the use cases, make those your level zero processes.



Context Diagram → Level 0 DFD

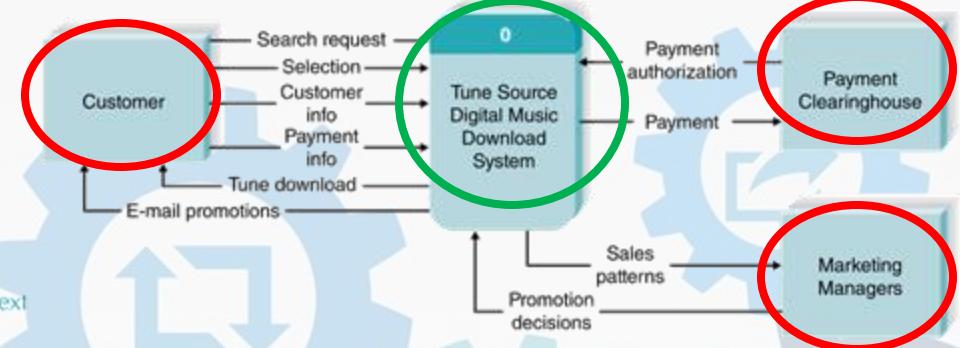


FIGURE 5-16
Tune Source Context
Diagram

Include new and additional information.
In particular, we include these data stores.

So, these data stores might end up being databases on a computer, or they might be like a set of files in a filing cabinet, or as simple as a stack of papers on your desk.

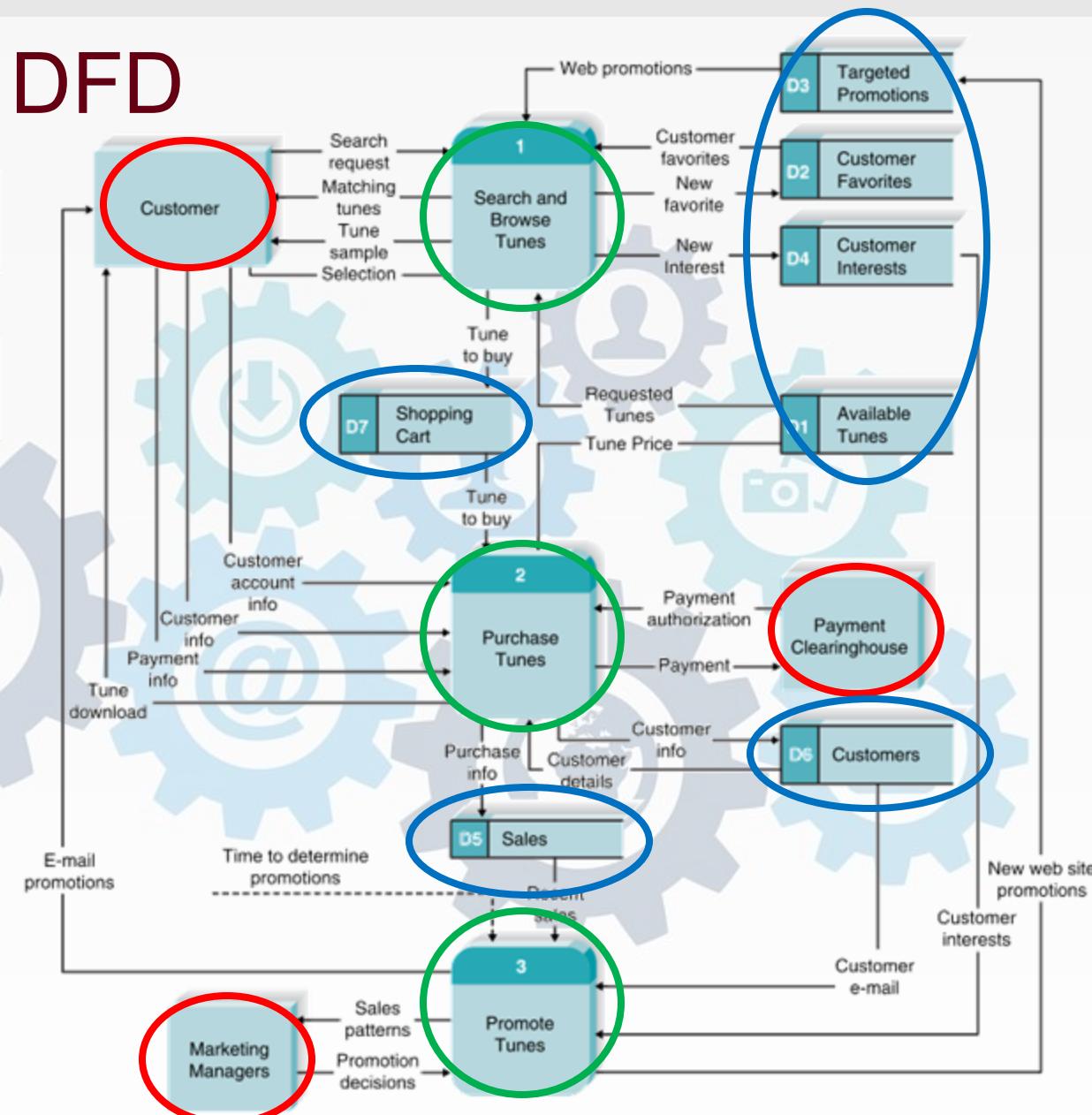
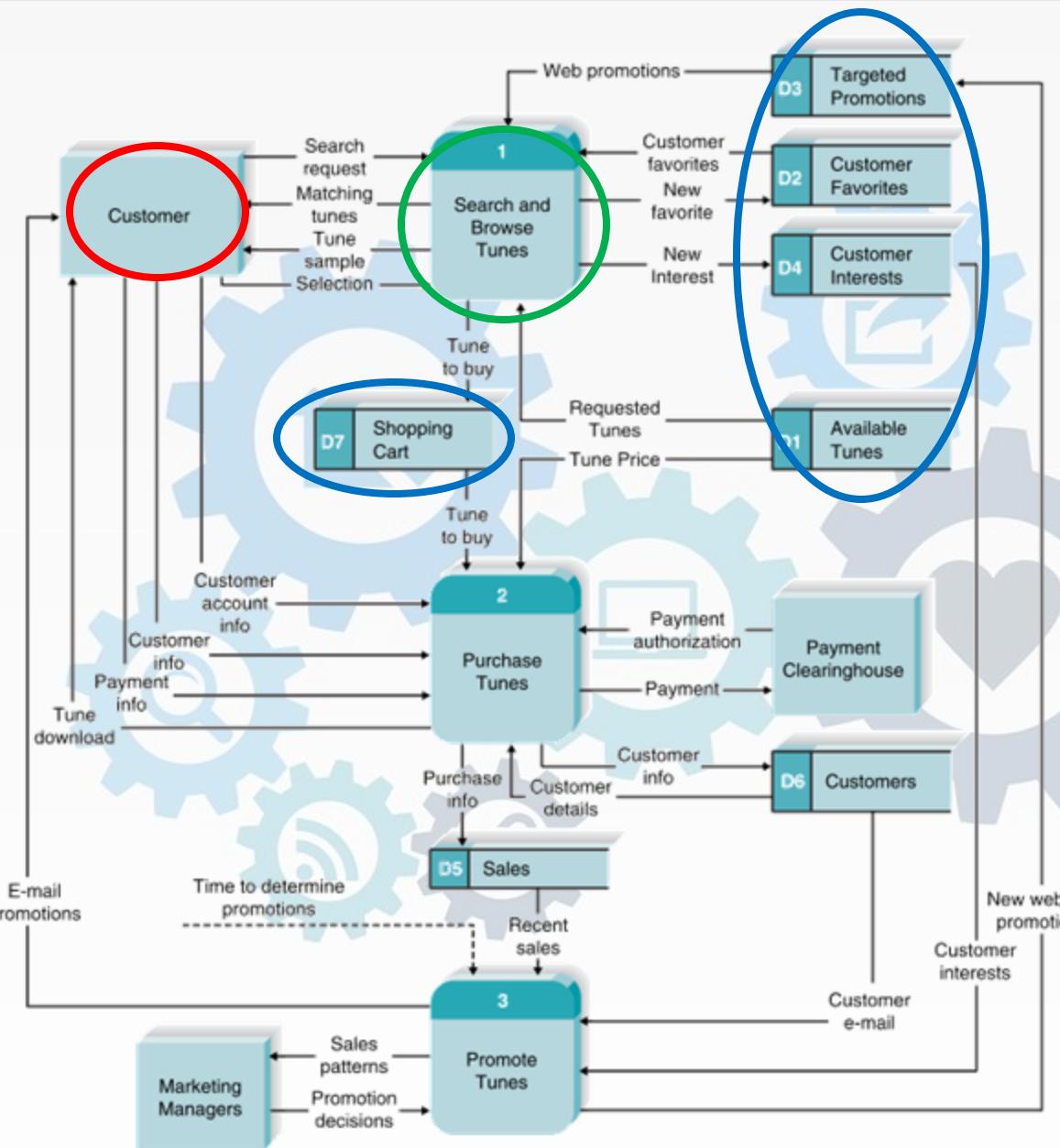


FIGURE 5-18 Tune Source Level 0 DFD



DFDs need to be *balanced*

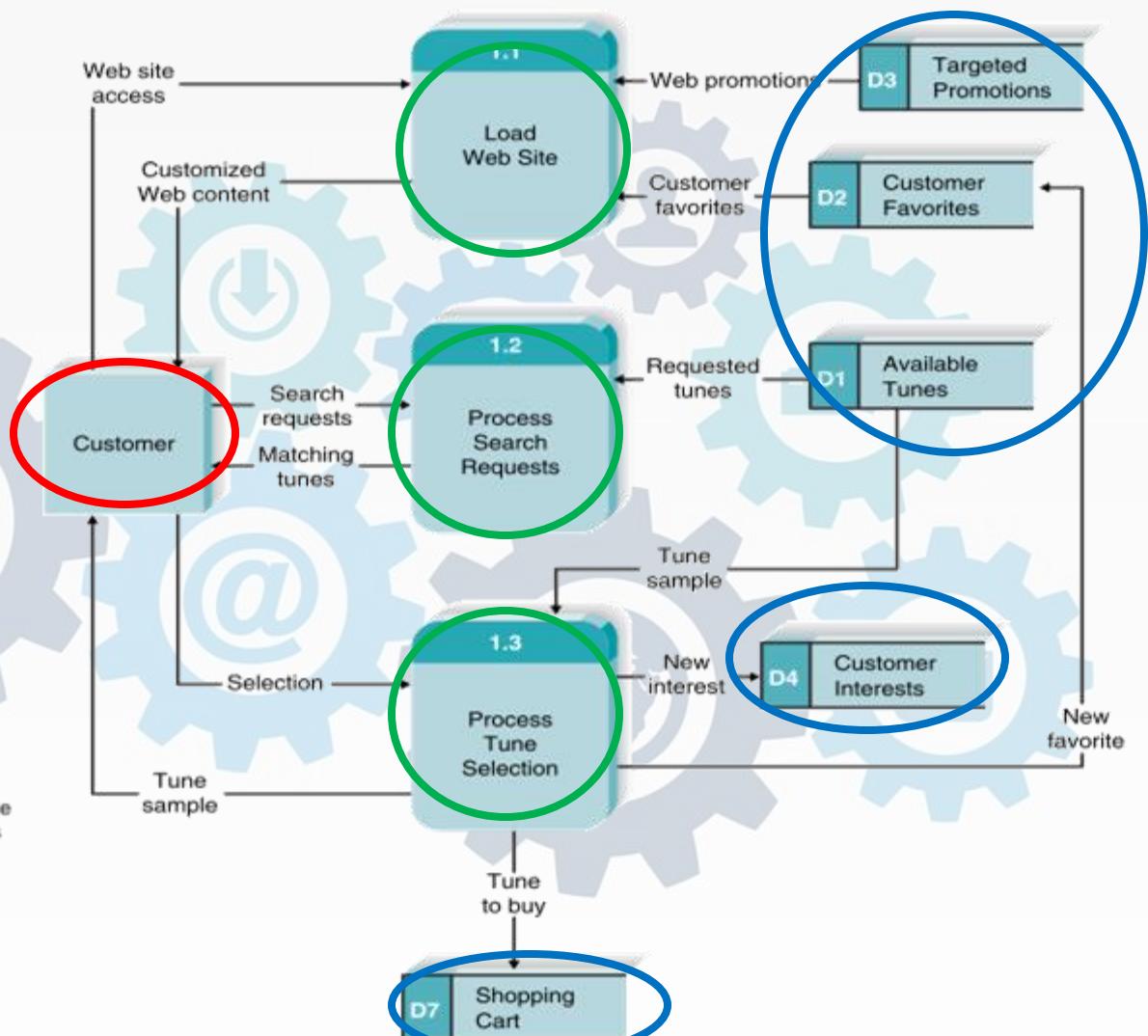
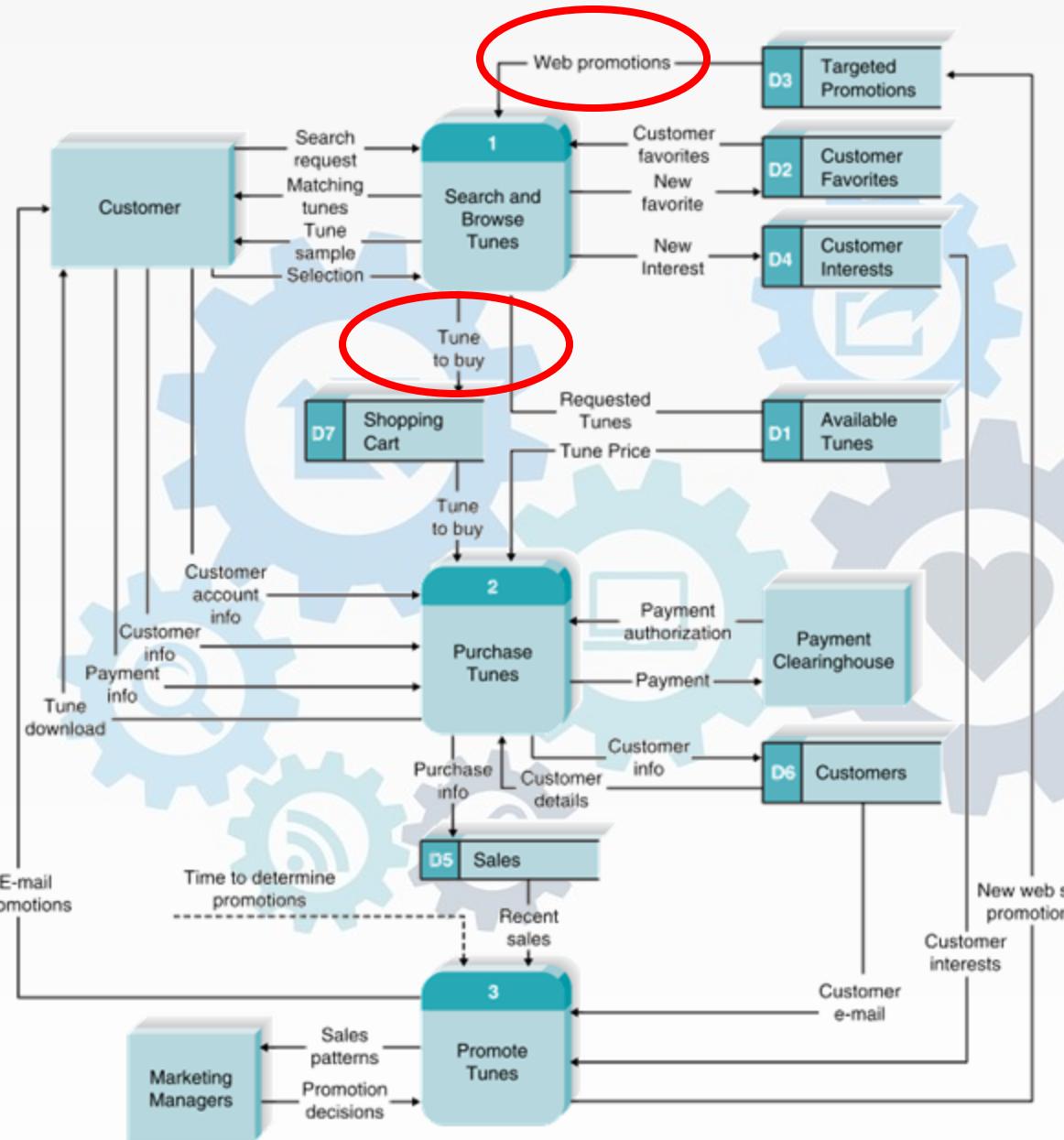


FIGURE 5-19 Level 1 DFD for Tune Source Process 1: Search and Browse Tunes

FIGURE 5-18 Tune Source Level 0 DFD

Decomposes the level zero DFD, into a level one DFD. Decompose Process 1 into something called a level one DFD that include 1.1, 1.2, 1.3 processes



DFDs need to be *balanced*

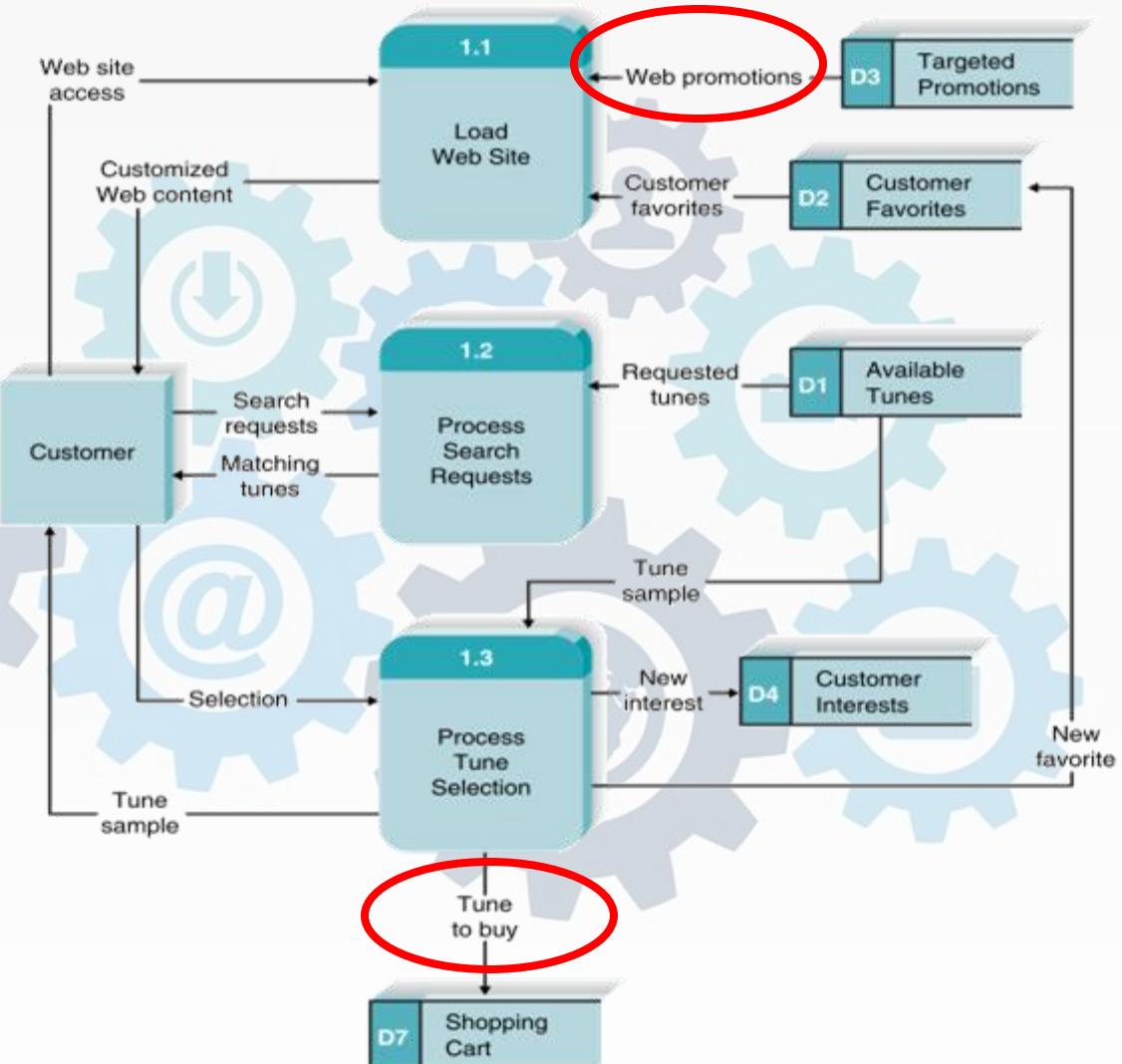
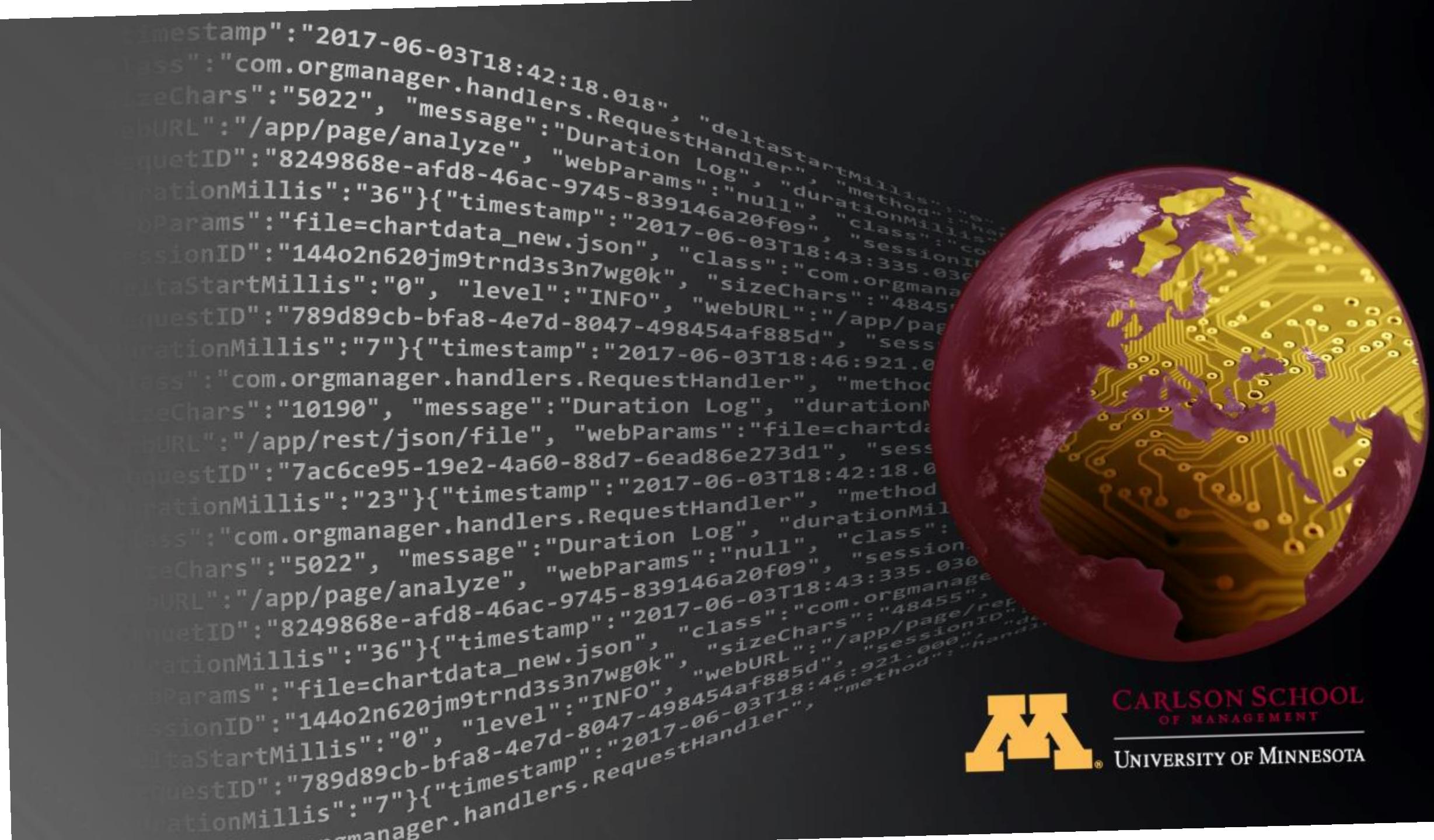


FIGURE 5-19 Level 1 DFD for Tune Source Process 1: Search and Browse Tunes

FIGURE 5-18 Tune Source Level 0 DFD



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Data Flow Diagram: Types



Learning Objectives

After this lesson, you will be able to:

- Identify errors in a data flow diagram.
- Name the notation used for a data flow diagram.
- List and describe the four different types of data flow diagrams.

DFD Guidelines

Syntax

Within DFD

Process

Miracle

Black hole

- Every process has a unique name that is an action-oriented verb phrase, a number, and a description.
- Every process has at least one input data flow.
- Every process has at least one output data flow.
- Output data flows usually have different names than input data flows because the process changes the input into a different output in some way.
- There are between three and seven processes per DFD.

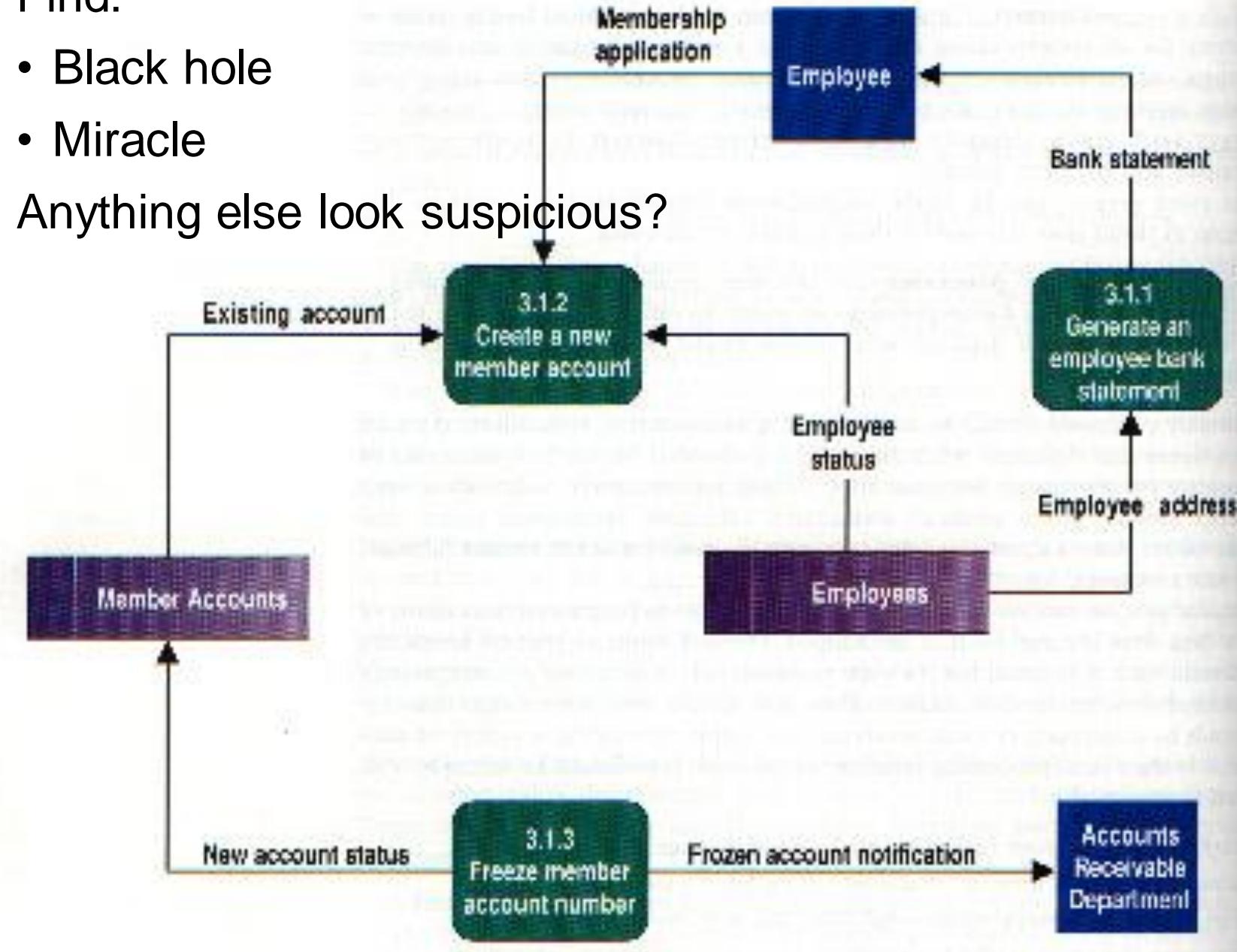


Find the errors

Find:

- Black hole
- Miracle

Anything else look suspicious?



Process 3.1.2, create a new bank account, you can see that there are three data flows coming in, but no data flows going out. This is our black hole. If we look just below that, down here at process 3.1.3, you can see, freeze member account number has information coming out of it, but no information coming in. This is something that we would call a miracle. If we look far on the right of the diagram, over at process 3.1.1, it says generate an employee bank statement. Well, this isn't necessarily a miracle or a black hole because it has data flowing in and data flowing out. It does look suspicious because can we really generate an employee bank statement with only the address as input? This is what I would call a semantic error.

DFD Guidelines

Guideline only:
Processes usually
change data

Data Flow

- Every data flow has a unique name that is a noun, and a description.
- Every data flow connects to at least one process.
- Data flows only in one direction (no two-headed arrows).
- A minimum number of data flow lines cross.

Data Store

- Every data store has a unique name that is a noun, and a description.

• Every data store has at least one input data flow (which means to add new data or change existing data in the data store) on some page of the process model.

• Every data store has at least one output data flow (which means to read data from the data store) on some page of the process model.

No ↔

External Entity

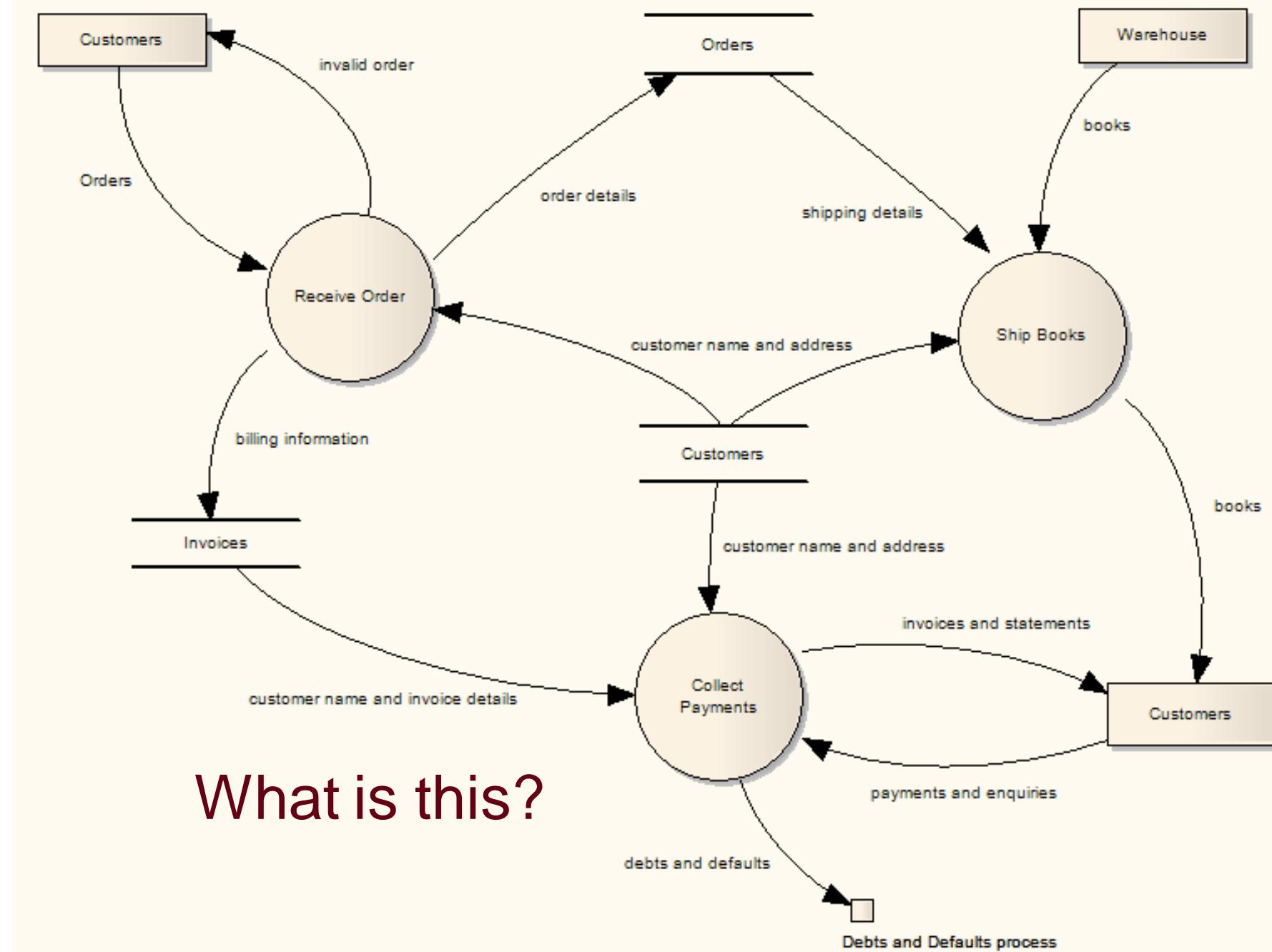
- Every external entity has a unique name that is a noun, and a description.
- Every external entity has at least one input or output data flow.

Guidelines Only

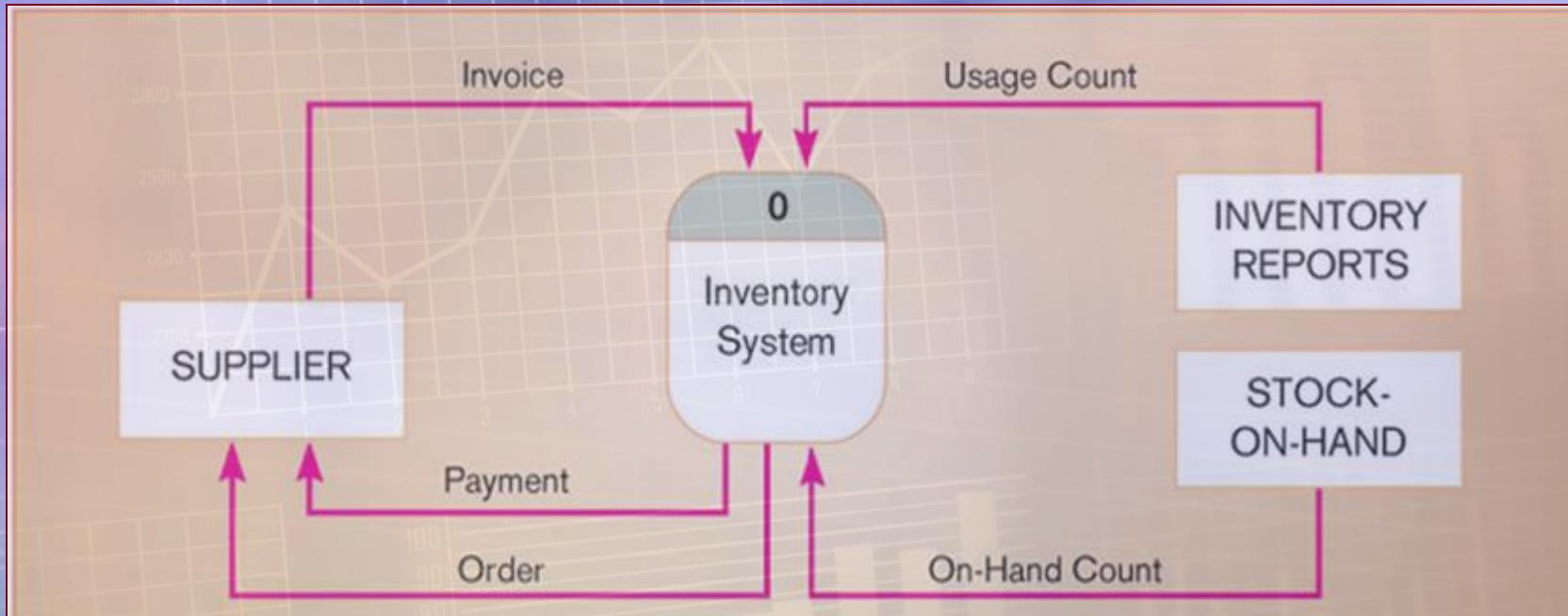


No EE to EE or
DS to DS flows

DeMarco/Yourdon Notation:
For the most part, the meaning in the semantics is the same. The shapes are just slightly different.



4 Types of DFDs



As-Is Process

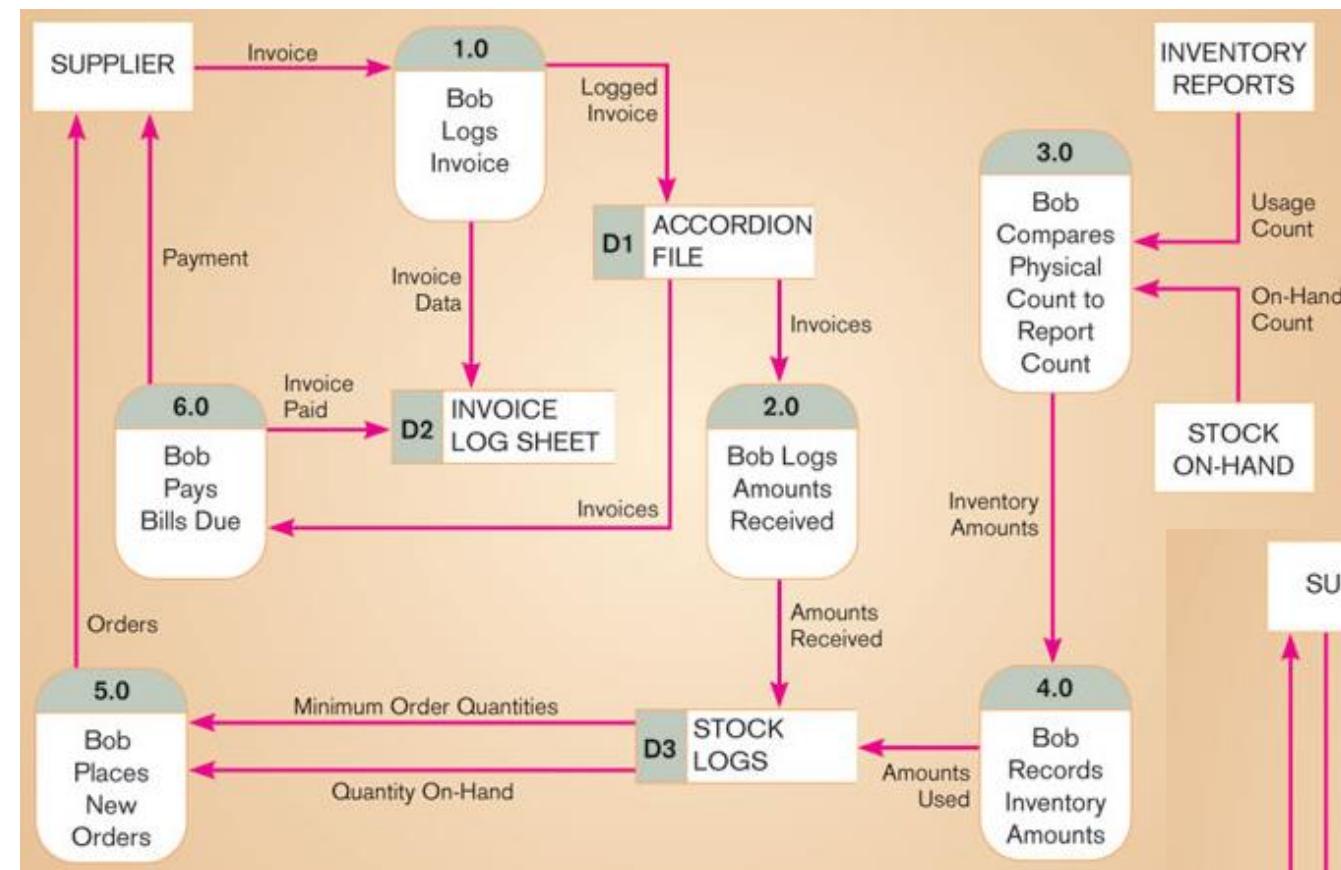
Our current inventory control process is largely manual. At the end of each day, the inventory analyst obtains:

- A **usage count** that tells him how much inventory should have been used for each item associated with a sale.
- The results of an inventory count (**on-hand count**)

The analyst then can calculate **how much inventory needs to be ordered** and pay invoices.

Logical vs. Physical

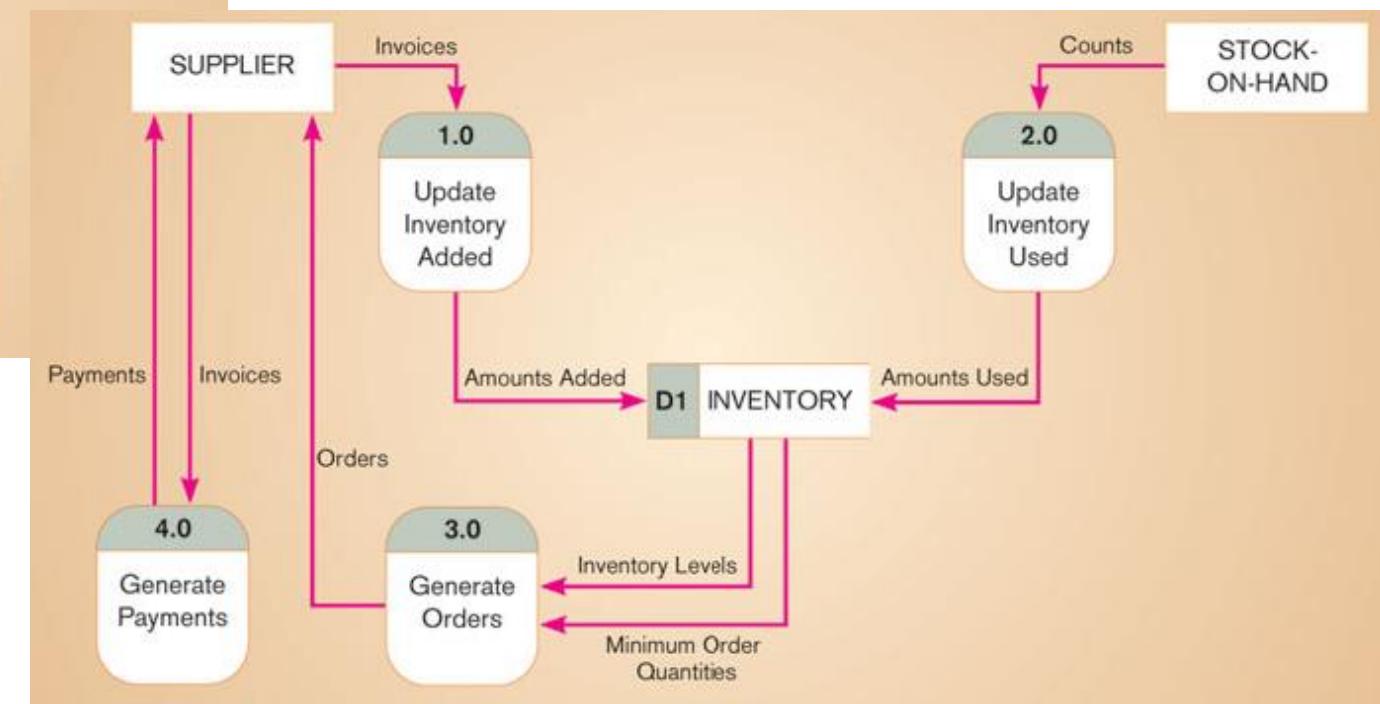
What level diagrams?
Why are there two?



A physical data flow diagram. This diagram often has specific names as you can see here, as well as specific locations where we might store particular information.

The best approach is to:

- 1- Start by drawing a physical as-is data flow diagram.
- 2- Transform the physical as-is DFD into a logical DFD that represents the current process.
- 3- Transform the logical DFD into a logical DFD that represents the to-be process.
- 4- Transform the logical to-be DFD into a physical to-be DFD.



A logical data flow diagram. It does not make any reference to specific physical implementations of some of the processes and the data stores.



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Decomposing DFDs



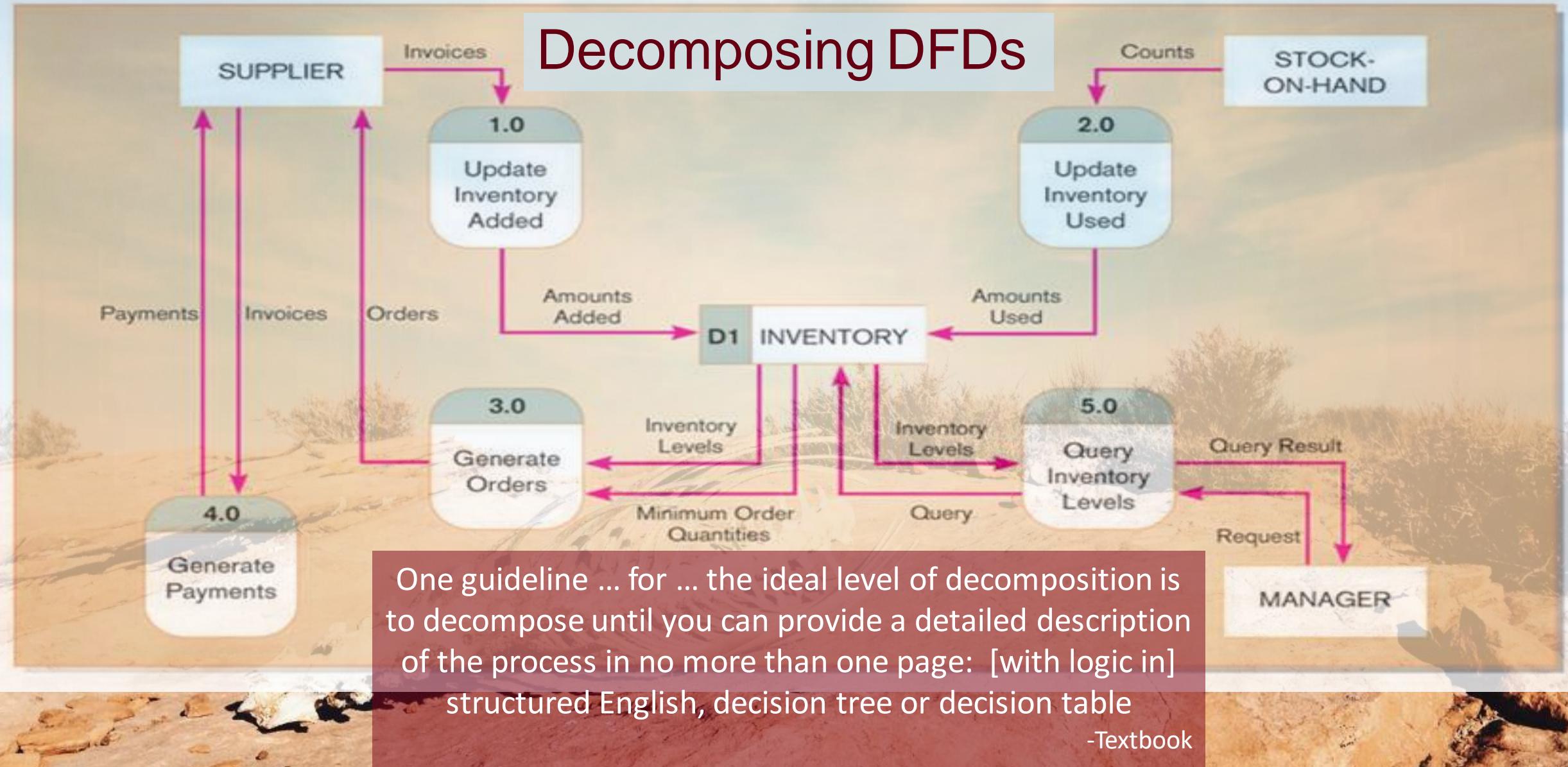
Learning Objectives



After this lesson, you will be able to:

- Describe why we decompose DFDs
- Evaluate whether a DFD is decomposed to the proper level
- Read and interpret primitives written in structured English or a decision table

Figure 7-16 Level-0 data flow diagram for Hoosier Burger's new logical inventory control system



Generate Orders as Structured English

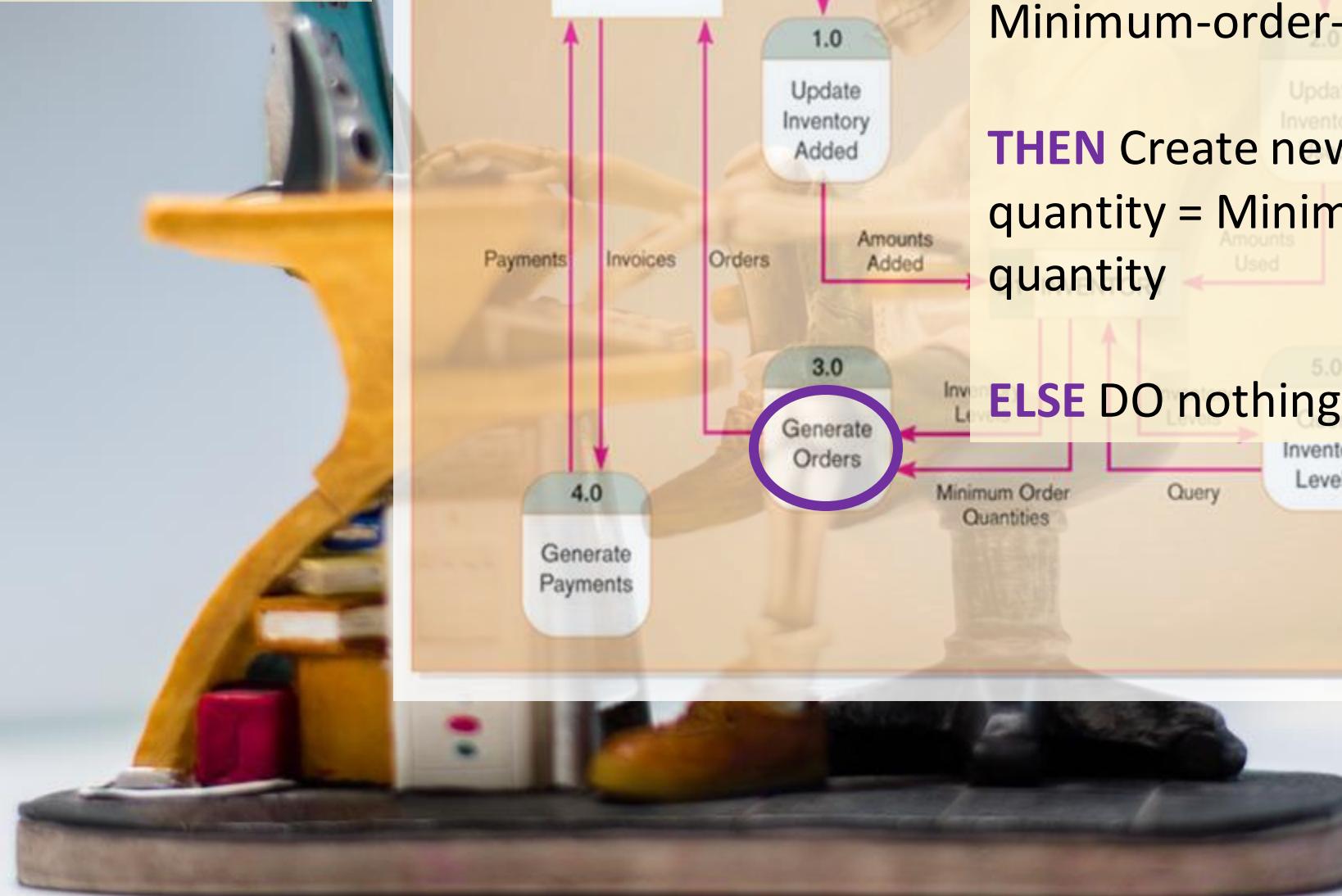


Figure 7-16 Level-0 data flow diagram for Hoosier Burger's new logical inventory control system

IF Inventory-level is less than Minimum-order-quantity

THEN Create new order with quantity = Minimum-order-quantity

ELSE DO nothing

Generate Orders as Structured English

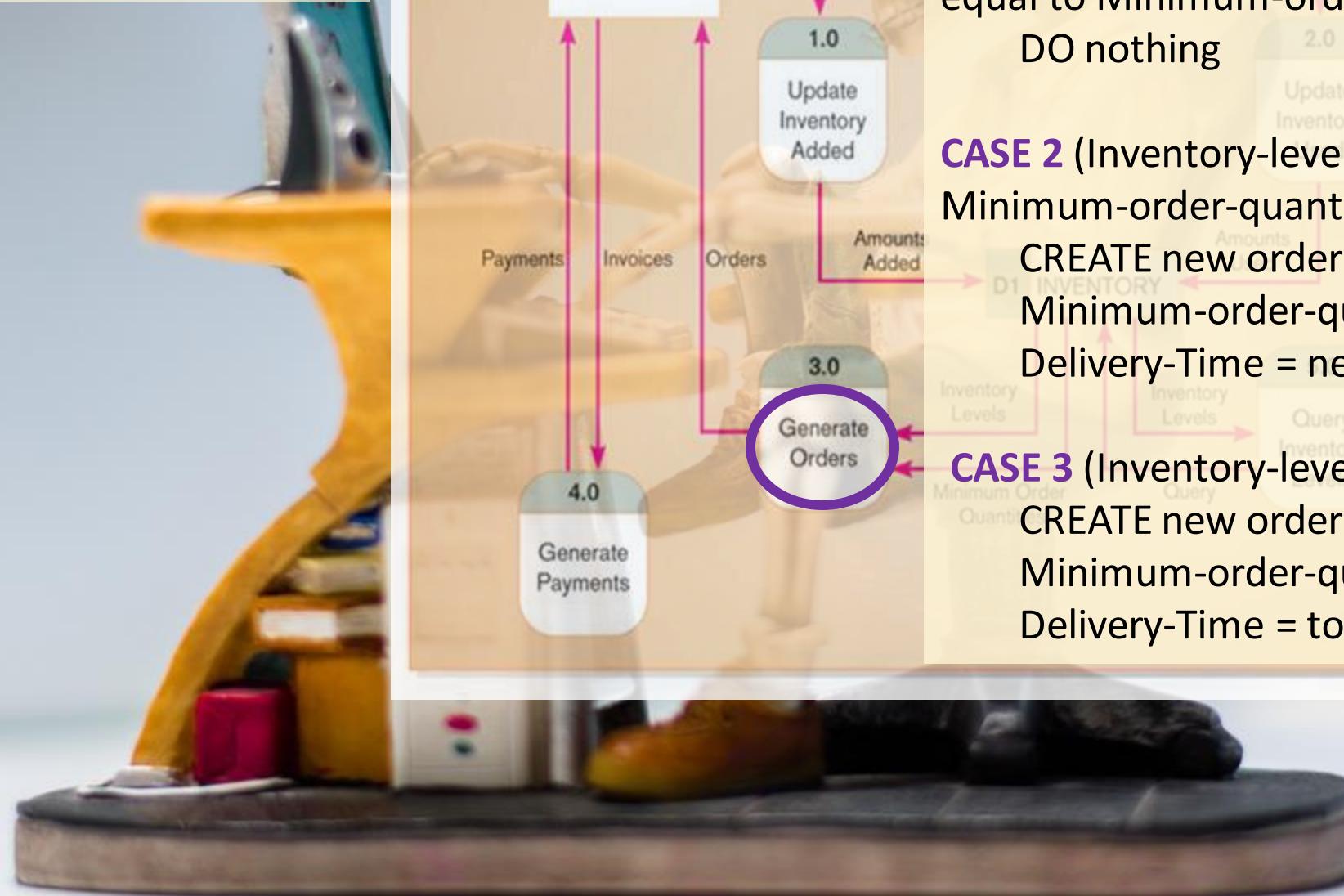


Figure 7-16 Level-0 data flow diagram for Hoosier Burger's new logical inventory control system

CASE 1 (Inventory-level is greater than or equal to Minimum-order-quantity)
DO nothing

CASE 2 (Inventory-level is less than Minimum-order-quantity and not equal 0)
CREATE new order with quantity = Minimum-order-quantity AND
Delivery-Time = next business day

CASE 3 (Inventory-level equals 0)
CREATE new order with quantity = Minimum-order-quantity AND
Delivery-Time = today

Generate Orders as a Decision Table

Conditions/ Courses of Action	Rules						
	1	2	3	4	5	6	7
Type of item	P	P	P	P	P	P	N
Time of week	D	W	D	W	D	W	
Season of year	A	A	S	S	H	H	

P = Perishable

N = Nonperishable

D = Weekday

W = Weekend

A = Academic year

S = Summer

H = Holiday



Generate Orders as a Decision Table

Conditions/ Courses of Action	Rules						
	1	2	3	4	5	6	7
Type of item	P	P	P	P	P	P	N
Time of week	D	W	D	W	D	W	-
Season of year	A	A	S	S	H	H	-
Standing daily order	X		X		X		
Standing weekend order		X		X		X	
Minimum order quantity	P = Perishable N = Nonperishable						
Holiday reduction					X	X	
Summer reduction			X	X			

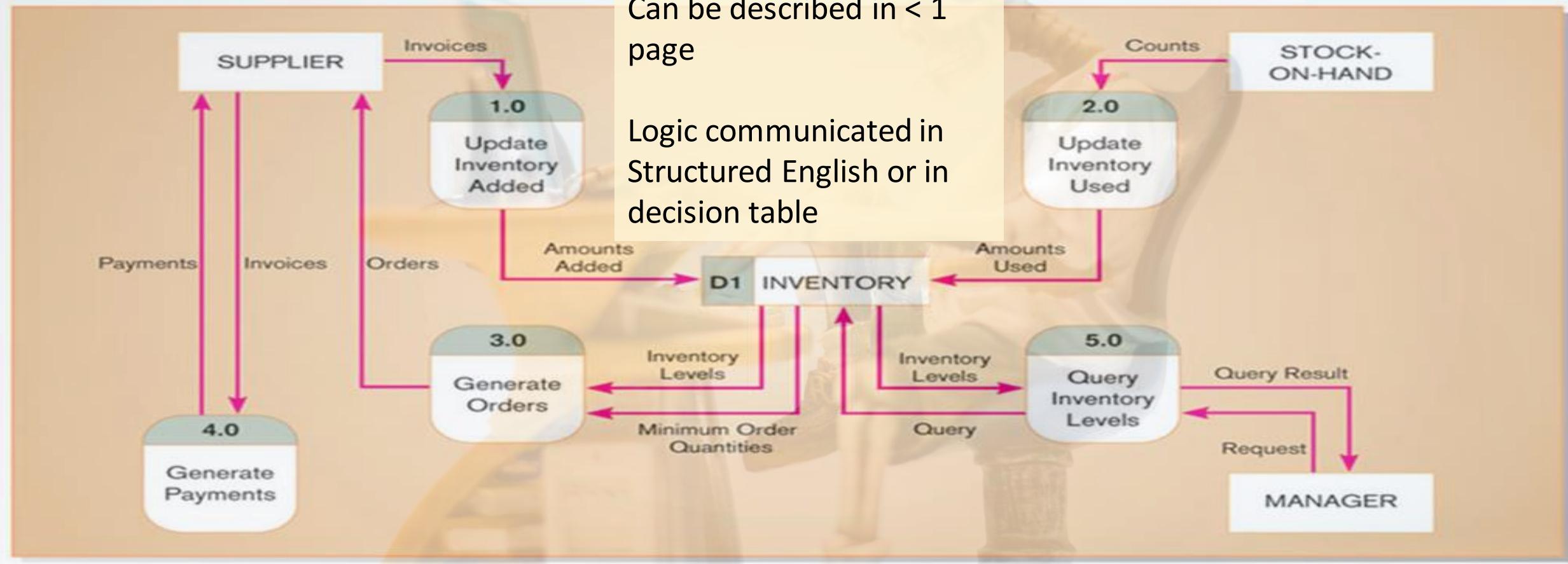


P = Perishable
 N = Nonperishable
 D = Weekday
 W = Weekend
 A = Academic year
 S = Summer
 H = Holiday



DFD Decomposition

Figure 7-16 Level-0 data flow diagram for Hoosier Burger's new logical inventory control system



Primitive (lowest level)

Can be described in < 1 page

Logic communicated in Structured English or in decision table

A primitive is essentially the lowest level of decomposition of a process within a data flow diagram. It should be something that we can describe in less than one page, and the logic should be able to be communicated in Structured English or in a Decision Table.



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Business Process Model and Notation



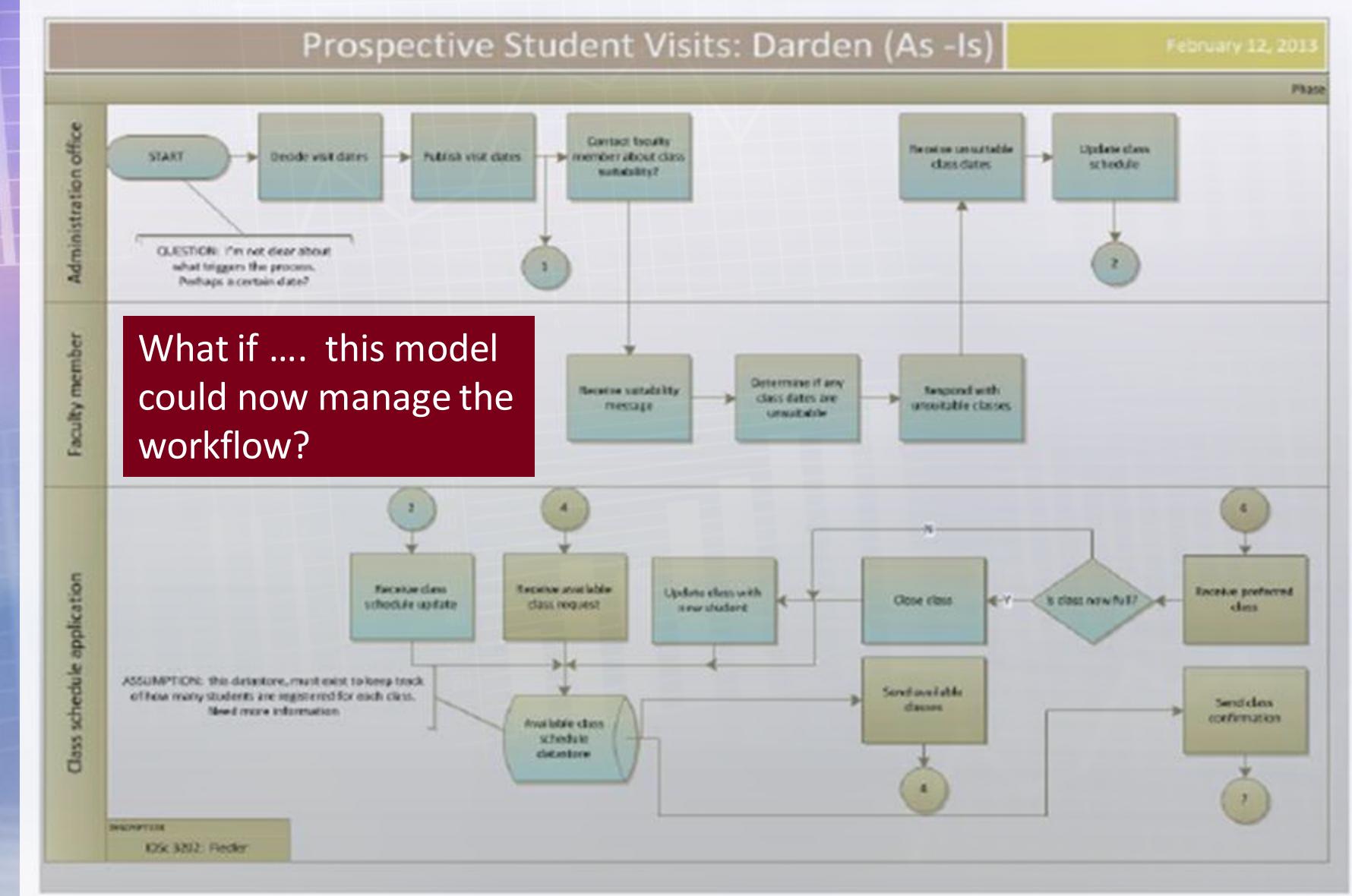
Learning Objectives



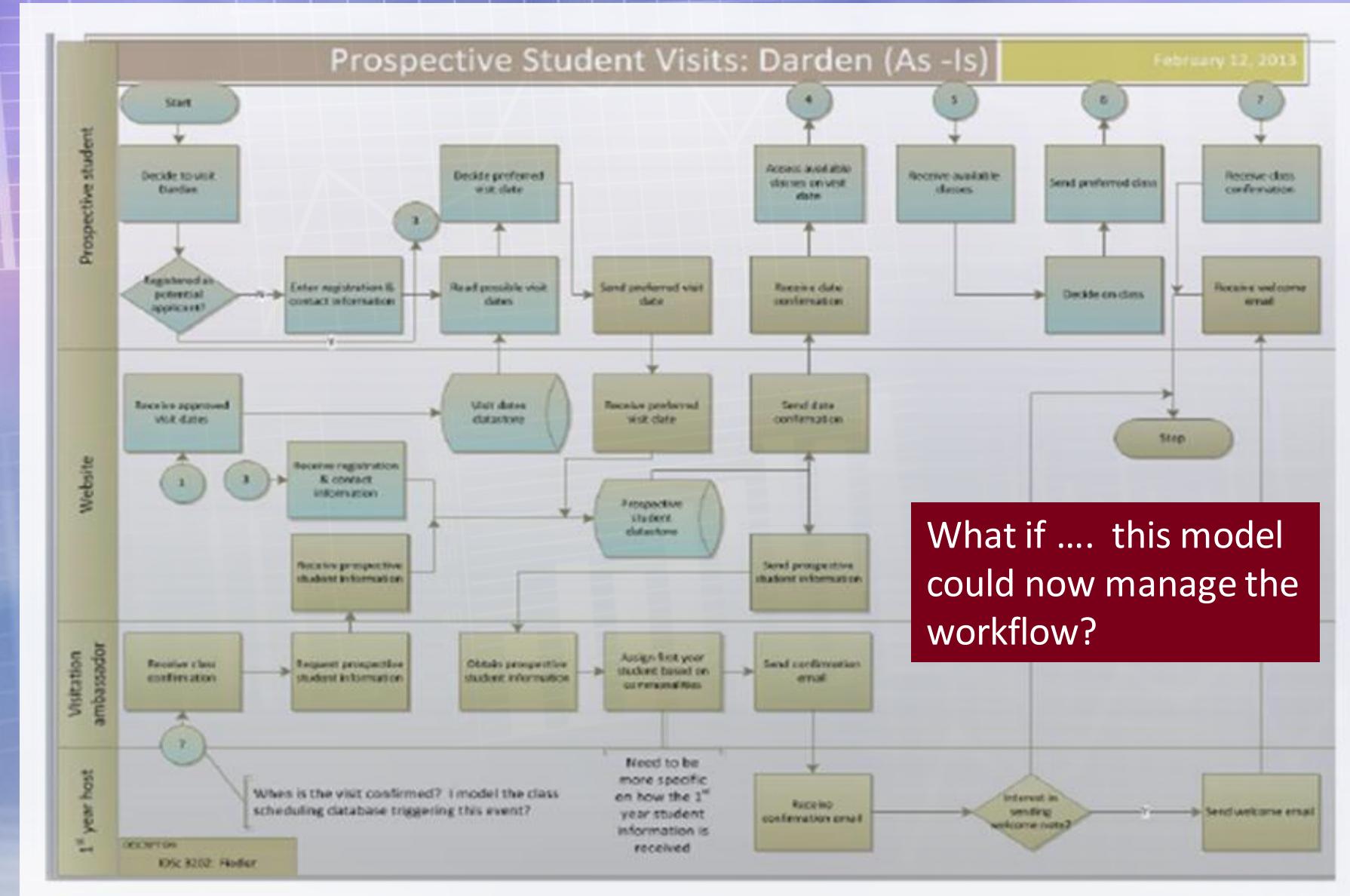
After this lesson, you will be able to:

- Define Business Process Model and Notation (BPMN)
- Describe the vision for BPMN
- Describe situations in which BPMN might be useful, and differentiate BPMN from other diagram types

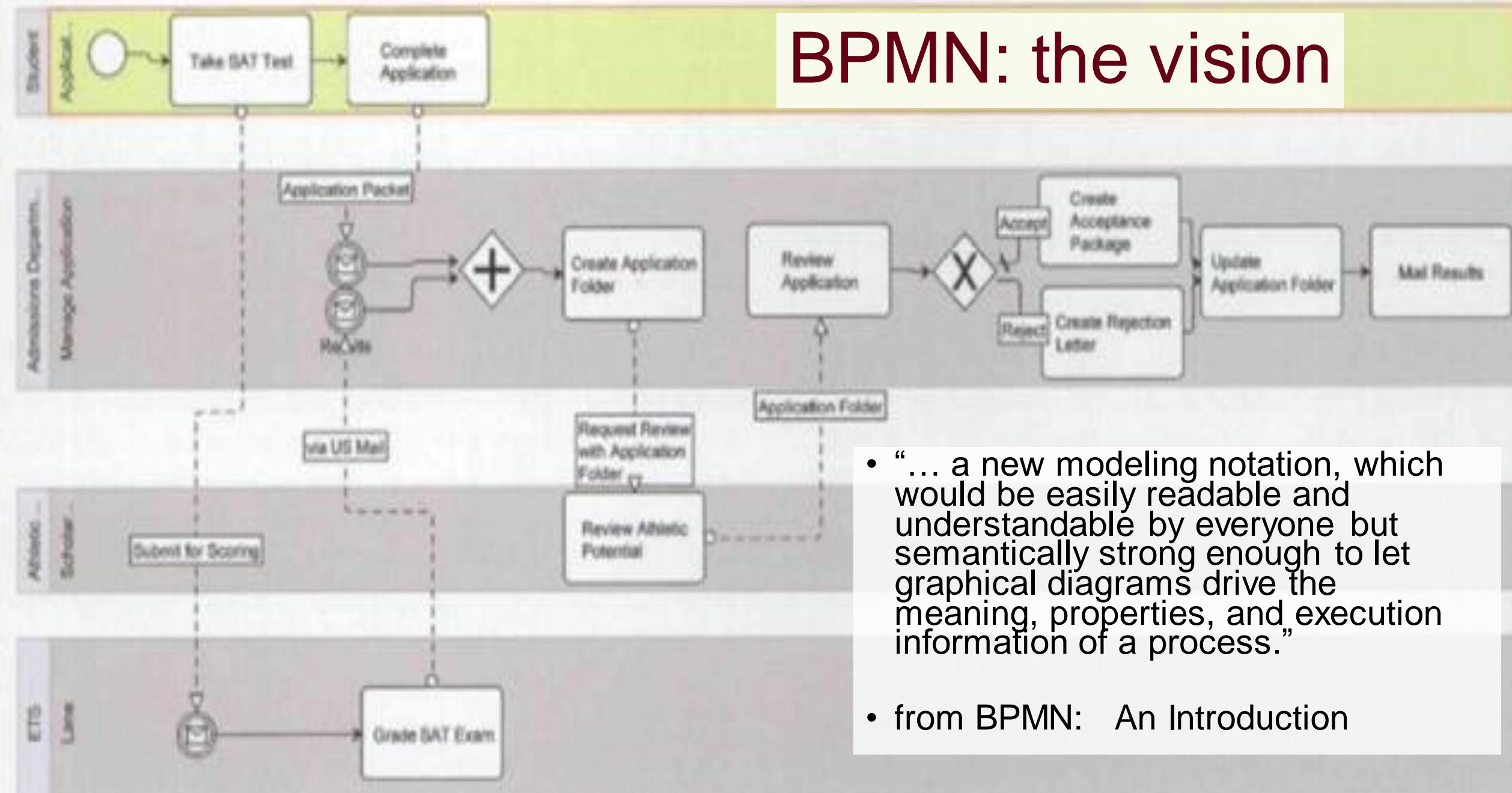
Process Modeling: The Vision



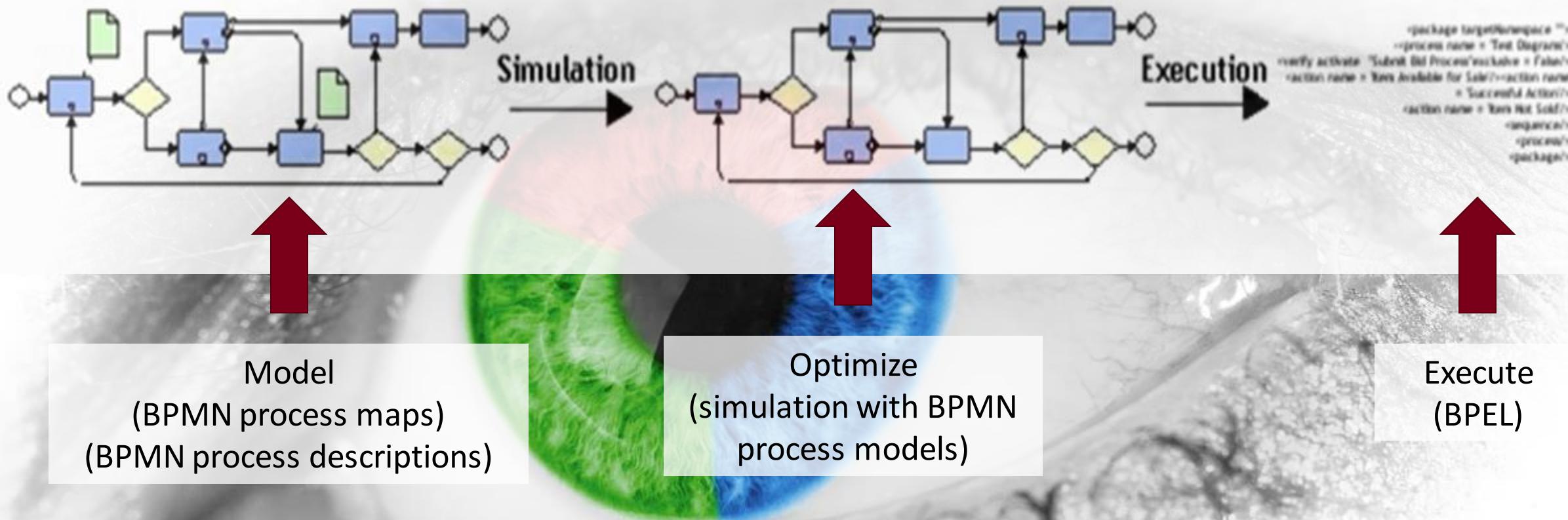
Process Modeling: The Vision



BPMN: the vision



BPMN: The Vision



BPMN Components

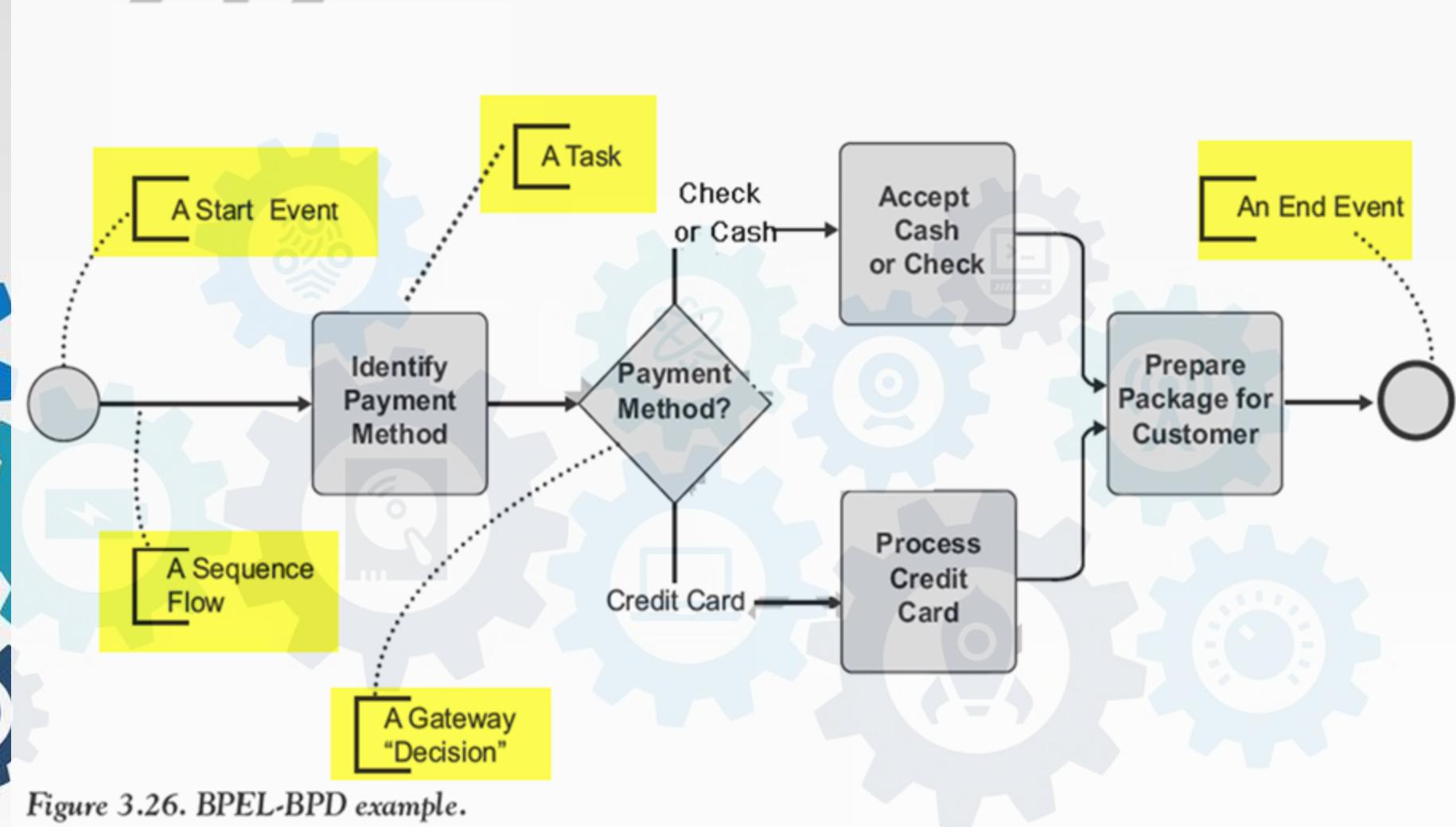


Figure 3.26. BPEL-BPD example.

BPMN Tools



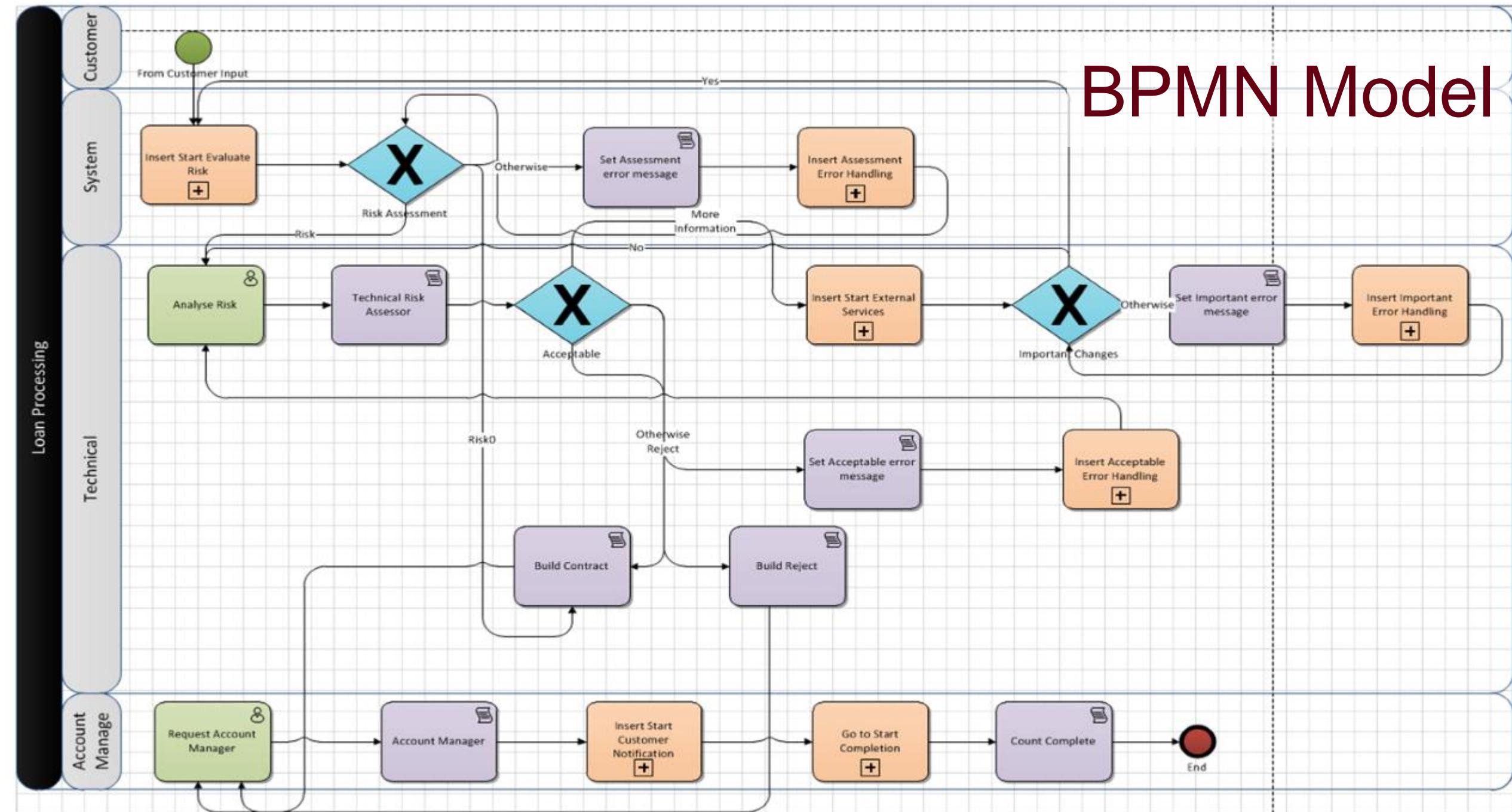
Available in Visio
2010 Premium only

Fully available in
starting in Visio
2013

Screenshot of the Microsoft Visio ribbon showing the 'Shapes' section. The 'BPMN Events (US units)' category is highlighted with a yellow background. A list of BPMN event shapes is displayed below:

Event Type	Shape Description
Start Event	Open circle
Intermediate Event	Circle with a dot
Intermediate (Throwing) Event	Circle with a diagonal line
End Event	Filled circle
Start Message Event	Envelope icon
Intermediate Message Event	Envelope with a dot
Intermediate (Throwing) Message Event	Envelope with a diagonal line
End Message Event	Envelope with a cross
Start Timer Event	Clock icon
Intermediate Timer Event	Clock with a dot
Intermediate (Throwing) Timer Event	Clock with a diagonal line
End Error Event	Cancel icon
Intermediate Cancel Event	Cancel with a dot
End Cancel Event	Cancel with a cross
End Terminate Event	Circle with a cross
Intermediate Compensate Event	Cancel with a diagonal line
End Compensate Event	Cancel with a diagonal line and a cross
Start Condition...	Document icon
Intermediate (Throwing) Condition...	Document with a diagonal line
Intermediate Condition...	Document with a cross
Start Link Event	Document with a diagonal line and a dot
Intermediate Link Event	Document with a diagonal line and a cross
Intermediate (Throwing) Link Event	Document with a diagonal line and a diagonal line
Start Signal Event	Triangle icon
Intermediate Signal Event	Triangle with a dot
Intermediate (Throwing) Signal Event	Triangle with a diagonal line
End Signal Event	Triangle with a cross
Start Multiple Event	Pentagon icon
Intermediate Multiple Event	Pentagon with a dot
Intermediate (Throwing) Multiple Event	Pentagon with a diagonal line
End Multiple Event	Pentagon with a cross

BPMN Model





CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Data Modeling



Learning Objectives



After this lesson, you will be able to:

- Recognize an entity-relationship diagram (ERD)
- Describe 2 different purposes for ERDs
- Give examples of entities, attributes, and relationships

What information is in a Vehicle Order?

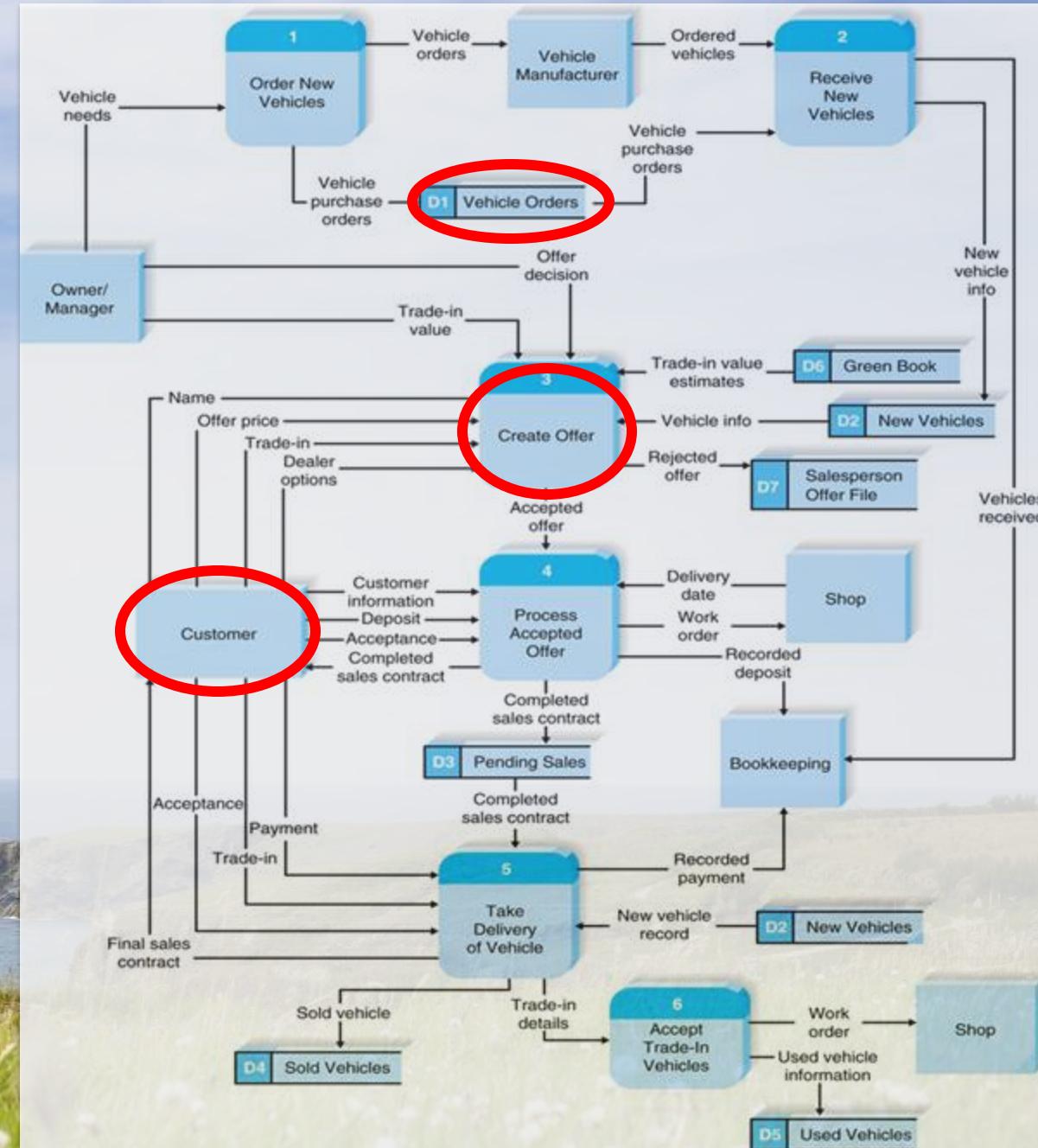
Business Rules

Can an offer be written for more than one vehicle (e.g. a fleet of vehicles)?

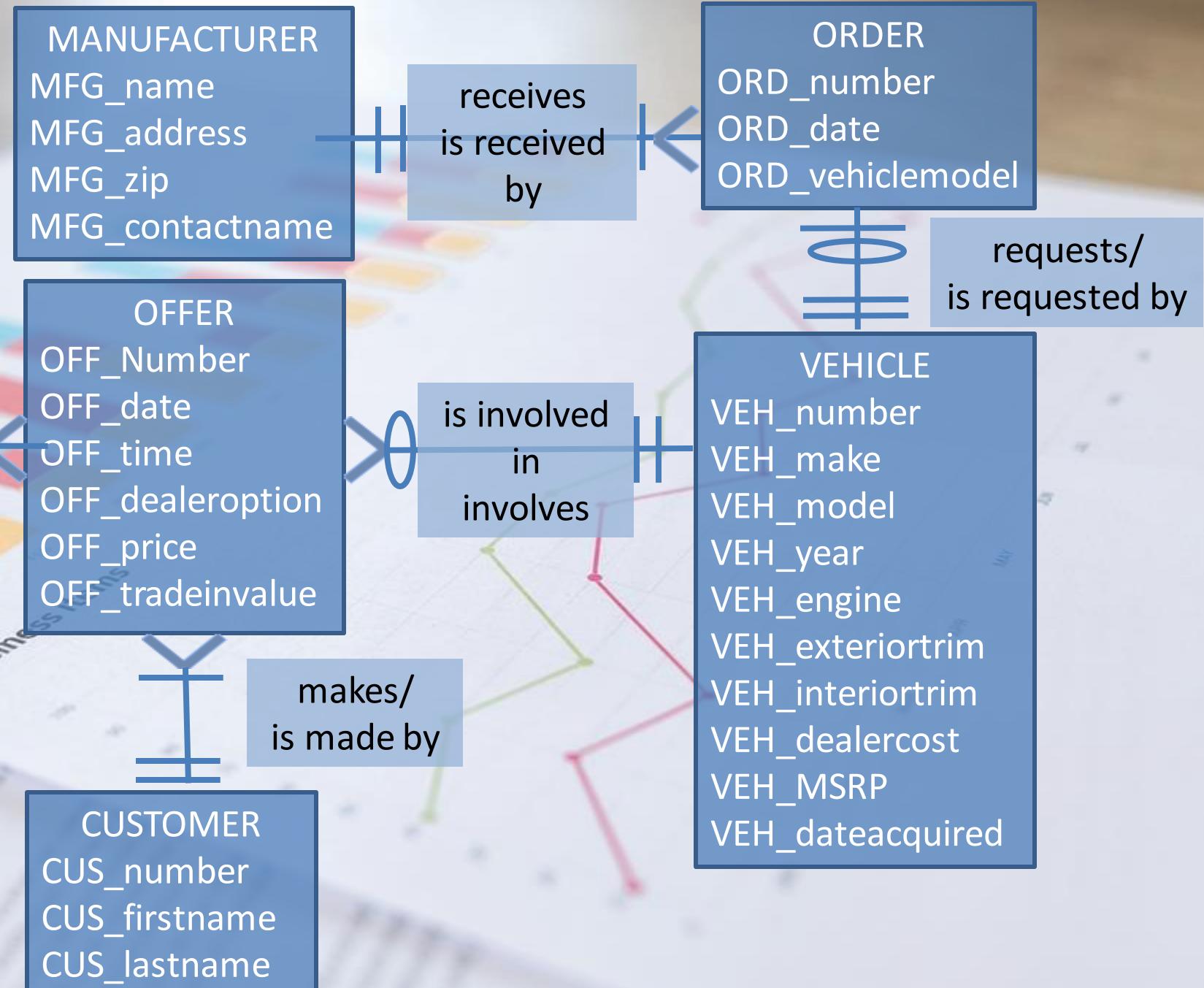
Information Requirements

What do we need to know about Customers?

Can an offer be written for more than one vehicle such as a fleet? This information isn't captured in data flow diagrams such as the one on this page, instead, we need a different type of diagram that shows the data we collect and the relationships between that data. This type of diagram is called an entity relationship diagram.

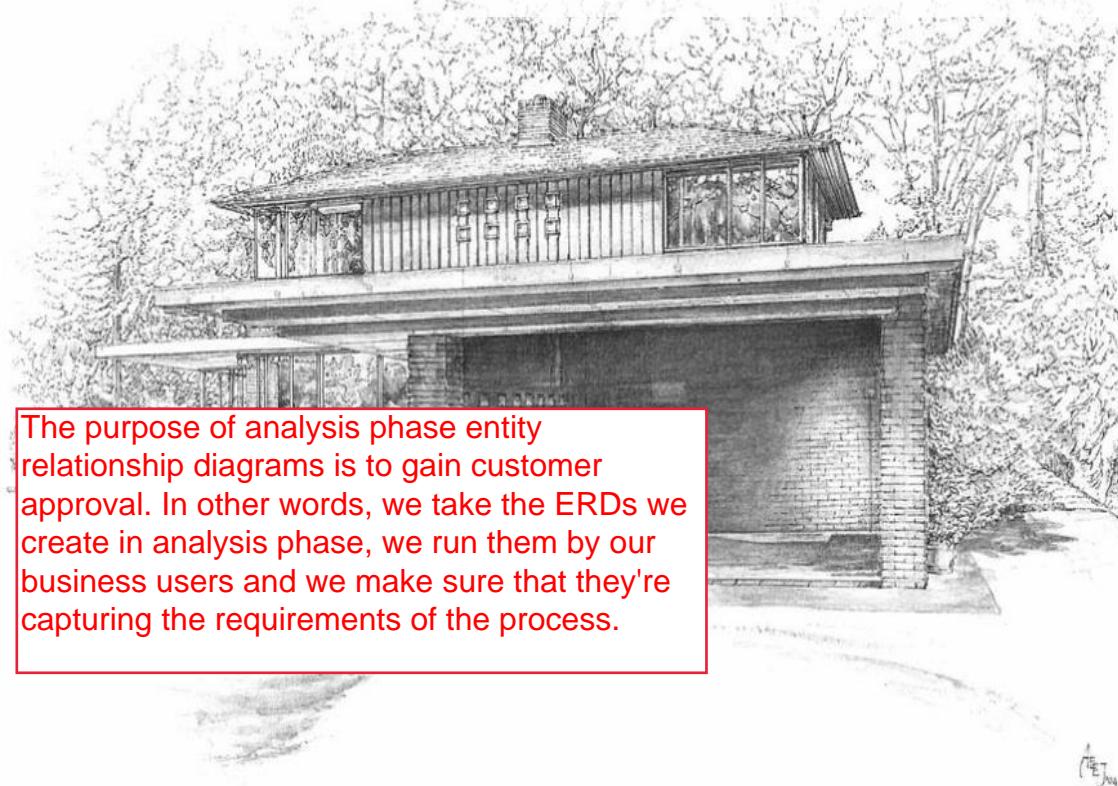


A Data Model Describes Data



One of the most common notations in use today is often called crow's foot notation. The lines that you see connecting the various entities on the diagram often end with a crow's foot,

There Are Different Types of Data Models



The purpose of analysis phase entity relationship diagrams is to gain customer approval. In other words, we take the ERDs we create in analysis phase, we run them by our business users and we make sure that they're capturing the requirements of the process.

Architect's rendering
“Analysis”

Purpose: To gain client approval

Engineer's drawing
“Design”

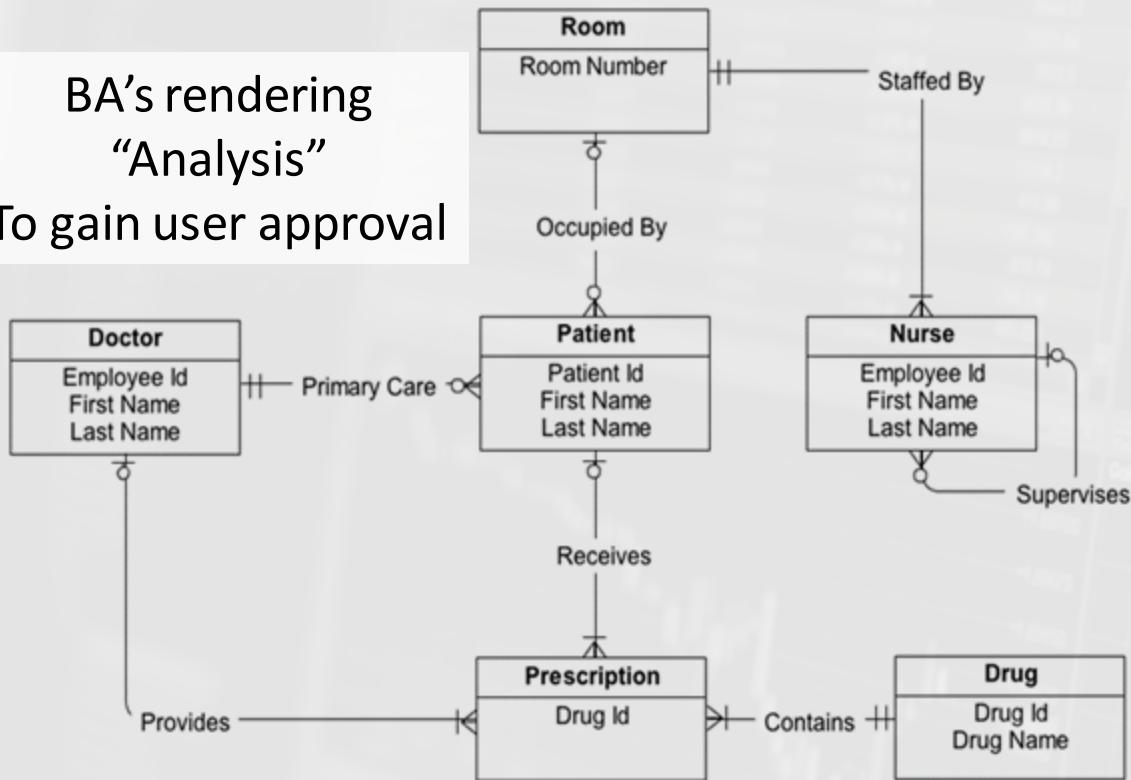
Purpose: To give builders direction



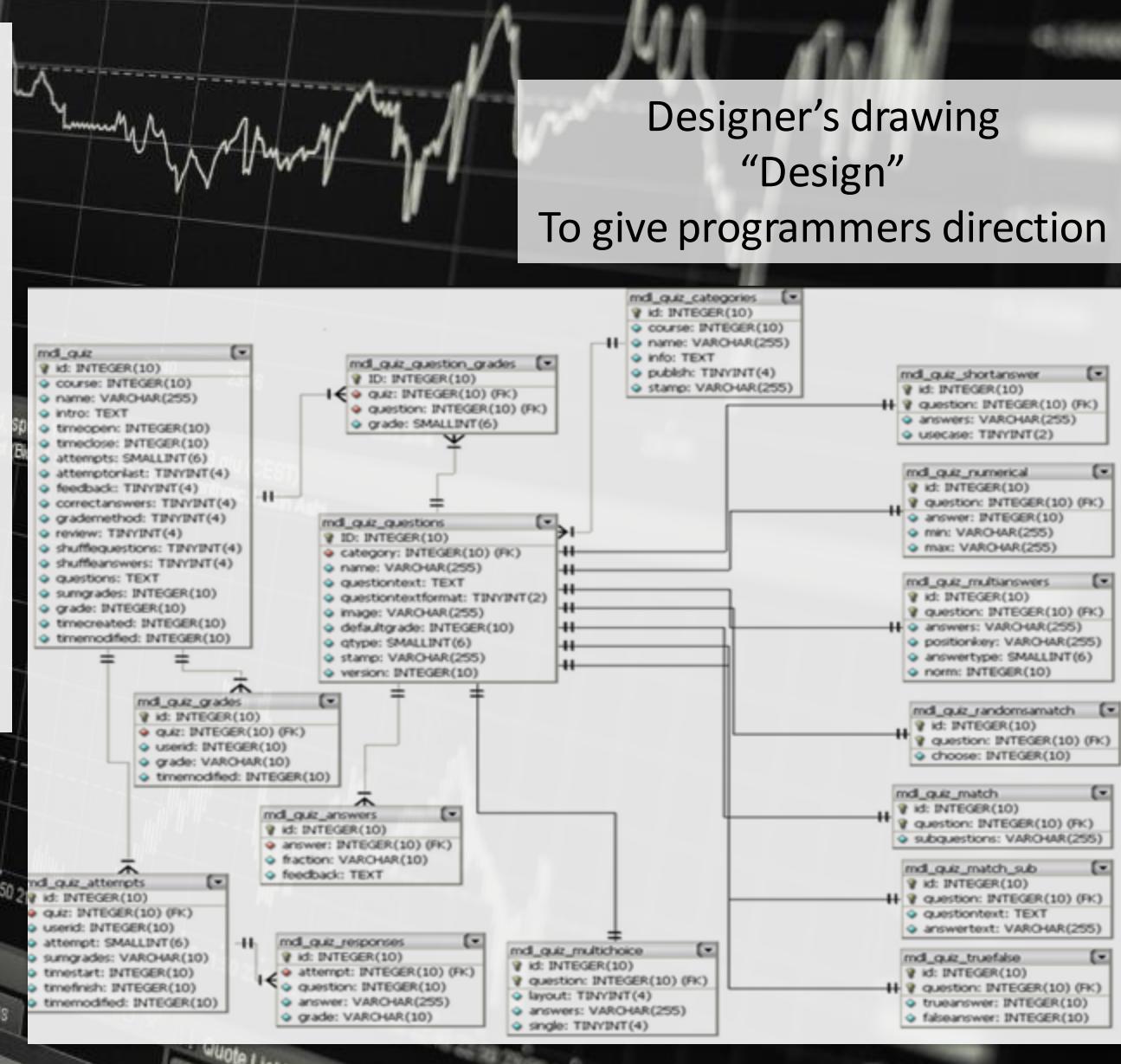
There Are Different Types of Data Models

BA's rendering
"Analysis"

To gain user approval



On the left, you have the analysis phase ERD, which is where we'll spend most of our time. On the right, you have more of a design phase ERD that includes much more information.



Designer's drawing
"Design"

To give programmers direction

Entity

Person, place, object, event or concept about which data are to be maintained

- Person: EMPLOYEE, STUDENT, PATIENT
- Place: STORE, WAREHOUSE, STATE
- Object: MACHINE, BUILDING, AUTOMOBILE, PRODUCT
- Event: SALE, REGISTRATION, RENEWAL
- Concept: ACCOUNT, COURSE, WORK_CENTER

Naming Conventions

Entity versus Entity Instance

Entity

It's important to differentiate between an entity and an entity instance. So, the entity itself might be called employee but a particular entity instance would be a specific employee, like Ken.

Entity

Pet Sitters Unlimited is a company providing pet-sitting services for homeowners while they are on vacation. Pet Sitters Unlimited keeps track of the data about its clients and their pets. In addition, Pet Sitters also keeps track of the families' vets, and the vets' phone number.

- What are our candidate entities?
- Give examples of entity instances?

Entity

Entity

Candidate entities might be things like pet, owner, veterinarian, and so on. They are the major nouns and the problem about which we want to keep information.

Example entity instances might be Fido, your dog.

The phone number in this case is not quite an entity of itself, even though it's a noun, it's something that would likely become an attribute.

Pet Sitters Unlimited is a company providing pet-sitting services for homeowners while they are on vacation. Pet Sitters Unlimited keeps track of the data about its clients and their pets. In addition, Pet Sitters also keeps track of the families' vets, and the vets' phone number.

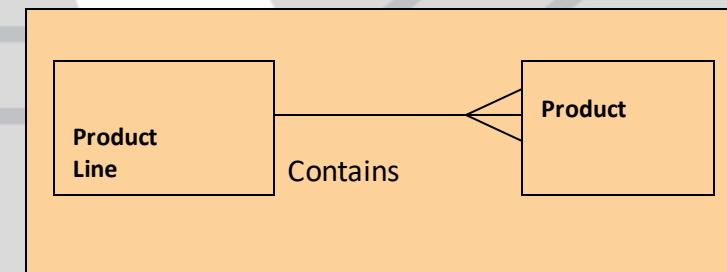
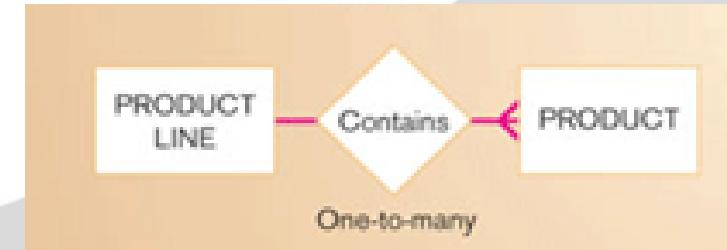
- What are our candidate entities?
- Give examples of entity instances?

Entity

Relationship

The glue that hold entities together. An association between instances of one or more entities that is of interest to the organization

- Registers for
- Teaches
- Is located in
- Receives grade for



Symbols
Naming Conventions
Cardinalities and Modalities

In the top example, we have an example of a relationship in a slightly different notation where the relationship is labeled using a diamond symbol. On the bottom, we have a perhaps a more common notation where the relationship is represented by a straight line, and then the label is just floating in free space.

Cardinality of the relationship, in this example, is equal to the number of products.

Modality refers to the concept of whether a particular entity has to have a relationship with another entity. So, for example, in our example here, does a product line have to have at least one product?

Relationship

Pet Sitters Unlimited is a company providing pet-sitting services for homeowners while they are on vacation. Pet Sitters Unlimited keeps track of the data about its clients and their pets. In addition, Pet Sitters also keeps track of the families' vets, and the vets' phone number.

- What are our relationships?
- Modality and Cardinality?

Can an owner have more than one pet?

Can a pet go to more than one veterinarian?

Does an owner have to have a pet?

We might argue that if they don't have a pet, then they aren't an owner or a customer of ours because they don't have a need for our services.

We might also ask the question, does a pet have to have a particular vet?

In this case, the answer might be yes and the answer might be no, that would be the modality of this particular relationship.

Attribute

An attribute is a property or characteristic of an entity or relationship that is of interest to the organization. So, for example, students might have a student ID, a name, a home address and so on. An automobile might have a vehicle ID, color, weight and so on and so forth.

Property or characteristic of an entity (or relationship) that is of interest to the organization

- STUDENT: Student_ID, Student_Name, Home_Address
- AUTOMOBILE: Vehicle_ID, Color, Weight, Horsepower
- EMPLOYEE: Employee_ID, Employee_Name, Birthday

Symbols

Naming conventions

Candidate identifier

MANUFACTURER

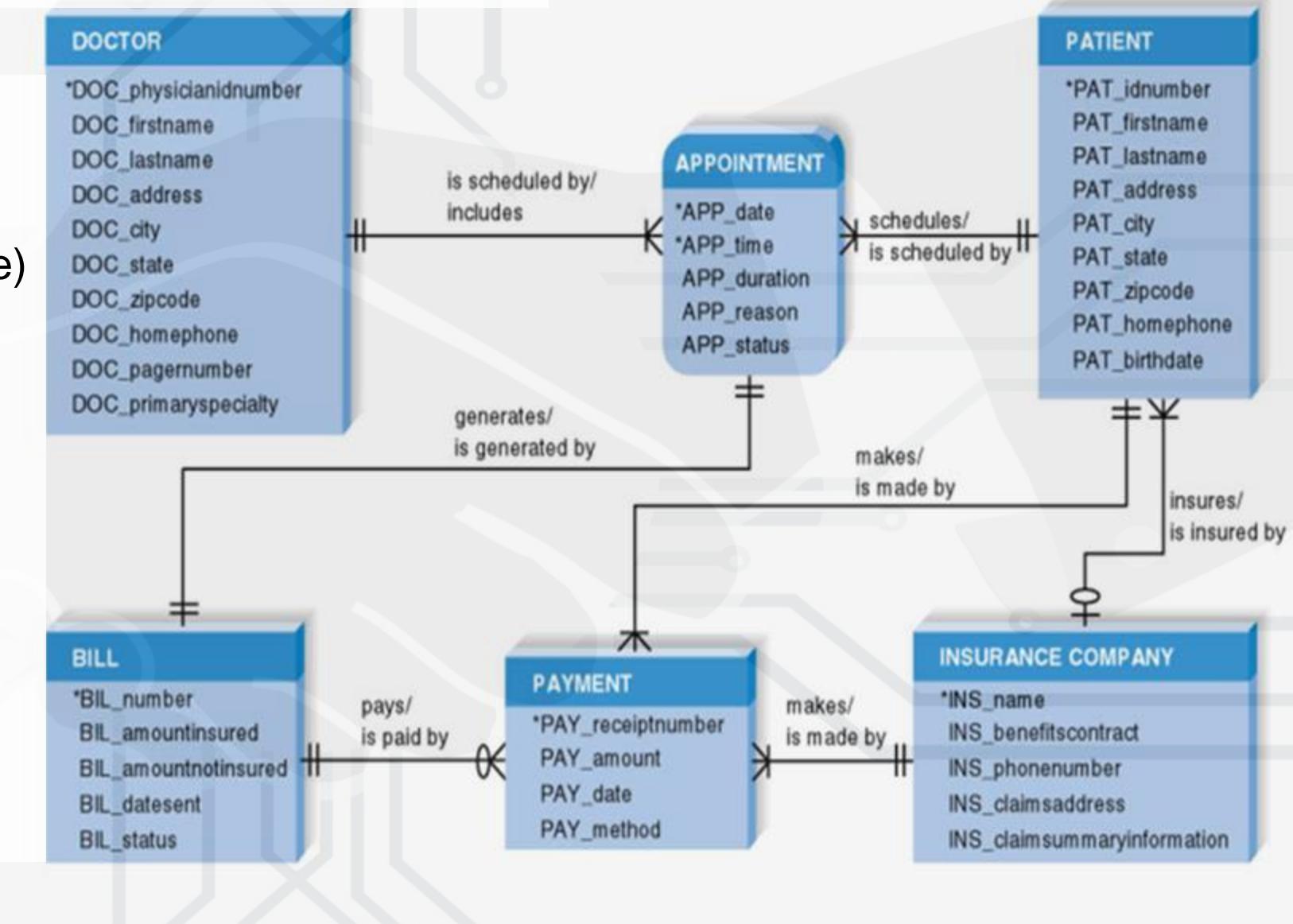
MFG_name
MFG_address
MFG_city
MFG_state
MFG_zipcode
MFG_contactname
MFG_contactphone

Entity Relationship Diagram

Entities which are represented by the boxes you see, relationships which are the lines and attributes which are listed in each entity itself.

Main Constructs

- ✓ Entities
 - Intersection (Associative) Entities
 - Subtypes & Supertypes
- ✓ Relationships
 - ✓ Cardinality
 - ✓ Modality
 - Hierarchies (recursive relationships)
- ✓ Attributes
 - ✓ Identifiers





CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Intersection Entities



Learning Objectives



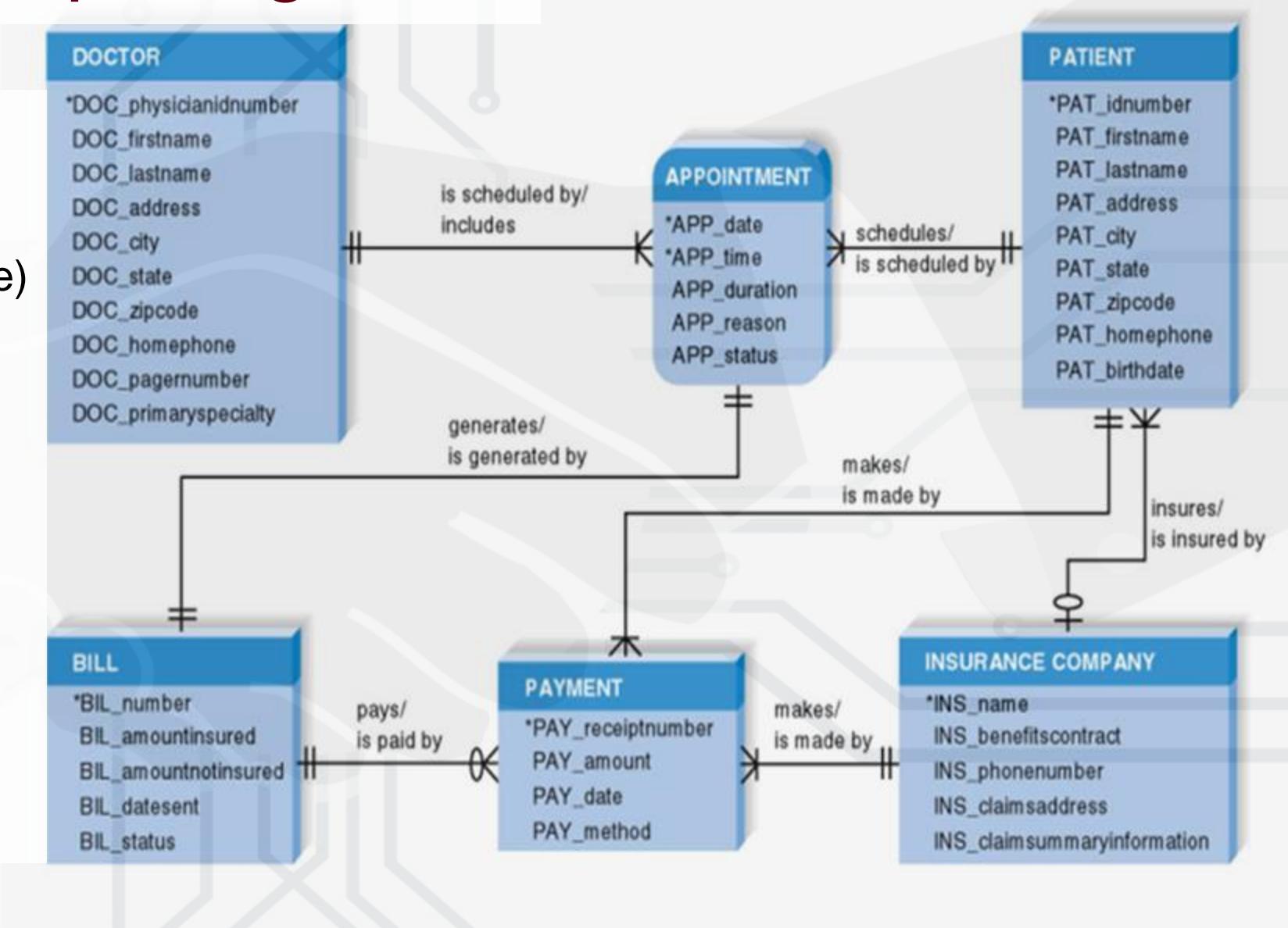
After this lesson, you will be able to:

- Define the term “intersection entity” in the context of entity-relationship diagrams
- Describe real-world situations that we might model with an intersection entity
- Define a foreign key, and determine whether or not to include foreign keys in your ERDs

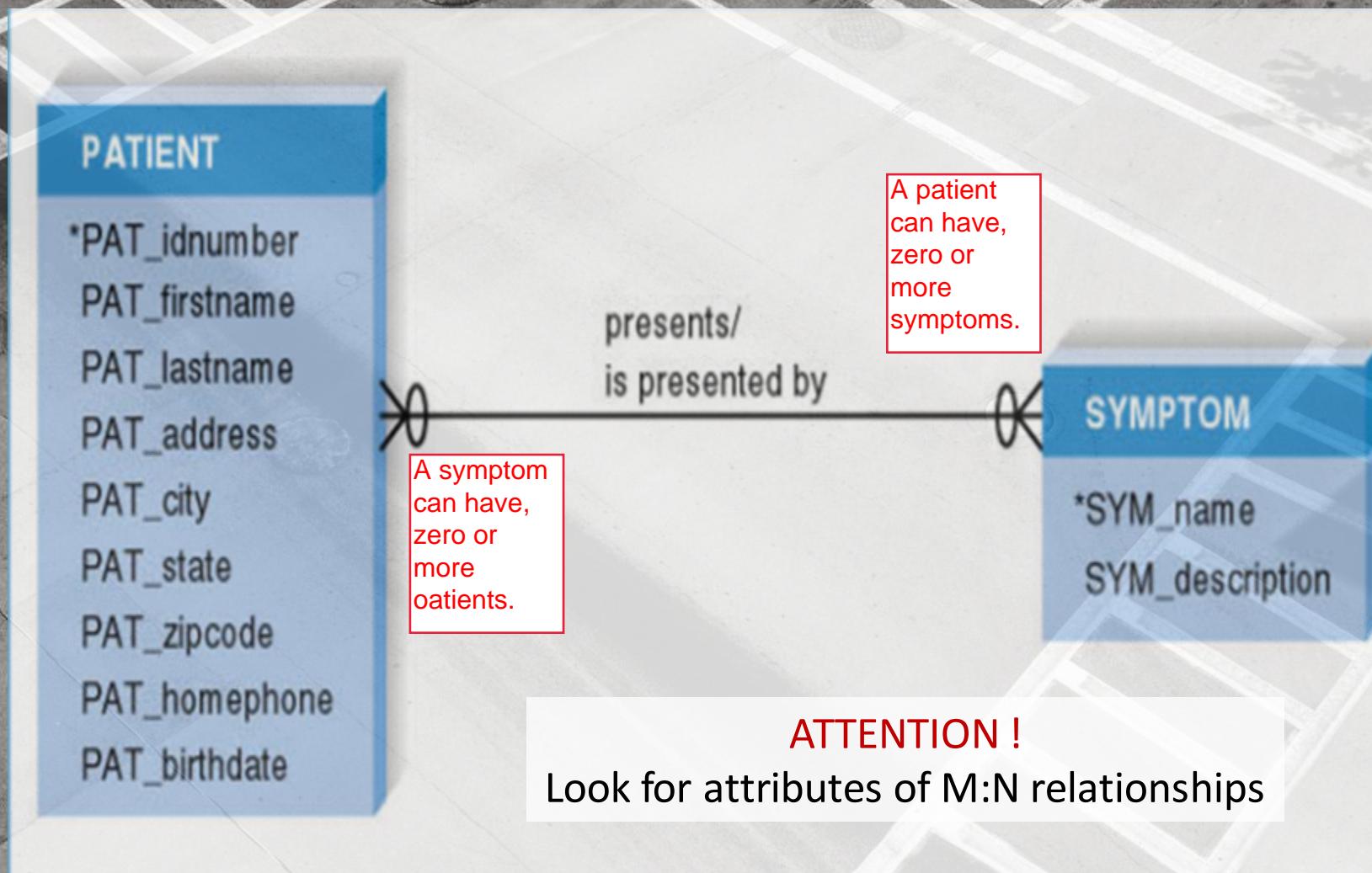
Entity Relationship Diagram

Main Constructs

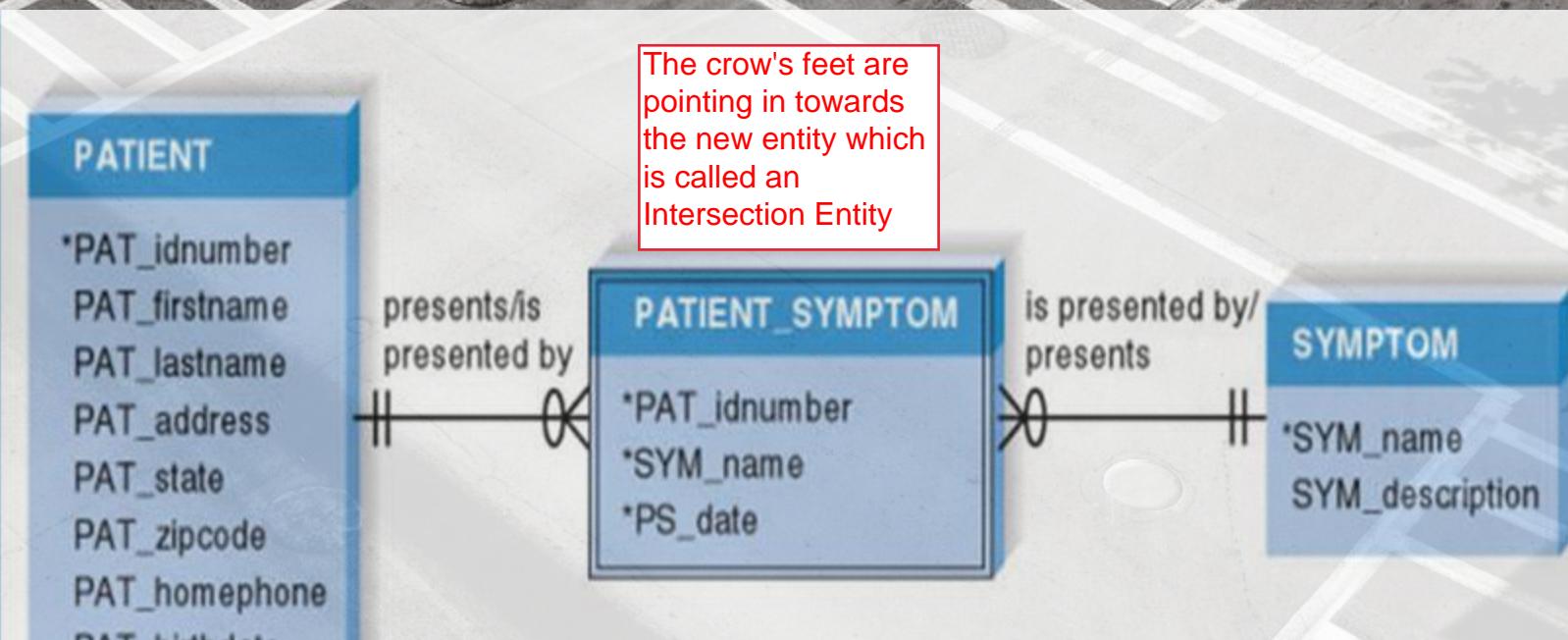
- ✓ Entities
 - Intersection (Associative) Entities
 - Subtypes & Supertypes
- ✓ Relationships
 - ✓ Cardinality
 - ✓ Modality
 - Hierarchies (recursive relationships)
- ✓ Attributes
- ✓ Identifiers



Intersection Entity



Intersection Entity



Relationships: Two 1:N relationships

Entity Name:

Option 1: Concatenate names of original entities

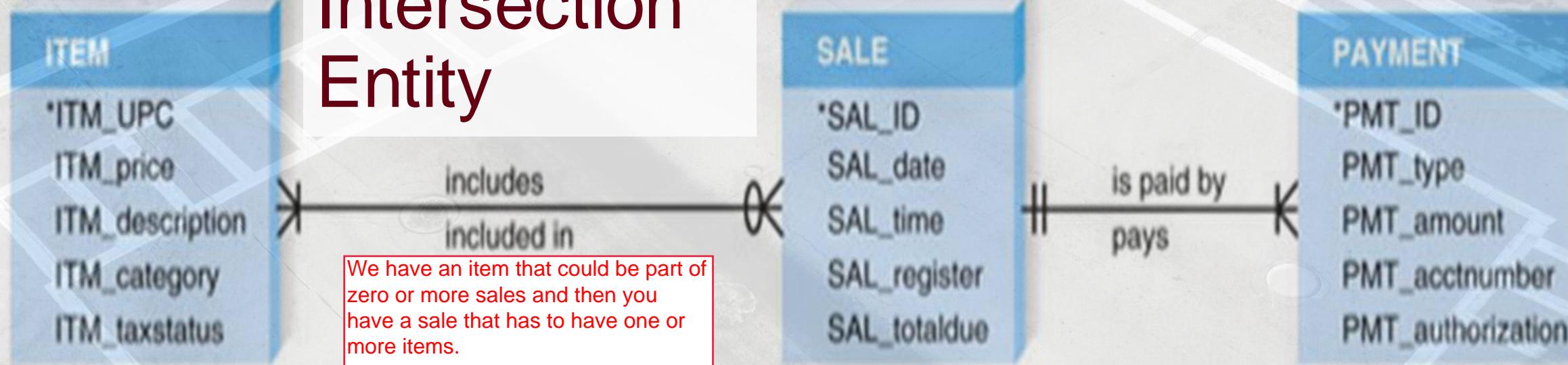
Option 2: Realistic name

Candidate Identifier:

Option A: An ID-type identifier

Option B: Both identifiers from original entities = concatenated identifier (Add additional identifier if needed for uniqueness)

Intersection Entity

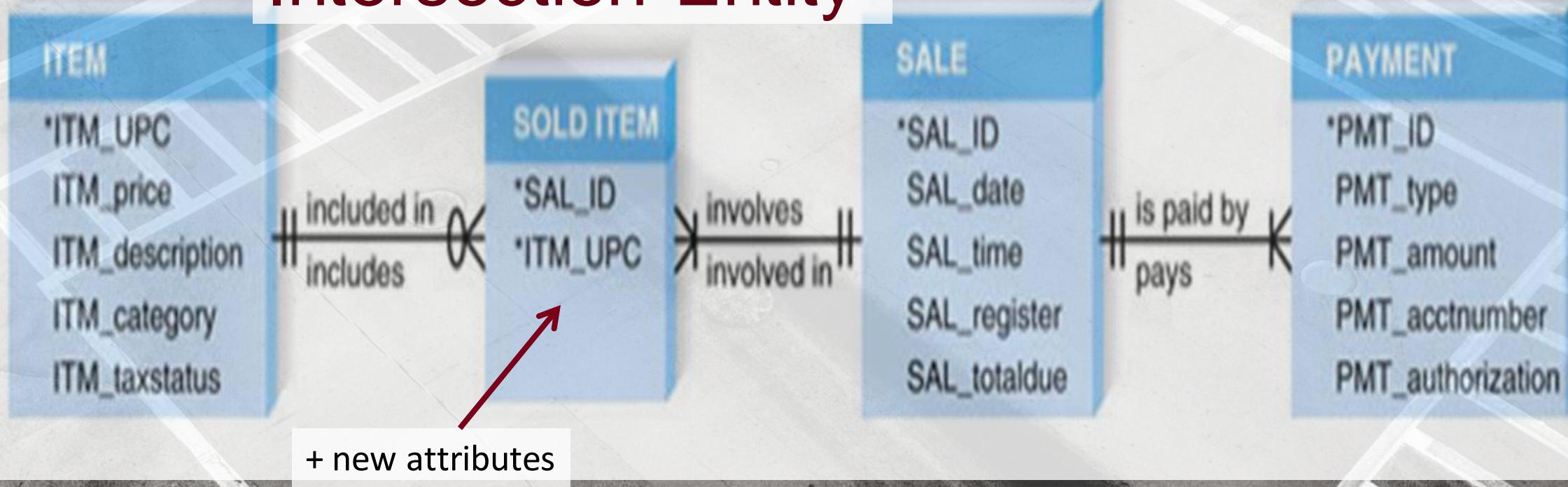


What attributes exist in the M:N relationship
Redraw the relationship

Reading the diagram, between item and sale from left to right, we see on the sales side of the relationship the oval represents a zero modality effectively meaning that an item does not have to be part of any sale, so it could for example just be sitting on the shelf without having sold any of them. Then the crow's foot means that a particular item might be part of one or more sales.

Reading in the opposite direction between sail and item, you can see that the item side of the relationship, there is a vertical bar there, that means that a sale has to include at least one item, and then the crow's foot means that a particular sale could include more than one item.

Intersection Entity



It seems that we need an intersection entity in this case. So, what do we do?

We put the intersection entity in the middle, and we then change our notation so that the crow's feet on the notation points in towards the new intersection entity, we then borrow the identifiers from the related entities.

We include the sale ID and the item UPC, which are the identifiers of the related entities and then we can put any additional attributes that we might want into this entity.



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Subtypes and Supertypes



Learning Objectives

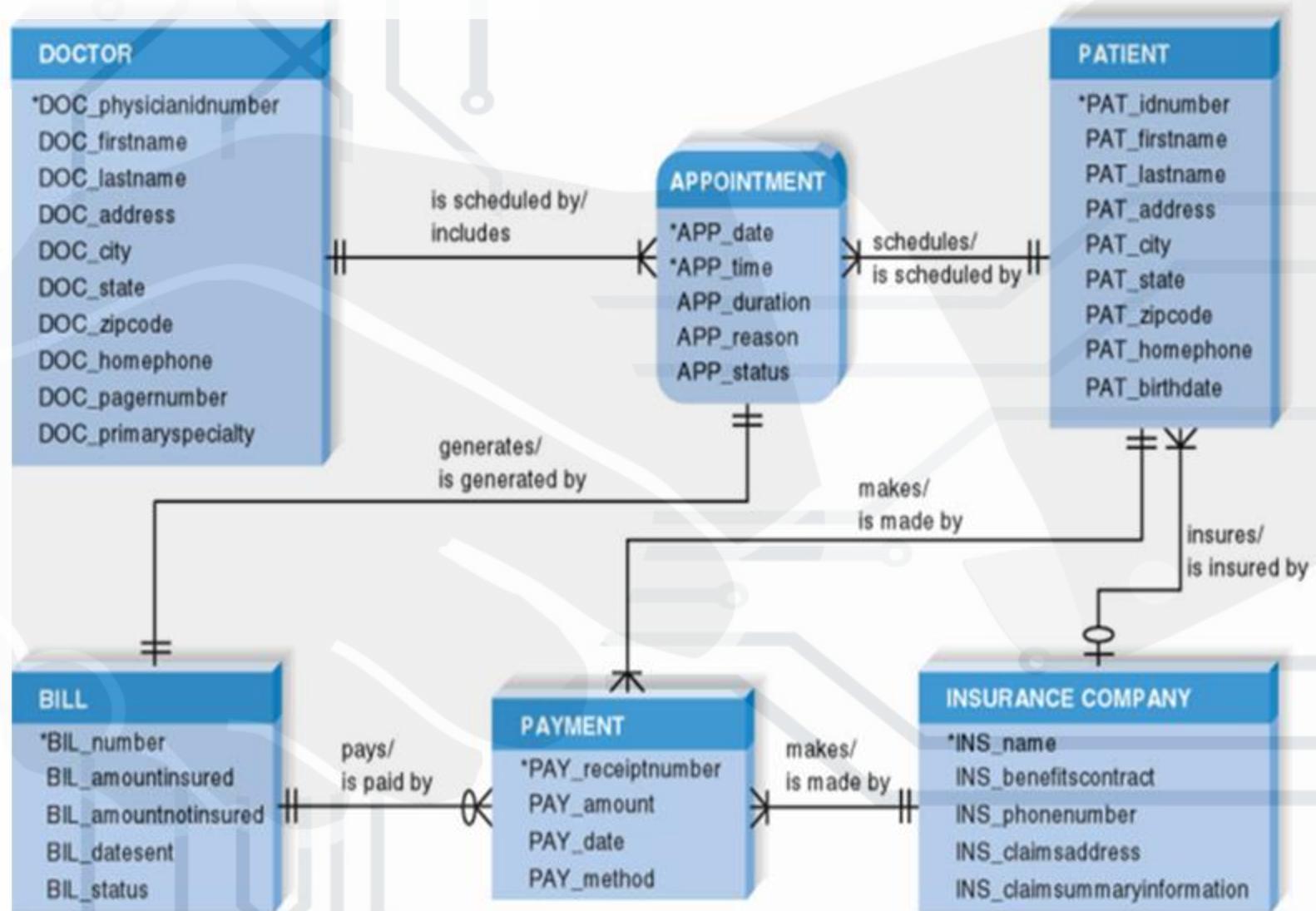
After this video, you will be able to:

- Describe the purpose of subtypes and supertypes in an entity-relationship diagram
- Read ERDs containing subtypes and / or supertypes
- Define the 4 different types of subtype / supertype relationship

Entity Relationship Diagram

Main Constructs

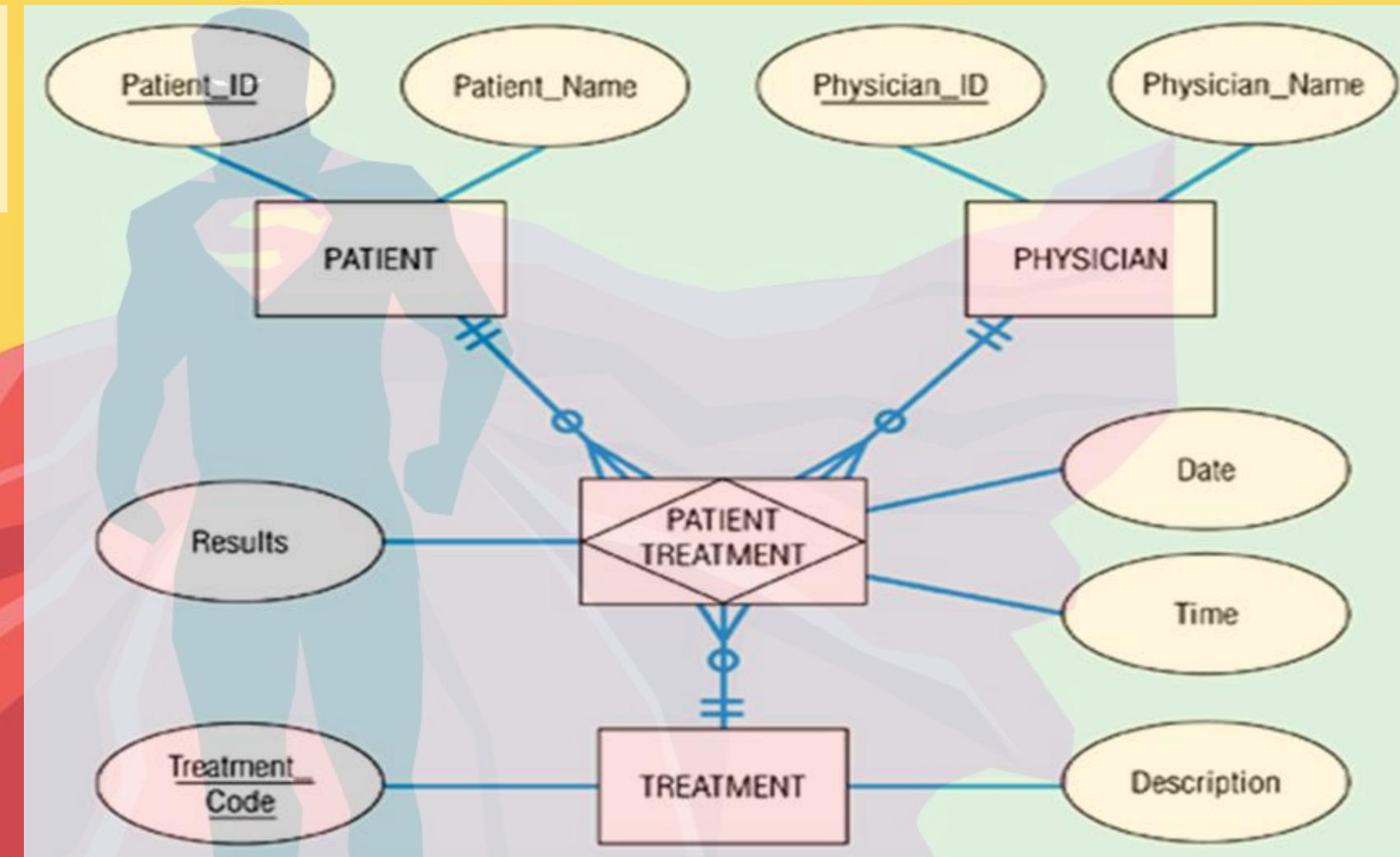
- ✓ Entities
 - Intersection (Associative) Entities
 - Subtypes & Supertypes
- ✓ Relationships
 - ✓ Cardinality
 - ✓ Modality
 - Hierarchies (recursive relationships)
- ✓ Attributes
- ✓ Identifiers



Subtype and Supertype

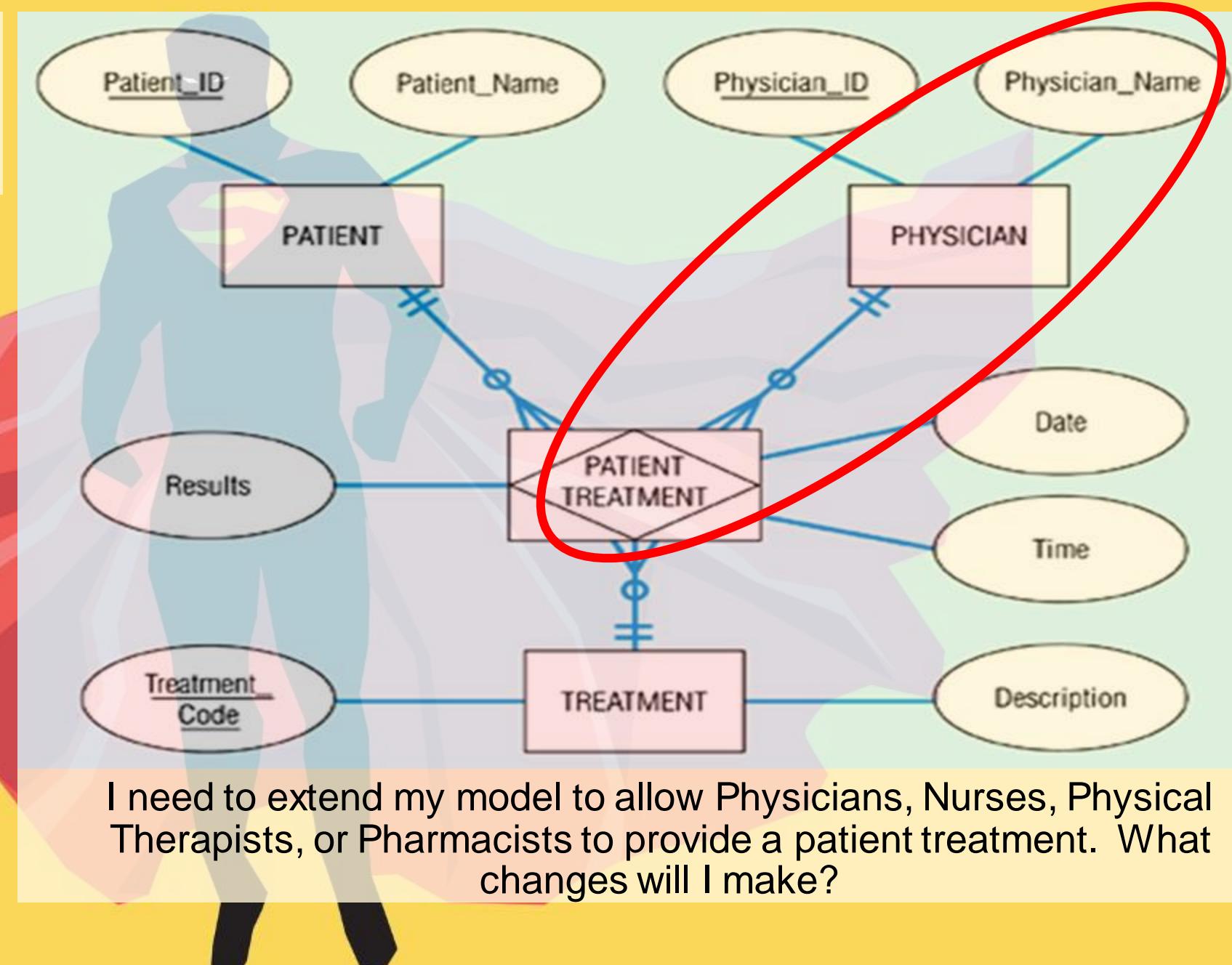
Chen Notation:

- Entities represented by rectangles
- Attributes are connected to the entities by a line and are represented in ovals.
- Relationships are labeled with the diamond (patient treatment)
- Subtypes and supertypes are often called a generalization relationship.
- Subtype inherits attributes from another entity which is called the supertype.
- All attributes of the supertype are also valid for instances of the subtypes.

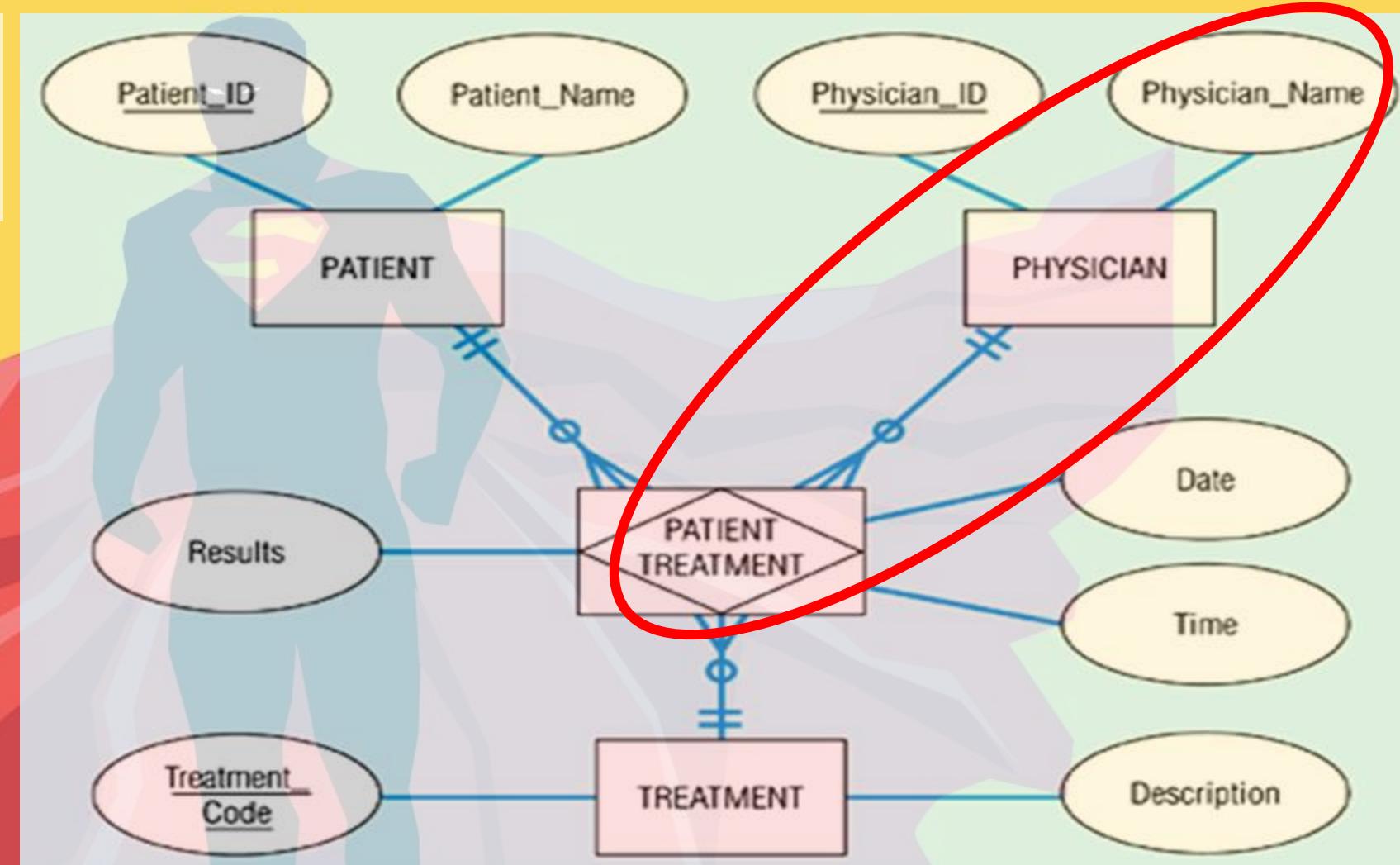


This syntax is different than our text:
what are the differences?

Subtype and Supertype



Subtype and Supertype



Generalization: shows that one entity (subtype) inherits from another entity (supertype), meaning that all the attributes of the supertype are also valid for entity instances of the subtype (p. 527, modified)

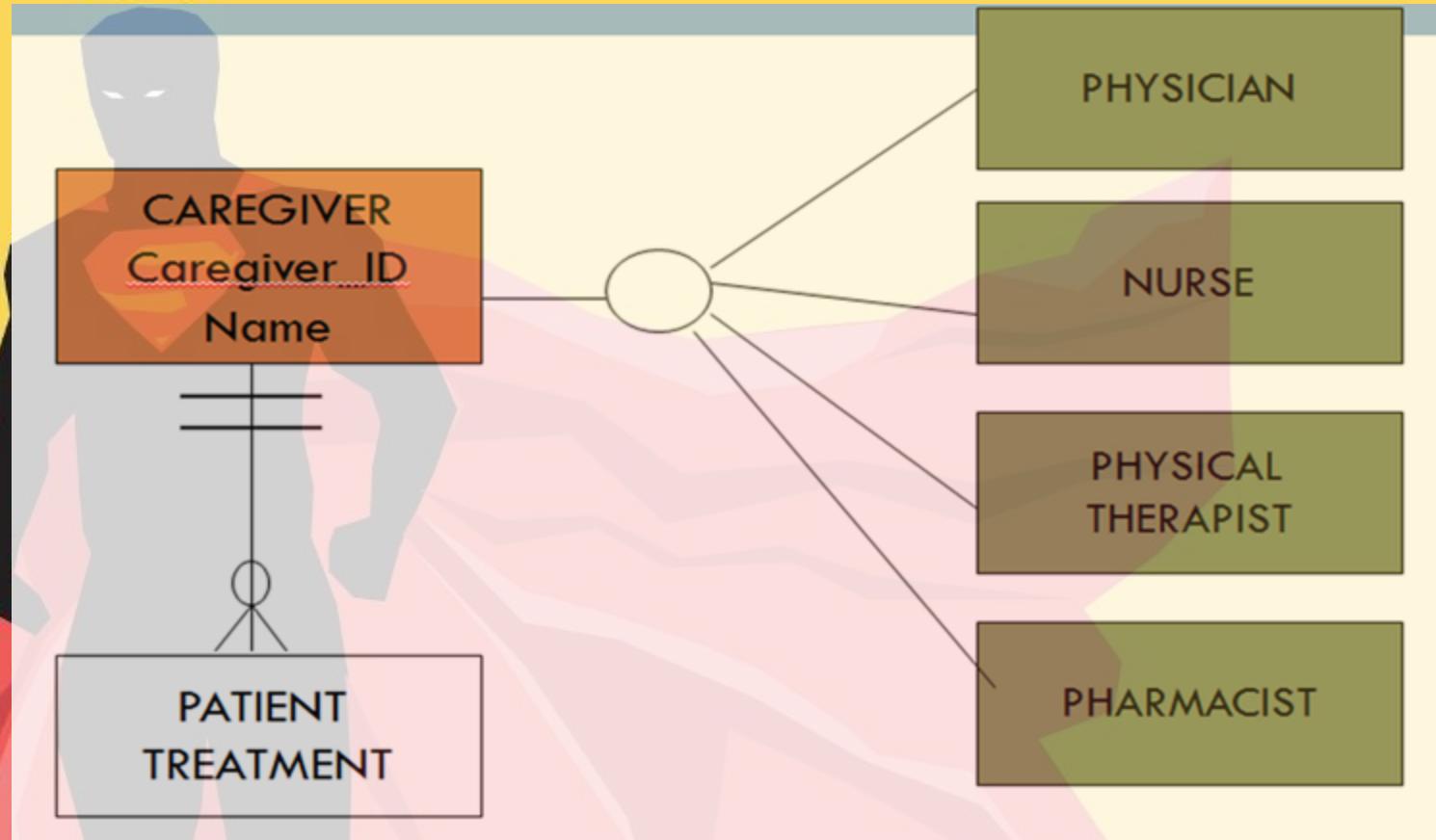
Subtype and Supertype

We create a generalized type on the left here which you see we're going to call caregiver.

We extract the common attributes such as their ID, their name, and anything else that might be common and we put that in the caregiver entity.

We then leave for the individual types such as physician, nurse, physical therapists, and pharmacists, any attributes that might be specific to those individuals.

Caregiver in this case is the supertype and physician, nurse, physical therapist, and pharmacist are all subtypes in this example.



Supertype: An entity whose instances store attributes that are common to one or more entity subtypes

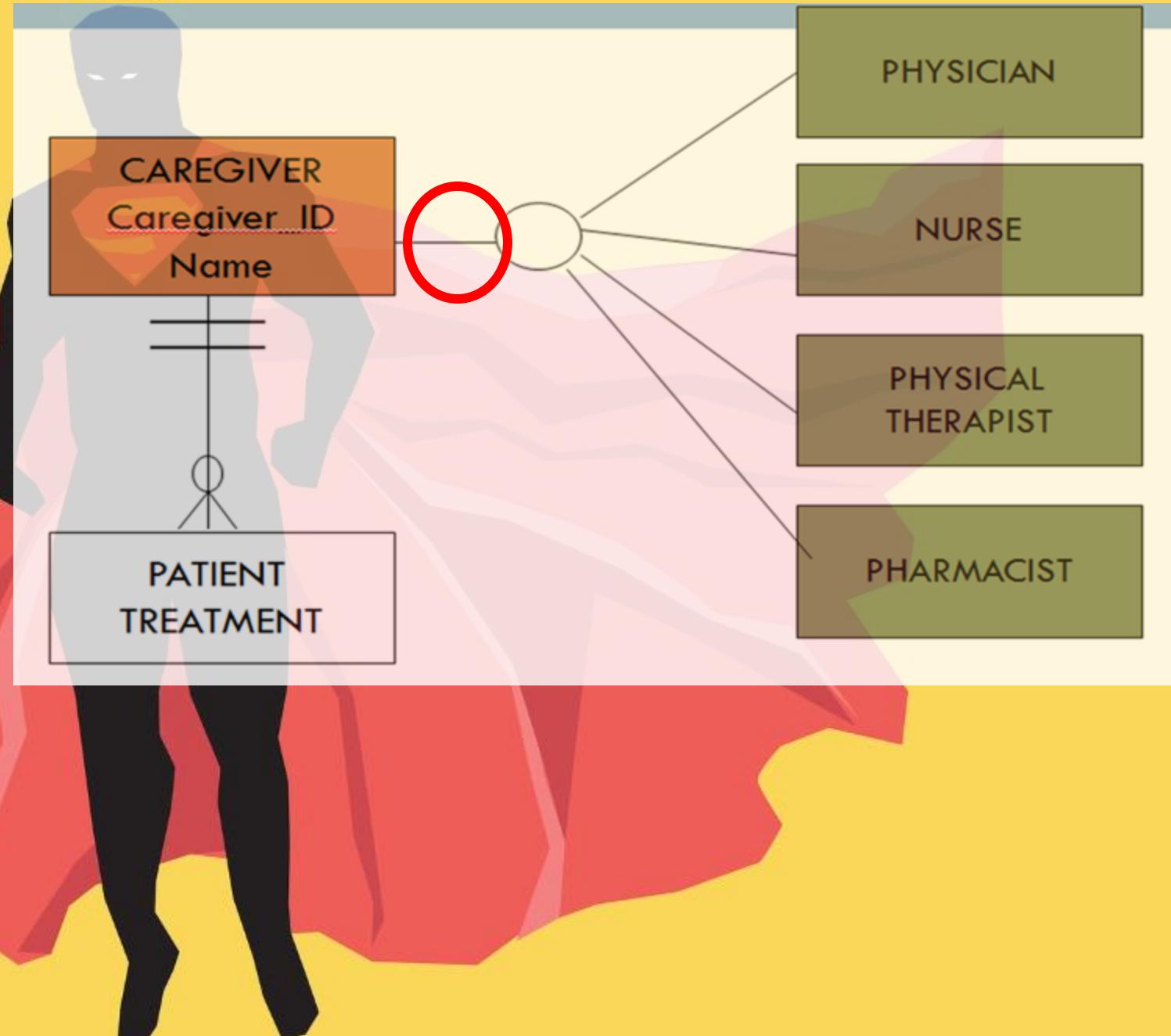
Subtype: An entity whose instances may inherit common attributes from its entity supertype.

Subtype and Supertype

Total Specialization: Each entity instance of a supertype must be a member of some subtype in the relationship

Partial Specialization: An entity instance of the supertype does not have to belong to any subtype

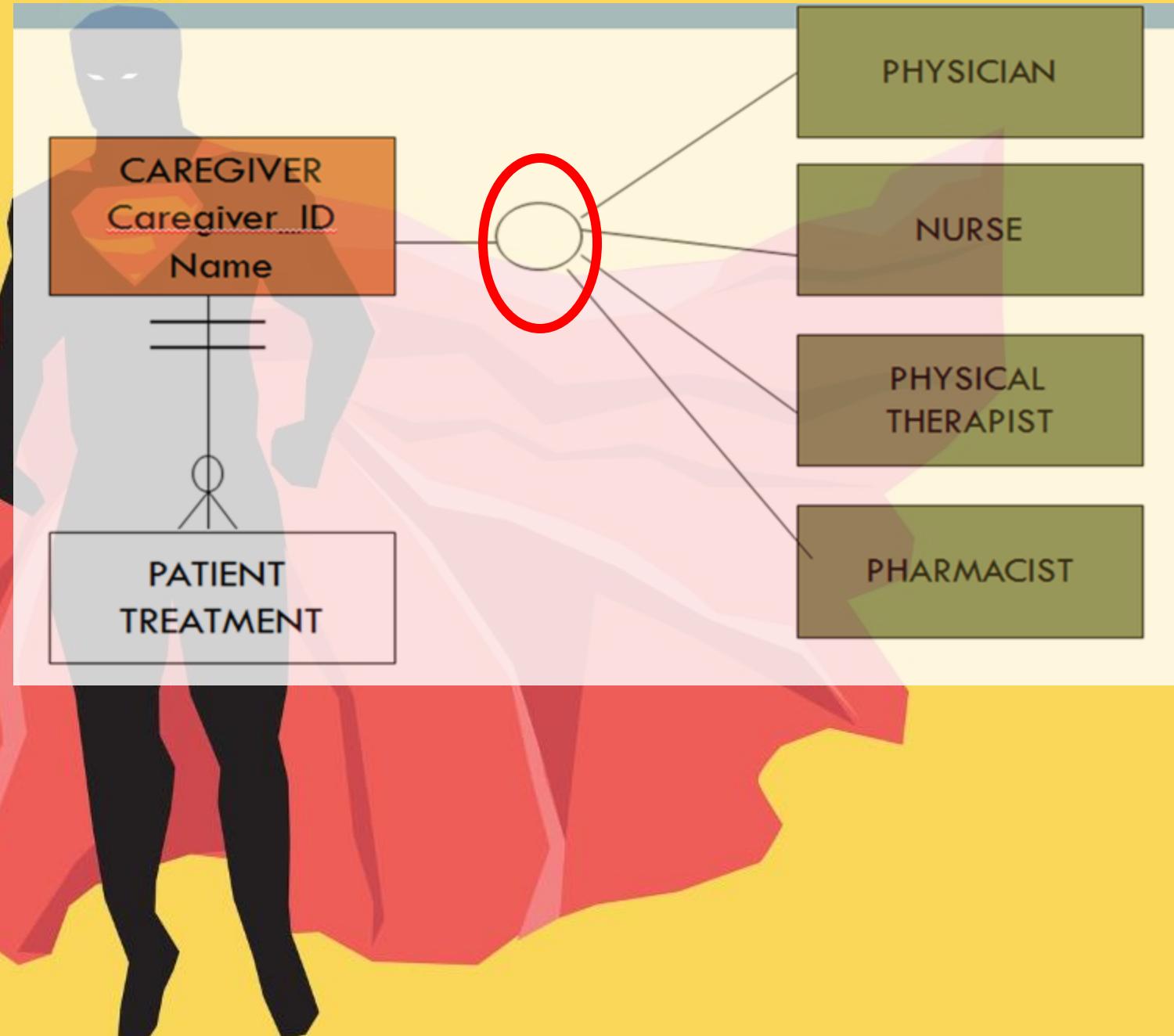
In partial specialization, we might have another type of caregiver other than the four listed here.



Subtype and Supertype

Disjoint: If an entity instance of the supertype is a member of one subtype, it cannot simultaneously be a member of any other subtype

Overlap: An entity instance can simultaneously be a member of two or more subtypes





CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

AIR JAM: The U.S. Federal Aviation Administration spent \$2.6 billion trying to upgrade its air-traffic-control system, only to cancel the project in 1994. Gridlocked skies are still with us today.



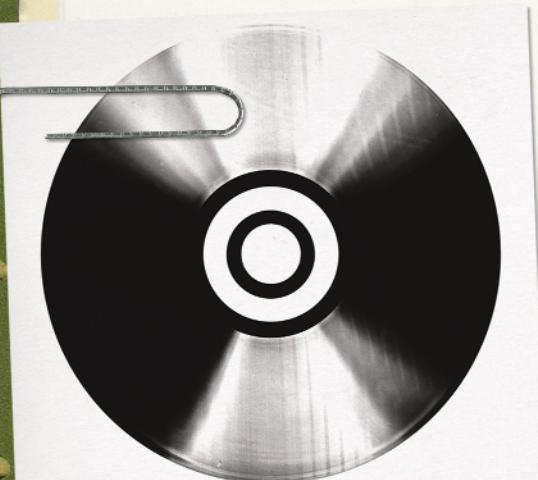
Why Software

FAILS

We waste billions of dollars each year on entirely preventable mistakes

By Robert N. Charette

Have you heard the one about the disappearing warehouse? One day, it vanished—not from physical view, but from the watchful eyes of a well-known retailer's automated distribution system. A software glitch had somehow erased the warehouse's existence, so that goods destined for the warehouse were rerouted elsewhere, while goods at the warehouse languished. Because the company was in financial trouble and had been shuttering other warehouses to save money, the employees at the "missing" warehouse kept quiet. For three years, nothing arrived or left. Employees were still getting their paychecks, however, because a different computer system handled the payroll. When the software glitch finally came to light, the merchandise in the warehouse was sold off, and upper management told employees to say nothing about the episode.



MARKET CRASH: After its new automated supply-chain management system failed last October, leaving merchandise stuck in company warehouses, British food retailer Sainsbury's had to hire 3000 additional clerks to stock its shelves.



This story has been floating around the information technology industry for 20-some years. It's probably apocryphal, but for those of us in the business, it's entirely plausible. Why? Because episodes like this happen all the time. Last October, for instance, the giant British food retailer J Sainsbury PLC had to write off its US \$526 million investment in an automated supply-chain management system. It seems that merchandise was stuck in the company's depots and warehouses and was not getting through to many of its stores. Sainsbury was forced to hire about 3000 additional clerks to stock its shelves manually [see photo, "Market Crash"].

This is only one of the latest in a long, dismal history of IT projects gone awry [see table, "Software Hall of Shame" for other notable fiascoes]. Most IT experts agree that such failures occur far more often than they should. What's more, the failures are universally unprejudiced: they happen in every country; to large companies and small; in commercial, nonprofit, and governmental organizations; and without regard to status or reputation. The business and societal costs of these failures—in terms of wasted taxpayer and shareholder dollars as

well as investments that can't be made—are now well into the billions of dollars a year.

The problem only gets worse as IT grows ubiquitous. This year, organizations and governments will spend an estimated \$1 trillion on IT hardware, software, and services worldwide. Of the IT projects that are initiated, from 5 to 15 percent will be abandoned before or shortly after delivery as hopelessly inadequate. Many others will arrive late and over budget or require massive reworking. Few IT projects, in other words, truly succeed.

The biggest tragedy is that software failure is for the most part predictable and avoidable. Unfortunately, most organizations don't see preventing failure as an urgent matter, even though that view risks harming the organization and maybe even destroying it. Understanding why this attitude persists is not just an academic exercise; it has tremendous implications for business and society.

SOFTWARE IS EVERYWHERE. It's what lets us get cash from an ATM, make a phone call, and drive our cars. A typical cellphone now contains 2 million lines of software code; by 2010 it will likely have 10 times as

Software Hall of Shame

YEAR	COMPANY	OUTCOME (COSTS IN US \$)
2005	Hudson Bay Co. [Canada]	Problems with inventory system contribute to \$33.3 million* loss.
2004–05	UK Inland Revenue	Software errors contribute to \$3.45 billion* tax-credit overpayment.
2004	Avis Europe PLC [UK]	Enterprise resource planning (ERP) system canceled after \$54.5 million† is spent.
2004	Ford Motor Co.	Purchasing system abandoned after deployment costing approximately \$400 million.
2004	J Sainsbury PLC [UK]	Supply-chain management system abandoned after deployment costing \$527 million.†
2004	Hewlett-Packard Co.	Problems with ERP system contribute to \$160 million loss.
2003–04	AT&T Wireless	Customer relations management (CRM) upgrade problems lead to revenue loss of \$100 million.
2002	McDonald's Corp.	The Innovate information-purchasing system canceled after \$170 million is spent.
2002	Sydney Water Corp. [Australia]	Billing system canceled after \$33.2 million† is spent.
2002	CIGNA Corp.	Problems with CRM system contribute to \$445 million loss.
2001	Nike Inc.	Problems with supply-chain management system contribute to \$100 million loss.
2001	Kmart Corp.	Supply-chain management system canceled after \$130 million is spent.
2000	Washington, D.C.	City payroll system abandoned after deployment costing \$25 million.
1999	United Way	Administrative processing system canceled after \$12 million is spent.
1999	State of Mississippi	Tax system canceled after \$11.2 million is spent; state receives \$185 million damages.
1999	Hershey Foods Corp.	Problems with ERP system contribute to \$151 million loss.
1998	Snap-on Inc.	Problems with order-entry system contribute to revenue loss of \$50 million.
1997	U.S. Internal Revenue Service	Tax modernization effort canceled after \$4 billion is spent.
1997	State of Washington	Department of Motor Vehicle (DMV) system canceled after \$40 million is spent.
1997	Oxford Health Plans Inc.	Billing and claims system problems contribute to quarterly loss; stock plummets, leading to \$3.4 billion loss in corporate value.
1996	Arianespace [France]	Software specification and design errors cause \$350 million Ariane 5 rocket to explode.
1996	FoxMeyer Drug Co.	\$40 million ERP system abandoned after deployment, forcing company into bankruptcy.
1995	Toronto Stock Exchange [Canada]	Electronic trading system canceled after \$25.5 million** is spent.
1994	U.S. Federal Aviation Administration	Advanced Automation System canceled after \$2.6 billion is spent.
1994	State of California	DMV system canceled after \$44 million is spent.
1994	Chemical Bank	Software error causes a total of \$15 million to be deducted from 100 000 customer accounts.
1993	London Stock Exchange [UK]	Taurus stock settlement system canceled after \$600 million** is spent.
1993	Allstate Insurance Co.	Office automation system abandoned after deployment, costing \$130 million.
1993	London Ambulance Service [UK]	Dispatch system canceled in 1990 at \$11.25 million**; second attempt abandoned after deployment, costing \$15 million.**
1993	Greyhound Lines Inc.	Bus reservation system crashes repeatedly upon introduction, contributing to revenue loss of \$61 million.
1992	Budget Rent-A-Car, Hilton Hotels, Marriott International, and AMR [American Airlines]	Travel reservation system canceled after \$165 million is spent.

Sources: *Business Week*, *CEO Magazine*, *Computerworld*, *InfoWeek*, *Fortune*, *The New York Times*, *Time*, and *The Wall Street Journal*

* Converted to U.S. dollars using current exchange rates as of press time.

† Converted to U.S. dollars using exchange rates for the year cited, according to the International Trade Administration, U.S. Department of Commerce.

** Converted to U.S. dollars using exchange rates for the year cited, according to the *Statistical Abstract of the United States*, 1996.

many. General Motors Corp. estimates that by then its cars will each have 100 million lines of code.

The average company spends about 4 to 5 percent of revenue on information technology, with those that are highly IT dependent—such as financial and telecommunications companies—spending more than 10 percent on it. In other words, IT is now one of the largest corporate expenses outside employee costs. Much of that money goes into hardware and software upgrades, software license fees, and so forth, but a big chunk is for new software projects meant to create a better future for the organization and its customers.

Governments, too, are big consumers of software. In 2003, the United Kingdom had more than 100 major government IT projects under way that totaled \$20.3 billion. In 2004, the U.S. government cataloged 1200 civilian IT projects costing more than \$60 billion, plus another \$16 billion for military software.

Any one of these projects can cost over \$1 billion. To take two current examples, the computer modernization effort at the U.S. Department of Veterans Affairs is projected to run \$3.5 billion, while automating the health records of the UK's National Health Service is likely to cost more than \$14.3 billion for development and another \$50.8 billion for deployment.

Such megaprojects, once rare, are now much more common, as smaller IT operations are joined into “systems of systems.” Air traffic control is a prime example, because it relies on connections among dozens of networks that provide communications, weather, navigation, and other data. But the trick of integration has stymied many an IT developer, to the point where academic researchers increasingly believe that computer science itself may need to be rethought in light of these massively complex systems.

WHEN A SOFTWARE PROJECT FAILS, it jeopardizes an organization's prospects. If the failure is large enough, it can steal the company's entire future. In one stellar meltdown, a poorly implemented resource planning system led FoxMeyer Drug Co., a \$5 billion wholesale drug distribution company in Carrollton, Texas, to plummet into bankruptcy in 1996.

IT failure in government can imperil national security, as the FBI's Virtual Case File debacle has shown. The \$170 million VCF system, a searchable database intended to allow agents to “connect the dots” and follow up on disparate pieces of intelligence, instead ended five months ago without any system's being deployed [see “Who Killed the Virtual Case File?” in this issue].

IT failures can also stunt economic growth and quality of life. Back in 1981, the U.S. Federal Aviation Administration began looking into upgrading its antiquated air-traffic-control system, but the effort to build a replacement soon became riddled with problems [see photo, “Air Jam”]. By 1994, when the agency finally gave up on the project, the predicted cost had tripled, more than \$2.6 billion had been spent, and the expected delivery date had slipped by several years. Every airplane passenger who is delayed because of gridlocked skyways still feels this cancellation; the cumulative economic impact of all those delays on just the U.S. airlines (never mind the passengers) approaches \$50 billion.

Worldwide, it's hard to say how many software projects fail or how much money is wasted as a result. If you define failure as

Case Study #1

COMPANY:

Oxford Health Plans

FAILURE:

New billing system cannot keep up with expanding business, resulting in uncollected payments of \$400 million from patients and \$650 million owed to caregivers.

LOSS:

October 1997 announcement of quarterly loss triggers stock price to drop from \$68 to \$26 in one day, wiping out \$3.4 billion in corporate value. Company later pays investors \$225 million to settle lawsuits.

the total abandonment of a project before or shortly after it is delivered, and if you accept a conservative failure rate of 5 percent, then billions of dollars are wasted each year on bad software.

For example, in 2004, the U.S. government spent \$60 billion on software (not counting the embedded software in weapons systems); a 5 percent failure rate means \$3 billion was probably wasted. However, after several decades as an IT consultant, I am convinced that the failure rate is 15 to 20 percent for projects that have budgets of \$10 million or more. Looking at the total investment in new software projects—both government and corporate—over the last five years, I estimate that project failures have likely cost the U.S. economy at least \$25 billion and maybe as much as \$75 billion.

Of course, that \$75 billion doesn't reflect projects that exceed their budgets—which most projects do. Nor does it reflect projects delivered late—which the majority are. It also fails to account for the opportunity costs of having to start over once a project is abandoned or the costs of bug-ridden systems that have to be repeatedly reworked.

Then, too, there's the cost of litigation from irate customers suing suppliers for poorly implemented systems. When you add up all these extra costs, the yearly tab for failed and troubled software conservatively runs somewhere from \$60 billion to \$70 billion in the United States alone. For that money, you could launch the space shuttle 100 times, build and deploy the entire 24-satellite Global Positioning System, and develop the Boeing 777 from scratch—and still have a few billion left over.

WHY DO SOFTWARE PROJECTS FAIL SO OFTEN?

Among the most common factors:

- Unrealistic or unarticulated project goals
- Inaccurate estimates of needed resources
- Badly defined system requirements
- Poor reporting of the project's status
- Unmanaged risks
- Poor communication among customers, developers, and users
- Use of immature technology
- Inability to handle the project's complexity
- Sloppy development practices
- Poor project management
- Stakeholder politics
- Commercial pressures

Of course, IT projects rarely fail for just one or two reasons. The FBI's VCF project suffered from many of the problems listed above. Most failures, in fact, can be traced to a combination of technical, project management, and business decisions. Each dimension interacts with the others in complicated ways that exacerbate project risks and problems and increase the likelihood of failure.

Consider a simple software chore: a purchasing system that automates the ordering, billing, and shipping of parts, so that a salesperson can input a customer's order, have it automatically checked against pricing and contract requirements, and arrange to have the parts and invoice sent to the customer from the warehouse.

The requirements for the system specify four basic steps. First, there's the sales process, which creates a bill of sale. That bill is then sent through a legal process, which reviews the contractual terms and conditions of the potential sale and approves them. Third in line is the provision process, which sends out the parts contracted for, followed by the finance process, which sends out an invoice.

Let's say that as the first process, for sales, is being written, the programmers treat every order as if it were placed in the company's main location, even though the company has branches in several states and countries. That mistake, in turn, affects how tax is calculated, what kind of contract is issued, and so on.

The sooner the omission is detected and corrected, the better. It's kind of like knitting a sweater. If you spot a missed stitch right after you make it, you can simply unravel a bit of yarn and move on. But if you don't catch the mistake until the end, you may need to unravel the whole sweater just to redo that one stitch.

If the software coders don't catch their omission until final system testing—or worse, until after the system has been rolled out—the costs incurred to correct the error will likely be many times greater than if they'd caught the mistake while they were still working on the initial sales process.

And unlike a missed stitch in a sweater, this problem is much harder to pinpoint; the programmers will see only that errors are appearing, and these might have several causes. Even after the original error is corrected, they'll need to change other calculations and documentation and then retest every step.

In fact, studies have shown that software specialists spend about 40 to 50 percent of their time on avoidable rework rather than on what they call value-added work, which is basically work that's done right the first time. Once a piece of software makes it into the field, the cost of fixing an error can be 100 times as high as it would have been during the development stage.

If errors abound, then rework can start to swamp a project, like a dinghy in a storm. What's worse, attempts to fix an error often introduce new ones. It's like you're bailing out that dinghy, but you're also creating leaks. If too many errors are produced, the cost and time needed to complete the system become so great that going on doesn't make sense.

In the simplest terms, an IT project usually fails when the rework exceeds the value-added work that's been budgeted for. This is what happened to Sydney Water Corp., the largest water provider in Australia, when it attempted to introduce an automated customer information and billing system in 2002 [see box, "Case Study #2"]. According to an investigation by the Australian Auditor General, among the factors that doomed the project were inadequate planning and specifications, which in turn led to numerous change requests and significant added costs and delays. Sydney Water aborted the project midway, after spending AU \$61 million (US \$33.2 million).

All of which leads us to the obvious question: why do so many errors occur?

SOFTWARE PROJECT FAILURES have a lot in common with airplane crashes. Just as pilots never intend to crash, software developers don't aim to fail. When a commercial plane crashes, investigators look at many factors, such as the weather, maintenance records, the pilot's disposition and training, and cultural factors within the airline. Similarly, we need to look at the business environment, technical management, project management, and organizational culture to get to the roots of software failures.

Chief among the business factors are competition and the need to cut costs. Increasingly, senior managers expect IT departments to do more with less and do it faster than before; they view software projects not as investments but as pure costs that must be controlled.

Political exigencies can also wreak havoc on an IT project's schedule, cost, and quality. When Denver International Airport attempted to roll out its automated baggage-handling system, state and local political leaders held the project to one unrealistic schedule after another. The failure to deliver the system on time delayed the 1995 opening of the airport (then the largest in the United States), which compounded the financial impact manyfold.

Even after the system was completed, it never worked reliably: it chewed up baggage, and the carts used to shuttle luggage around frequently derailed. Eventually, United Airlines, the airport's main tenant, sued the system contractor, and the episode became a testament to the dangers of political expediency.

A lack of upper-management support can also damn an IT undertaking. This runs the gamut from failing to allocate enough money and manpower to not clearly establishing the IT project's relationship to the organization's business. In 2000, retailer Kmart Corp., in Troy, Mich., launched a \$1.4 billion IT modernization effort aimed at linking its sales, marketing, supply, and logistics systems, to better compete with rival Wal-Mart Corp., in Bentonville, Ark. Wal-Mart proved too formidable, though, and 18 months later, cash-strapped Kmart cut back on modernization, writing off the \$130 million it had already invested in IT. Four months later, it declared bankruptcy; the company continues to struggle today.

Frequently, IT project managers eager to get funded resort to a form of liar's poker, overpromising what their project will do, how much it will cost, and when it will be completed. Many, if not most, software projects start off with budgets that are too small. When that happens, the developers have to make up for the shortfall somehow, typically by trying to increase productivity, reducing the scope of the effort, or taking risky shortcuts in the review and testing phases. These all increase the likelihood of error and, ultimately, failure.

A state-of-the-art travel reservation system spearheaded by a consortium of Budget Rent-A-Car, Hilton Hotels, Marriott, and AMR, the parent of American Airlines, is a case in point. In 1992, three and a half years and \$165 million into the project, the group abandoned it, citing two main reasons: an overly optimistic development schedule and an underestimation of the technical difficulties involved. This was the same group that had earlier built the hugely successful Sabre reservation system, proving that past performance is no guarantee of future results.

AFTER CRASH INVESTIGATORS CONSIDER the weather as a factor in a plane crash, they look at the airplane itself. Was there something in the plane's design that caused the crash? Was it carrying too much weight?

In IT project failures, similar questions invariably come up regarding the project's technical components: the hardware and software used to develop the system and the development prac-

tices themselves. Organizations are often seduced by the siren song of the technological imperative—the uncontrollable urge to use the latest technology in hopes of gaining a competitive edge. With technology changing fast and promising fantastic new capabilities, it is easy to succumb. But using immature or untested technology is a sure route to failure.

In 1997, after spending \$40 million, the state of Washington shut down an IT project that would have processed driver's licenses and vehicle registrations. Motor vehicle officials admitted that they got caught up in chasing technology instead of concentrating on implementing a system that met their requirements. The IT debacle that brought down FoxMeyer Drug a year earlier also stemmed from adopting a state-of-the-art resource-planning system and then pushing it beyond what it could feasibly do.

A project's sheer size is a fountainhead of failure. Studies indicate that large-scale projects fail three to five times more often than small ones. The larger the project, the more complexity there is in both its static elements (the discrete pieces of software, hardware, and so on) and its dynamic elements (the couplings and interactions among hardware, software, and users; connections to other systems; and so on). Greater complexity increases the possibility of errors, because no one really understands all the interacting parts of the whole or has the ability to test them.

Sobering but true: it's impossible to thoroughly test an IT system of any real size. Roger S. Pressman pointed out in his book *Software Engineering*, one of the classic texts in the field, that "exhaustive testing presents certain logistical problems....Even a small 100-line program with some nested paths and a single loop executing less than twenty times may require 10 to the power of 14 possible paths to be executed." To test all of those 100 trillion paths, he noted, assuming each could be evaluated in a millisecond, would take 3170 years.

All IT systems are intrinsically fragile. In a large brick building, you'd have to remove hundreds of strategically placed bricks to make a wall collapse. But in a 100 000-line software program, it takes only one or two bad lines to produce major problems. In 1991, a portion of AT&T's telephone network went out, leaving 12 million subscribers without service, all because of a single mistyped character in one line of code.

Sloppy development practices are a rich source of failure, and they can cause errors at any stage of an IT project. To help organizations assess their software-development practices, the U.S. Software Engineering Institute, in Pittsburgh, created the Capability Maturity Model, or CMM. It rates a company's practices against five levels of increasing maturity. Level 1 means the organization is using ad hoc and possibly chaotic development practices. Level 3 means the company has characterized its practices and now understands them. Level 5 means the organization quantitatively understands the variations in the processes and practices it applies.

As of January, nearly 2000 government and commercial organizations had voluntarily reported CMM levels. Over half acknowledged being at either level 1 or 2, 30 percent were at level 3, and only 17 percent had reached level 4 or 5. The percentages are even more dismal when you realize that this is a self-selected group; obvi-

ously, companies with the worst IT practices won't subject themselves to a CMM evaluation. (The CMM is being superseded by the CMM-Integration, which aims for a broader assessment of an organization's ability to create software-intensive systems.)

Immature IT practices doomed the U.S. Internal Revenue Service's \$4 billion modernization effort in 1997, and they have continued to plague the IRS's current \$8 billion modernization. It may just be intrinsically impossible to translate the tax code into software code—tax law is complex and based on often-vague legislation, and it changes all the time. From an IT developer's standpoint, it's a requirements nightmare. But the IRS hasn't been helped by open hostility between in-house and outside programmers, a laughable underestimation of the work involved, and many other bad practices.

THE PILOT'S ACTIONS JUST BEFORE a plane crashes are always of great interest to investigators. That's because the pilot is the ultimate decision-maker, responsible for the safe operation of

Case Study #2

COMPANY:

Sydney Water Corp.

FAILURE:

Project to automate customer information and billing for Australia's largest water provider is canceled in 2002, due to inadequate planning, numerous change requests, and cost and schedule overruns.



COST:

\$33.2 million

the craft. Similarly, project managers play a crucial role in software projects and can be a major source of errors that lead to failure.

Back in 1986, the London Stock Exchange decided to automate its system for settling stock transactions. Seven years later, after spending \$600 million, it scrapped the Taurus system's development, not only because the design was excessively complex and cumbersome but also because the management of the project was, to use the word of one of its own senior managers, "delusional." As investigations revealed, no one seemed to want to know the true status of the project, even as more and more problems appeared, deadlines were missed, and costs soared [see box, "Case Study #3"].

The most important function of the IT project manager is to allocate resources to various activities. Beyond that, the project manager is responsible for project planning and estimation, control, organization, contract management, quality management, risk management, communications, and human resource management.

Bad decisions by project managers are probably the single greatest cause of software failures today. Poor technical management, by contrast, can lead to technical errors, but those can generally be isolated and fixed. However, a bad project management decision—

such as hiring too few programmers or picking the wrong type of contract—can wreak havoc. For example, the developers of the doomed travel reservation system claim that they were hobbled in part by the use of a fixed-price contract. Such a contract assumes that the work will be routine; the reservation system turned out to be anything but.

Project management decisions are often tricky precisely because they involve tradeoffs based on fuzzy or incomplete knowledge. Estimating how much an IT project will cost and how long it will take is as much art as science. The larger or more novel the project, the less accurate the estimates. It's a running joke in the industry that IT project estimates are at best within 25 percent of their true value 75 percent of the time.

There are other ways that poor project management can hasten a software project's demise. A study by the Project Management Institute, in Newton Square, Pa., showed that risk management is the least practiced of all project management disciplines across all industry sectors, and nowhere is it more infrequently applied than

way. The same attitudes existed among those responsible for the travel reservation system, the London Stock Exchange's Taurus system, and the FAA's air-traffic-control project—all indicative of organizational cultures driven by fear and arrogance.

A recent report by the National Audit Office in the UK found numerous cases of government IT projects' being recommended not to go forward yet continuing anyway. The UK even has a government department charged with preventing IT failures, but as the report noted, more than half of the agencies the department oversees routinely ignore its advice. I call this type of behavior irrational project escalation—the inability to stop a project even after it's obvious that the likelihood of success is rapidly approaching zero. Sadly, such behavior is in no way unique.

IN THE FINAL ANALYSIS, big software failures tend to resemble the worst conceivable airplane crash, where the pilot was inexperienced but exceedingly rash, flew into an ice storm in an untested aircraft, and worked for an airline that gave lip service to safety while cutting back on training and maintenance. If you read the investigator's report afterward, you'd be shaking your head and asking, "Wasn't such a crash inevitable?"

So, too, the reasons that software projects fail are well known and have been amply documented in countless articles, reports, and books [see sidebar, To Probe Further]. And yet, failures, near-failures, and plain old bad software continue to plague us, while practices known to avert mistakes are shunned. It would appear that getting quality software on time and within budget is not an urgent priority at most organizations.

It didn't seem to be at Oxford Health Plans Inc., in Trumbull, Conn., in 1997. The company's automated billing system was vital to its bottom line, and yet senior managers there were more interested in expanding Oxford's business



Case Study #3

COMPANY:

London Stock Exchange

in the IT industry. Without effective risk management, software developers have little insight into what may go wrong, why it may go wrong, and what can be done to eliminate or mitigate the risks. Nor is there a way to determine what risks are acceptable, in turn making project decisions regarding tradeoffs almost impossible.

Poor project management takes many other forms, including bad communication, which creates an inhospitable atmosphere that increases turnover; not investing in staff training; and not reviewing the project's progress at regular intervals. Any of these can help derail a software project.

THE LAST AREA THAT INVESTIGATORS look into after a plane crash is the organizational environment. Does the airline have a strong safety culture, or does it emphasize meeting the flight schedule above all? In IT projects, an organization that values openness, honesty, communication, and collaboration is more apt to find and resolve mistakes early enough that rework doesn't become overwhelming.

If there's a theme that runs through the tortured history of bad software, it's a failure to confront reality. On numerous occasions, the U.S. Department of Justice's inspector general, an outside panel of experts, and others told the head of the FBI that the VCF system was impossible as defined, and yet the project continued any-

FAILURE:

Effort to design new stock settlement system is scrapped in 1993, after seven years, because of overly complex and cumbersome design and poor project management.

COST:

\$600 million

than in ensuring that its billing system could meet its current needs [see box, "Case Study #1"]. Even as problems arose, such as invoices' being sent out months late, managers paid little attention. When the billing system effectively collapsed, the company lost tens of millions of dollars, and its stock dropped from \$68 to \$26 per share in one day, wiping out \$3.4 billion in corporate value. Shareholders brought lawsuits, and several government agencies investigated the company, which was eventually fined \$3 million for regulatory violations.

Even organizations that get burned by bad software experiences seem unable or unwilling to learn from their mistakes. In a 2000 report, the U.S. Defense Science Board, an advisory body to the Department of Defense, noted that various studies commissioned by the DOD had made 134 recommendations for improving its software development, but only 21 of those recommendations had been acted on. The other 113 were still valid, the board noted, but were being ignored, even as the DOD complained about the poor state of defense software development!

Some organizations do care about software quality, as the experience of the software development firm Praxis High Integrity Systems, in Bath, England, proves. Praxis demands that its customers be committed to the project, not only financially, but as active participants in the IT system's creation. The company also spends a

tremendous amount of time understanding and defining the customer's requirements, and it challenges customers to explain what they want and why. Before a single line of code is written, both the customer and Praxis agree on what is desired, what is feasible, and what risks are involved, given the available resources.

After that, Praxis applies a rigorous development approach that limits the number of errors. One of the great advantages of this model is that it filters out the many would-be clients unwilling to accept the responsibility of articulating their IT requirements and spending the time and money to implement them properly. [See "The Exterminators," in this issue.]

SOME LEVEL OF SOFTWARE FAILURE will always be with us. Indeed, we need true failures—as opposed to avoidable blunders—to keep making technical and economic progress. But too many of the failures that occur today are avoidable. And as our society comes to rely on IT systems that are ever larger, more integrated, and more expensive, the cost of failure may become disastrously high.

Even now, it's possible to take bets on where the next great software debacle will occur. One of my leading candidates is the IT systems that will result from the U.S. government's American Health Information Community, a public-private collaboration that seeks to define data standards for electronic medical records. The idea is that once standards are defined, IT systems will be built to let medical professionals across the country enter patient records digitally, giving doctors, hospitals, insurers, and other health-care specialists instant access to a patient's complete medical history. Health-care experts believe such a system of systems will improve patient care, cut costs by an estimated \$78 billion per year, and reduce medical errors, saving tens of thousands of lives.

But this approach is a mere pipe dream if software practices and failure rates remain as they are today. Even by the most optimistic estimates, to create an electronic medical record system will require 10 years of effort, \$320 billion in development costs, and \$20 billion per year in operating expenses—assuming that there are no failures, overruns, schedule slips, security issues, or shoddy software. This is hardly a realistic scenario, especially because most IT experts consider the medical community to be the least computer-savvy of all professional enterprises.

Patients and taxpayers will ultimately pay the price for the development, or the failure, of boondoggles like this. Given today's IT practices, failure is a distinct possibility, and it would be a loss of unprecedented magnitude. But then, countries throughout the world are contemplating or already at work on many initiatives of similar size and impact—in aviation, national security, and the military, among other arenas.

Like electricity, water, transportation, and other critical parts of our infrastructure, IT is fast becoming intrinsic to our daily existence. In a few decades, a large-scale IT failure will become more than just an expensive inconvenience: it will put our way of life at risk. In the absence of the kind of industrywide changes that will mitigate software failures, how much of our future are we willing to gamble on these enormously costly and complex systems?

We already know how to do software well. It may finally be time to act on what we know. ■

ABOUT THE AUTHOR

ROBERT N. CHARETTE is president of ITABHI Corp., a risk-management consultancy in Spotsylvania, Va. An IEEE member, he is the author of several books on risk management and chair of the ISO/IEEE committee revising the I6085 standard on software and systems engineering risk management.

TO PROBE FURTHER: SOFTWARE SPECIAL REPORT

WHO KILLED THE VIRTUAL CASE FILE?

For background on the FBI, read Ronald Kessler's *The Bureau: The Secret History of the FBI* (St. Martin's Press, 2002).

Track FBI CIO Zalmi Azmi's attempts to drag the bureau into the 21st century at http://www.fbi.gov/hq/ocio/ocio_home.htm.

Read the record testimony of Arnold Punaro, executive vice president and general manager of Science International Applications Corp.—prepared for the 3 February 2005 U.S. Senate hearing on the Virtual Case File (VCF)—at <http://www.saic.com/cover-archive/law/trilogy.html>.

The Government Accountability Office (GAO) and the Department of Justice's Office of the Inspector General (OIG) warned Congress of serious problems with the VCF. Start exploring GAO documents at <http://www.gao.gov>. For the September 2003 report "FBI Needs an Enterprise Architecture to Guide Its Modernization Activities," search on report number GAO-03-959.

For OIG reports related to the FBI's IT systems, start digging at <http://www.usdoj.gov/oig/reports/FBI>. One must-read: "The Federal Bureau of Investigation's Management of the Trilogy Information Technology Modernization Project" at <http://www.usdoj.gov/oig/reports/FBI/a0507/index.htm>.

The graybeards evaluate the VCF in a National Research Council report at http://www7.nationalacademies.org/cstb/pub_fbi.html.

THE EXTERMINATORS

For an introduction to formal software-design methods, see Jeannette M. Wing's "A Specifier's Introduction to Formal Methods," in *Computer*, September 1990, Vol. 23, no. 9. "An Invitation to Formal Methods," by Jonathan P. Bowen et al. in *Computer*, April 1996, Vol. 29, no. 4, discusses how to make formal methods more widely used.

For many examples of formal methods used in industry, see "Formal Methods: State of the Art and Future Directions" at http://www-2.cs.cmu.edu/~emc/papers/Invited%20Journal%20Articles/state_art_future.pdf.

Praxis High Integrity Systems offers a number of technical articles and presentations at <http://www.praxis-his.com/publications>.

WHY SOFTWARE FAILS

Many good books have been written about the causes of software failure, including Frederick P. Brooks Jr.'s *The Mythical Man-Month: Essays on Software Engineering*, 20th Anniversary Edition (Addison-Wesley, 1995); Kweku Ewusi-Mensah's *Software Development Failures* (MIT Press, 2003); Stephen Flowers's *Software Failure: Management Failure* (John Wiley & Sons, 1996); Robert L. Glass's *Software Runaways* (Prentice Hall PTR, 1998); Capers Jones's *Patterns of Software Systems Failure and Success* (International Thomson Computer Press, 1996); Peter Neumann's *Computer Related Risks* (Addison Wesley, 1995); Ivars Peterson's *Fatal Defect* (Vintage Books, 1996); Susan A. Sherer's *Software Failure Risk* (Plenum, 1992); and Edward Yourdon's *Death March* (Prentice Hall, 1997).

The online "Forum on Risks to the Public in Computers and Related Systems" is at <http://catless.ncl.ac.uk/risks>.

Frederick P. Brooks Jr.'s classic paper "No Silver Bullet: Essence and Accidents of Software Engineering" appeared in *Computer*, April 1987, pp. 10–19.

IT Alignment

1. IT Governance Definition and Significance
2. Framework for IT Governance
3. Operating Model & Maturity of Enterprise Architecture

The Operating Model

4. What is the operating model of a company?
5. How does the operating model and maturity of enterprise architecture relate to each other.
6. The strategic implications of a specific mode of enterprise architecture.

Operating Model Definition

Operating model is the degree to which a company standardizes its core processes and integrates its data.

Elements of the Operating Model:

1. Process Standardization

Standardization is the degree to which a company wishes to standardize its core processes. Standardization means perform activities in a similar way across all units of the organization. Therefore, standardization reduces variation in organization's business processes.

Standardization requires having:

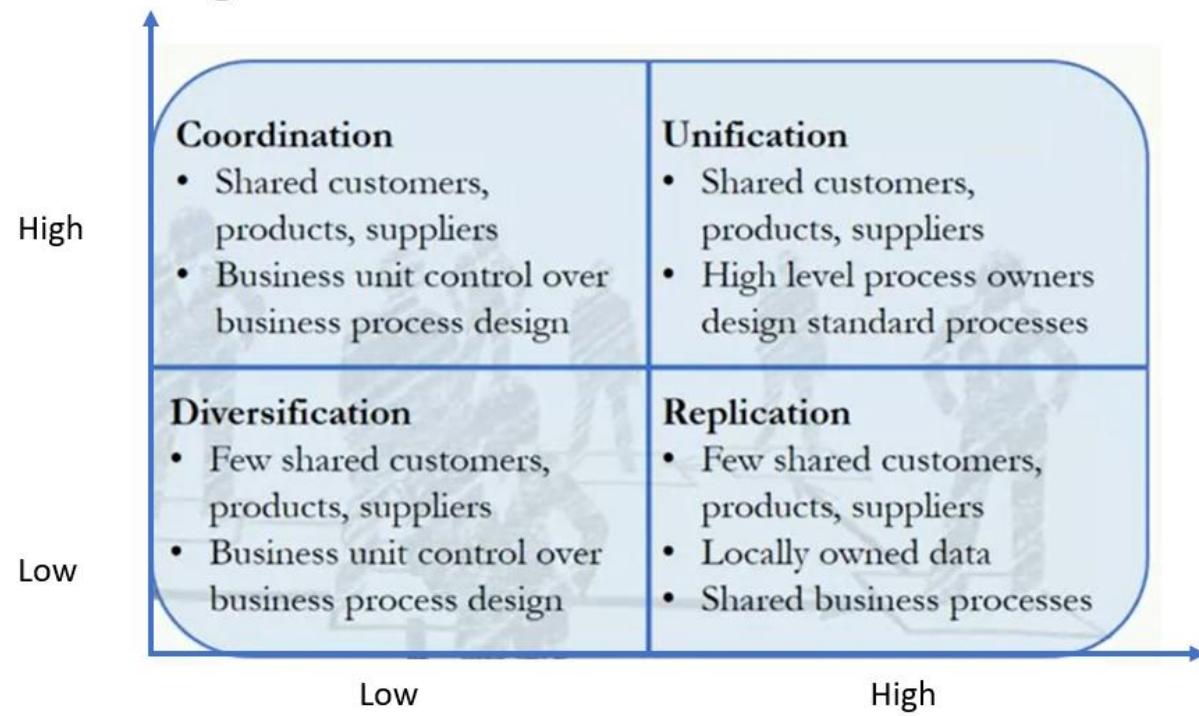
- Central control over business processes.
- Central body that coordinates how a specific process will be performed.

2. Data Integration

Data integration is the sharing of data across the organization's business units including:

- Customer Information
- Product Information
- Supplier

Data Integration



Process Standardization



Information Systems Management: Key Principles and Practices

Course 1: IS/IT Governance
Module 1: IT Alignment

Module Overview and Goals

1. Define IT Governance and Understand its Significance
2. A Framework for IT Governance
3. Operating Model and the Maturity of Enterprise Architecture

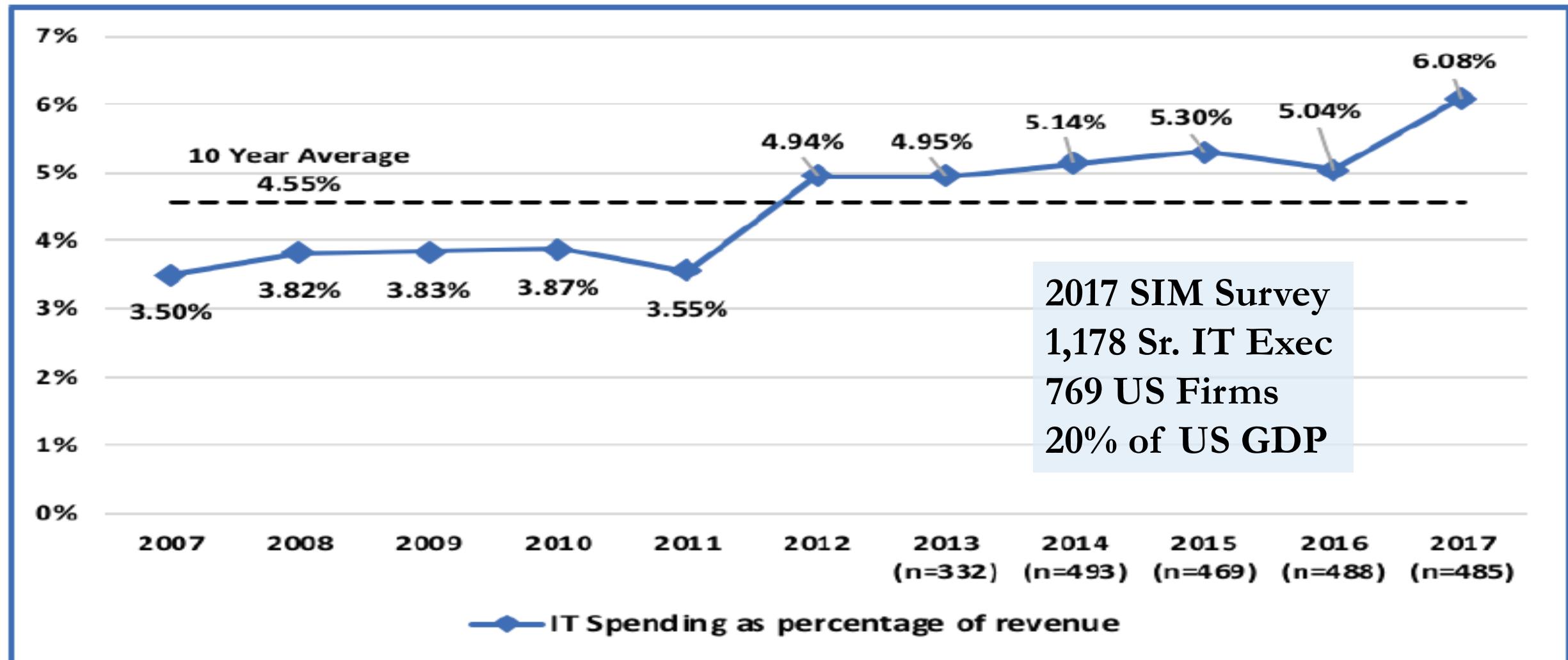
Figure 1: Average IT Spending as a Percentage of Revenue, 2007-17

Figure 5: Percentage of Organizations Increasing, Not Changing and Decreasing IT Budgets from Prior Year, 2007-17

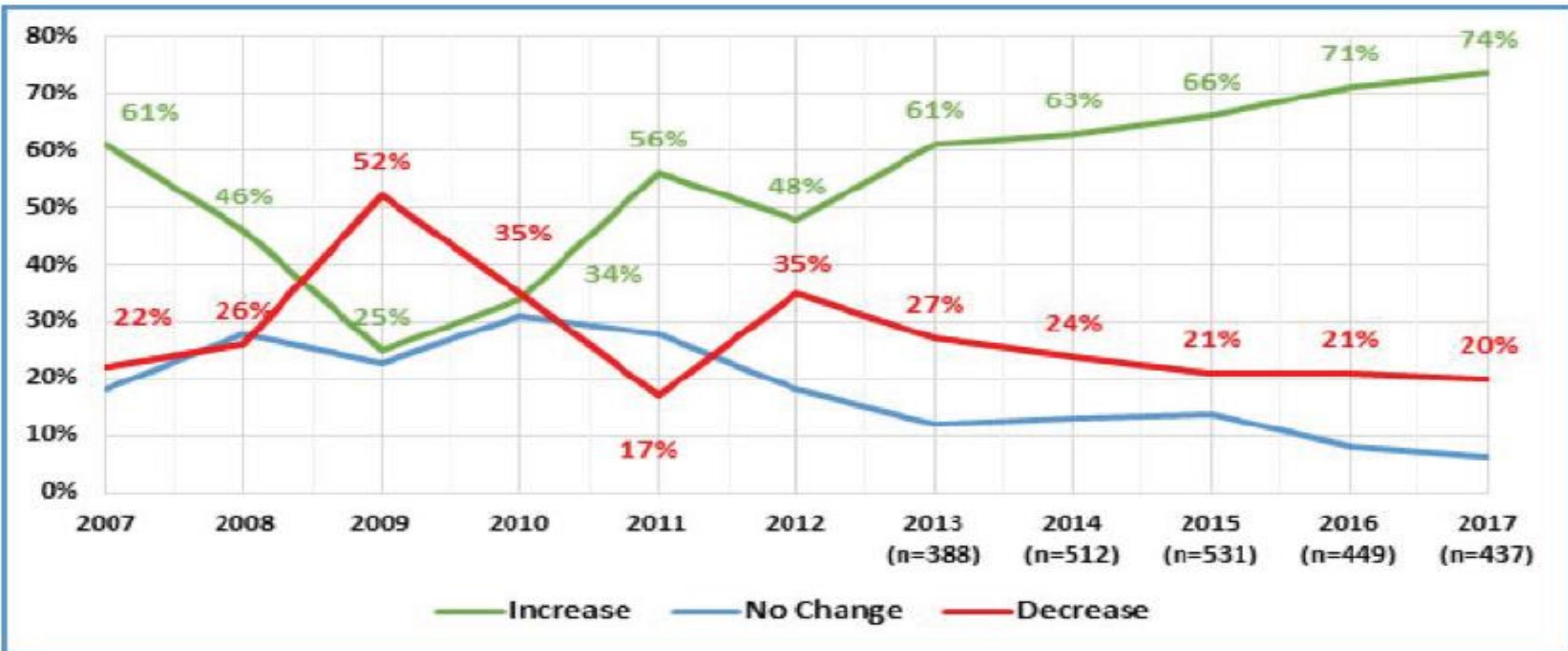


Exhibit 1

Performance of IT Projects

The performance of different types of IT projects varies significantly.

%, projects >\$15 million, in 2010 dollars

Project type	Average cost overrun	Average schedule overrun	Average benefits shortfall
Software	66	33	17
Nonsoftware	43	3.6	133
Total	45	7	56

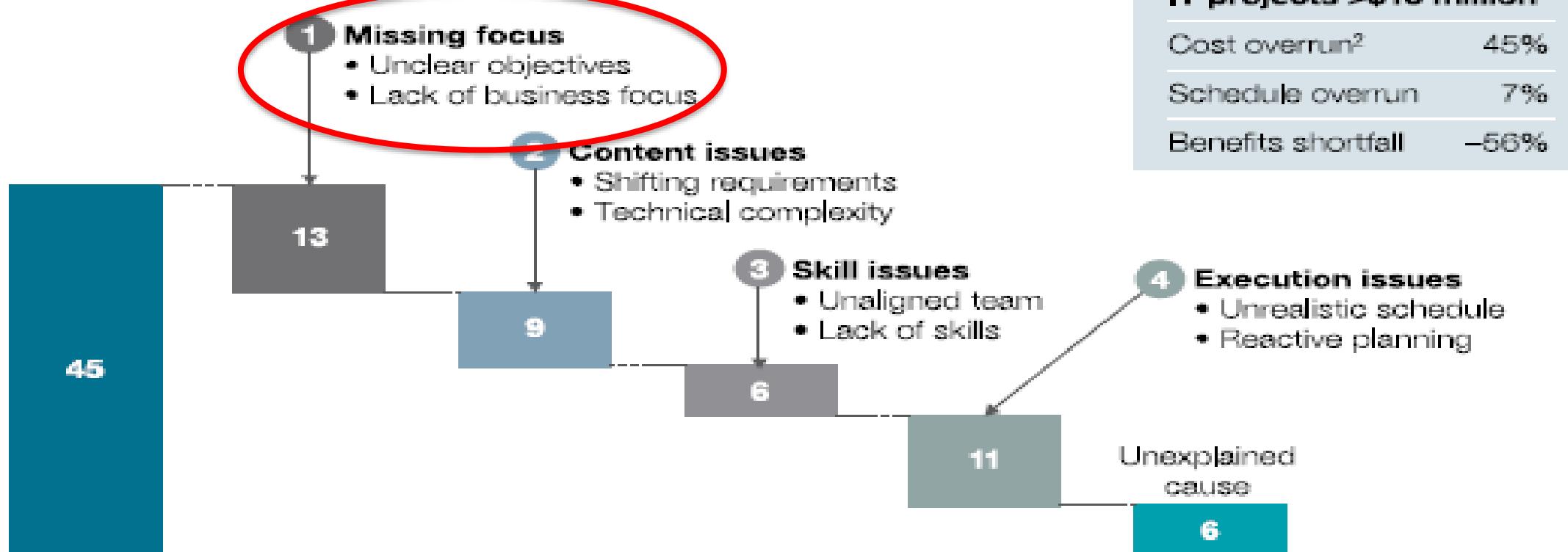
Source: McKinsey–Oxford study on reference-class forecasting for IT projects

Drivers of IT Project Performance

Exhibit 2

IT executives identify 4 groups of issues that cause most project failures.

Rough cost-overrun disaggregation, %



¹With cost overrun, in 2010 dollars.

²Cost increase over regular cost.

Source: McKinsey–Oxford study on reference-class forecasting for IT projects

Table 1: Organizations' Top 10 Most Important IT Management Issues, 2007-17

IT Management Concerns/Issues ^a	2017 (% Selecting)	2016	2015	2014	2013	2012	2011	2010	2009	2008	2007
n (unique organizations)	769	801	785	717	484	195	275	172	243	291	112
Security/Cybersecurity/Privacy ^b	1 (41.9%)	2	2	2	7	9	8	9	9	8	6
Alignment of IT with the Business	2 (37.3%)	1	1	1	1	2	1	3	2	1	2
Data Analytics/Data Management	3 (25.4%)										
Compliance and Regulations	4 (20.7%)	12									
Cost Reduction/Cost Controls (IT) ^c	5 (20.0%)	6	10	9	4						
Cost Reduction/Controls (Business) ^c	6 (19.9%)	7	8	17	5	5	10	8	5	7	4
Innovation	7 (19.5%)	3	4	8							
Digital Transformation	8 (18.7%)										
Agility/Flexibility (Business) ^d	9 (17.8%)	5	9	3	2	3	2	2	3	13	17
Agility/Flexibility (IT) ^d	10 (16.6%)	4	7	13							

^aBlank cells, unless otherwise noted, indicate that the issue was not included that year.

^bIn previous years, "Security/Cybersecurity/Privacy" was "Security/Privacy."

^c"Business Cost Reduction/Controls" and "IT Cost Reduction/Controls" were merged into a "Cost Reduction/Controls" category with Business and IT selections. "Business Cost Controls" was combined with "Business Productivity" in prior years.

^d"Business Agility/Flexibility" and "IT Agility" were merged into an "Agility/Flexibility" category with Business and IT selections in 2015.

"Agility/Flexibility (IT)" was "Architecture Agility" in 2008.

Definition of IT Governance

Institute of Internal Auditors (IIA)

“The leadership, organizational structures, and processes that ensure that the enterprise's information technology supports the organization's strategies and objectives.”

Gartner Inc., IT

“The set of processes that ensure the effective and efficient use of IT in enabling an organization to achieve its goals.”



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Image Credits

<u>Image/Source</u>	<u>Copyright</u>	<u>Location</u>	<u>Notes</u>
Figure 1: Average IT Spending as a Percentage of Revenue, 2007-17	Kappelman, L., Johnson, V., McLean, E., & Maurer, C. (2018). The 2017 SIM IT Issues and Trends Study. <i>MISQ Exec</i> , 17(1), 53-88.	Slide 3	
Figure 5: Percentage of Organizations Increasing, Not Changing and Decreasing IT Budgets from Prior Year, 2007-17	Kappelman, L., Johnson, V., McLean, E., & Maurer, C. (2018). The 2017 SIM IT Issues and Trends Study. <i>MISQ Exec</i> , 17(1), 53-88.	Slide 4	
Exhibit 1: The Performance of Different Types of IT Projects Varies Significantly	Bloch, Michael, Sven Blumberg, and Jürgen Laartz. "Delivering large-scale IT projects on time, on budget, and on value." <i>Harvard Business Review</i> (2012).	Slide 5	
Exhibit 2: IT Executives Identify 4 Groups of Issues That Cause Most Project Failures	Bloch, Michael, Sven Blumberg, and Jürgen Laartz. "Delivering large-scale IT projects on time, on budget, and on value." <i>Harvard Business Review</i> (2012).	Slide 6	
Table 1: Organizations' Top 10 Most Important IT Management Issues, 2007-17	Kappelman, L., Johnson, V., McLean, E., & Maurer, C. (2018). The 2017 SIM IT Issues and Trends Study. <i>MISQ Exec</i> , 17(1), 53-88.	Slide 7	

Definition of IT Governance

The Institute of internal Auditors defines IT Governance as the leadership organizational structures and processes that ensure the enterprises information technology supports the organization strategies and objectives.

leadership

leadership includes:

- (i) the directors on the board that are responsible for governing IT,
- (ii) the CIO or any other IT executive on the board that is responsible for governing IT,
- (iii) board level committees that are there to monitor and govern IT.

What does leadership do?

Leadership has two roles: (i) monitoring or oversight and (ii) providing their expertise to the organization.

(i) Monitoring or Oversight

This includes asking questions like:

- "Are we making good investments in IT?"
- Is IT contributing to organizational performance?
- Are we complying with all the regulations that go along with IT?"

(ii) Providing expertise to the organization

This includes providing expertise to make sure that the organization understands the role of IT in their industry and takes advantage of IT opportunities. The board is also responsible for providing preferential access to external IT providers. It's very natural for any organization to lack some expertise. So, it is the role of the IT Governance Board to get access to external service provider and make it available to the organization.

What happens if the leadership fails to perform its role?

Michael Benrock and his co-author conducted a study that examined 110 operational failures in financial services firms in the US. They found that

- (i) whenever there is an IT failure at a large US financial services firm, the stock price of the firm goes down, and
- (ii) the drop in stock price is associated with changes on the board. What are the changes on the board?

The **first change** is there is an increase in number of directors with IT competence. This indicates that an IT failure was an IT Governance failure, which means the leadership did not have the expertise to provide the oversight that they needed to provide. One resolution to this problem is to add more directors to the leadership so that the organization is able to provide the oversight that is needed. The **second change** is that there is a turnover in the CIO.

or the CTO role. Therefore, when an IT failure occurs, stock price goes down and the CIO or the CTO is replaced.

Example: Target

Target is the largest discount retailer in the US after Walmart.

On December 19th, 2013, Target disclosed a breach of 40 million credit card accounts over a three week period before Christmas.

On January 10th, 2014, the company said, "Hackers also stole personal information from as many as 70 million customers."

On March 2014, Target shares were down 6% since the breach was disclosed.

This illustrates that there was an IT failure, the IT failure was associated with a significant drop in the market price, significant drop in the share price of the company.

On March sixth, 2014, Target's CIO resigned.

On May 5th, 2014, Target's CEO and Chairman resigned amid fallout from the database.

Significance of IT Governance

IT Governance is associated with IT-Business alignment and IT-Business alignment is associated with firm performance.

IT-business alignment leads to improved firm performance in terms of:

- (i) Financial performance measured as improvement in Return On Assets and Return On Equity,
- (ii) Operational excellence, which is measured efficiency of processes in the organization compared to the efficiency in competitors' processes
- (iii) Customer satisfaction measured as how satisfied the firm's customers when compared to competitors' customers.

Measuring IT Governance

Two methods for measuring IT Governance:

- (i) Steering committee to monitor and govern IT investments and IT plans
- (ii) IT Investment Prioritization Process to evaluate all the IT ideas and proposals and then choose the best ones.

IT Governance is associated with IT-business alignment. IT-business alignment is measured as alignment between business and IT strategy alignment for products, processes, and markets. IT business alignment was measured as process alignment, that is, business and IT align in terms of how different processes will work and finally alignment between business and IT about which markets to enter.



Information Systems Management: Key Principles and Practices

Course 1: IS/IT Governance
Module 1.2: IT Alignment

Module Overview and Goals

1. Define IT Governance and Understand its Significance
2. A Framework for IT Governance
3. Operating Model and the Maturity of Enterprise Architecture

Lesson Overview and Goals

- A Framework for IT Governance
 1. Explore key decisions

Framework for IT Governance

Institute of Internal Auditors (IIA)

“The leadership, organizational structures, and processes that ensure that the enterprise's information technology supports the organization's strategies and objectives.”

1. What decisions need to be made?
2. Who makes those decisions?
3. How they make those decisions?

Key Decisions

Decisions	Description
IT Principles	<p>Enterprise level objectives for IT – what is the role of IT.</p> <p>Business Principle: Economy of Scale and Operational Efficiency <-></p> <p>IT Principle: Standardized Business Processes</p>
IT Architecture	<p>The organizing logic for process, applications, data, and infrastructure. The requirements for process standardization and data integration.</p>
IT Infrastructure	<p>Servers, desktops, networks, databases, and common applications. What shared infrastructure service needs to be provided.</p>
Business Applications	<p>Software application that directly serve business needs. Which business requirements need IT applications at the corporate and BU level.</p>
IT Investment	<p>How much to invest in IT? Which IT projects to fund?</p>

Key Decisions

Decisions	Description
IT Principles	<p>Enterprise level objectives for IT – what is the role of IT.</p> <p>Business Principle: Economy of Scale and Operational Efficiency <-></p> <p>IT Principle: Standardized Business Processes</p>
IT Architecture	<p>The organizing logic for process, applications, data, and infrastructure. The requirements for process standardization and data integration.</p>
IT Infrastructure	<p>Servers, desktops, networks, databases, and common applications. What shared infrastructure service needs to be provided.</p>
Business Applications	<p>Software application that directly serve business needs. Which business requirements need IT applications at the corporate and BU level.</p>
IT Investment	<p>How much to invest in IT? Which IT projects to fund?</p>

Key Decisions

Decisions	Description
IT Principles	<p>Enterprise level objectives for IT – what is the role of IT.</p> <p>Business Principle: Economy of Scale and Operational Efficiency <-></p> <p>IT Principle: Standardized Business Processes</p>
IT Architecture	<p>The organizing logic for process, applications, data, and infrastructure. The requirements for process standardization and data integration.</p>
IT Infrastructure	<p>Servers, desktops, networks, databases, and common applications. What shared infrastructure service needs to be provided.</p>
Business Applications	<p>Software application that directly serve business needs. Which business requirements need IT applications at the corporate and BU level.</p>
IT Investment	<p>How much to invest in IT? Which IT projects to fund?</p>

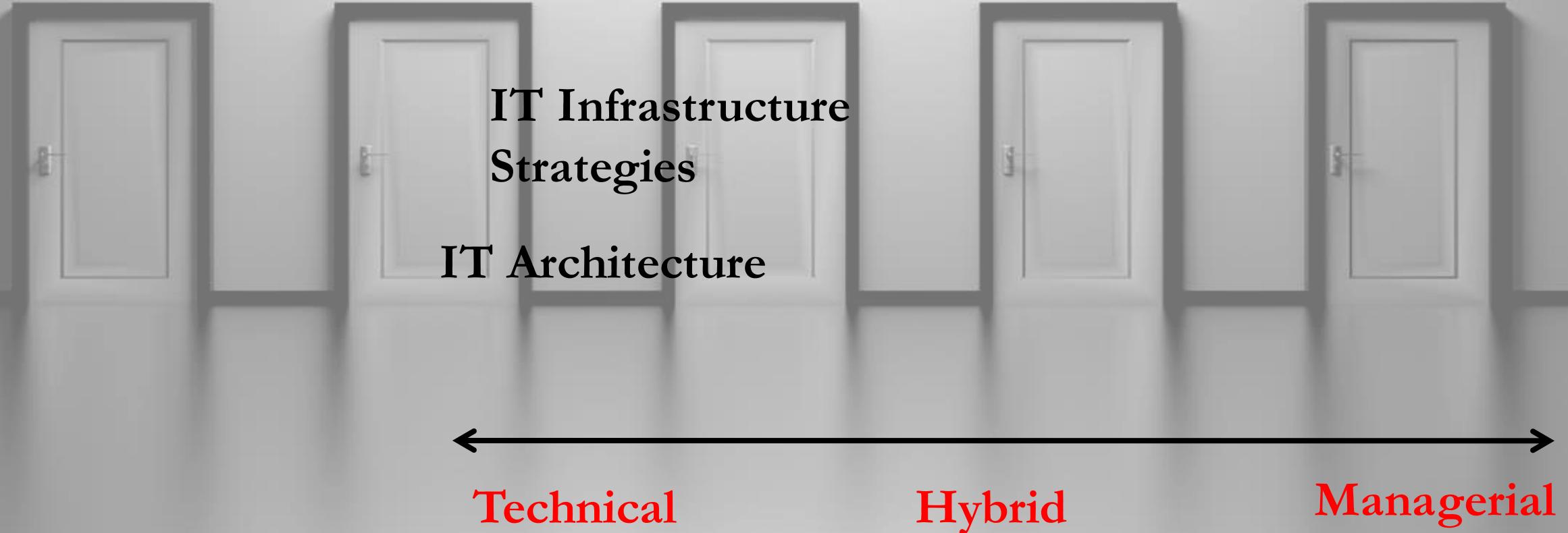
Key Decisions

Decisions	Description
IT Principles	<p>Enterprise level objectives for IT – what is the role of IT.</p> <p>Business Principle: Economy of Scale and Operational Efficiency <-></p> <p>IT Principle: Standardized Business Processes</p>
IT Architecture	<p>The organizing logic for process, applications, data, and infrastructure. The requirements for process standardization and data integration.</p>
IT Infrastructure	<p>Servers, desktops, networks, databases, and common applications. What shared infrastructure service needs to be provided.</p>
Business Applications	<p>Software application that directly serve business needs. Which business requirements need IT applications at the corporate and BU level.</p>
IT Investment	<p>How much to invest in IT? Which IT projects to fund?</p>

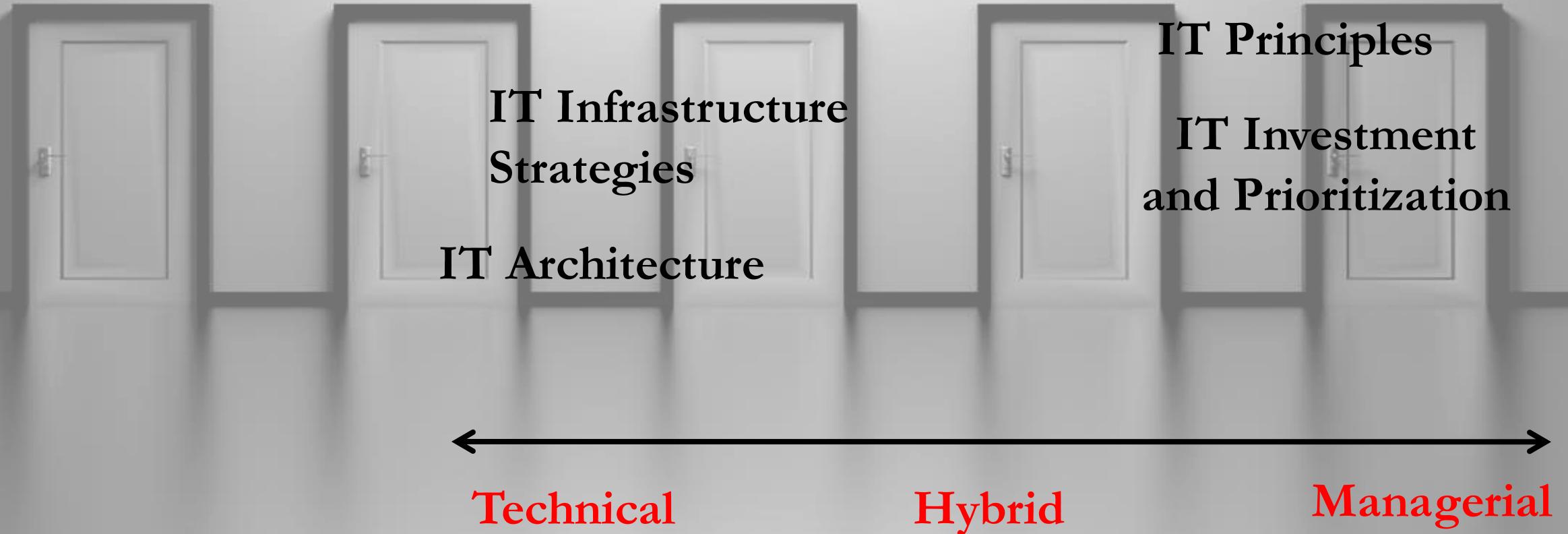
Key Decisions

Decisions	Description
IT Principles	<p>Enterprise level objectives for IT – what is the role of IT.</p> <p>Business Principle: Economy of Scale and Operational Efficiency <-></p> <p>IT Principle: Standardized Business Processes</p>
IT Architecture	<p>The organizing logic for process, applications, data, and infrastructure. The requirements for process standardization and data integration.</p>
IT Infrastructure	<p>Servers, desktops, networks, databases, and common applications. What shared infrastructure service needs to be provided.</p>
Business Applications	<p>Software application that directly serve business needs. Which business requirements need IT applications at the corporate and BU level.</p>
IT Investment	<p>How much to invest in IT? Which IT projects to fund?</p>

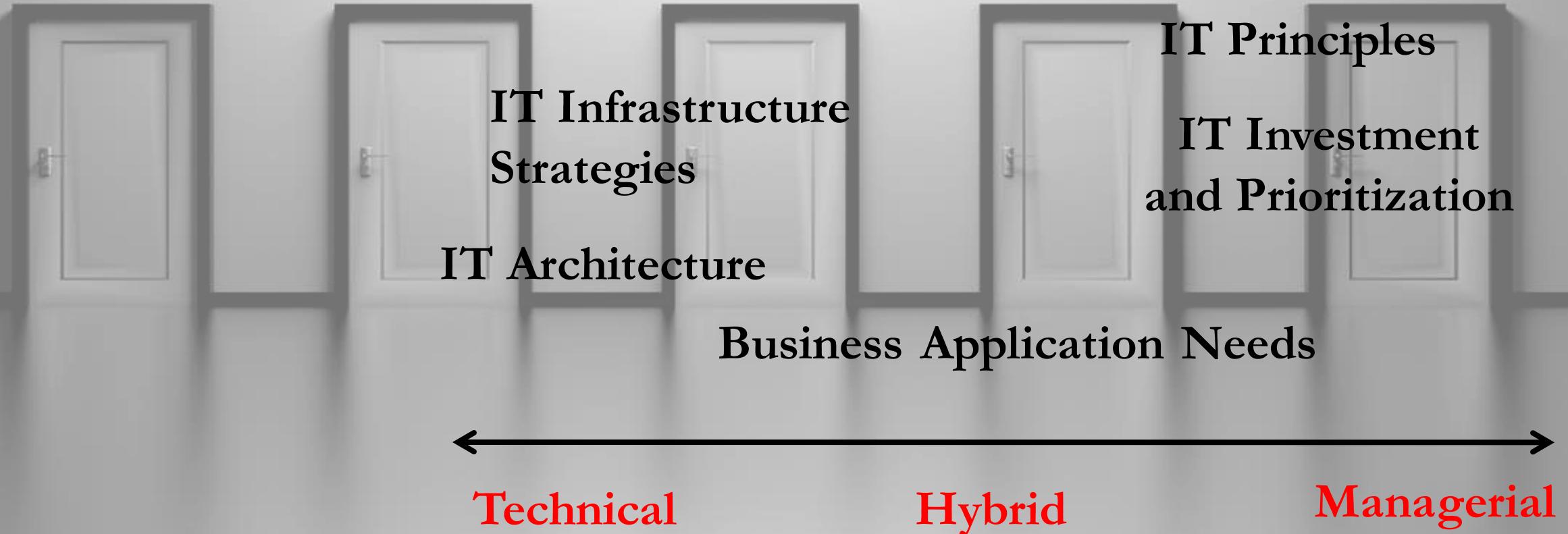
What: Five IT Decision Types



What: Five IT Decision Types



What: Five IT Decision Types





CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Image Credits

<u>Image/Source</u>	<u>Copyright</u>	<u>Location</u>	<u>Notes</u>
Key Decisions, Who Makes the Decisions	Weill, P., & Ross, J. (2005). A matrixed approach to designing IT governance. <i>MIT Sloan Management Review</i> , 46(2), 26.	Slide 5-9	

Key IT Questions

If you are in the leadership position in IT, and are responsible for IT governance, what questions should you be asking?

1. How will IT change the basis of competition in our industry?

- Who are our emerging competitors?
- How is IT helping us win against traditional and new competitors?
- How can we use IT to enter new markets?

If you are in the leadership position in your company and are responsible for IT governance, you should have a good understanding of what is the role of IT in your industry, and what are you doing to take advantage of new opportunities?

2. Do our business plans reflect the full potential of IT to improve performance?

Does IT support operational efficiency as well as strategic agility?

That is, you should be asking the question, is IT doing everything to improve the operational efficiency of the organization, as well as providing opportunities to differentiate and innovate.

3. Do we have the capabilities required to deliver value from IT?

Do we have the human capital to leverage the opportunities that IT provides.

4. Is our IT investment portfolio aligned with opportunities and threats?

Does the portfolio balance short-term as well as long-term needs?

That is, is the IT investment portfolio meeting today's needs, as well as preparing the company for whatever may come in the future.

5. Who is responsible for realizing value from IT?

Who is accountable for IT?

There should be individual or individuals in the IT organization who should make sure that the company is making the right investments, and then deriving the value from those investments.

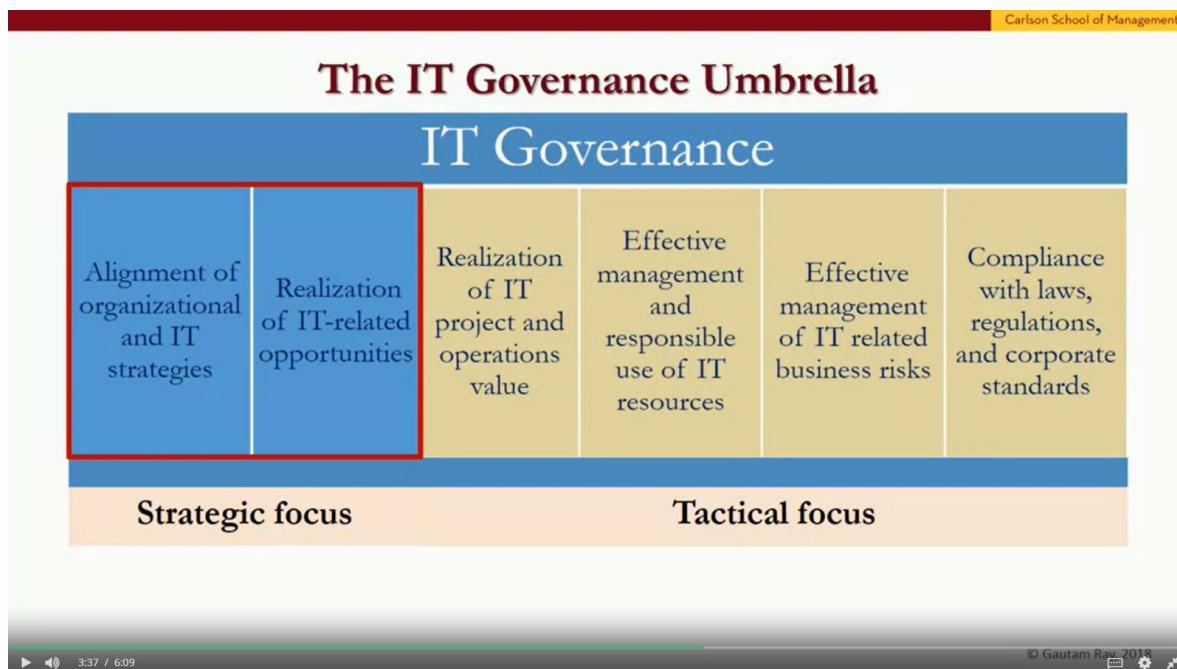
6. Are we comfortable with our level of IT risk?

We know that our IT investment portfolio has many risks associated with it.

So, we should be asking the question, do we understand all the risks?

Have we taken measures to protect ourselves against those risks?

So, we can think of IT governance as an umbrella, and there are two key dimensions or parts to this umbrella. The first is the strategic focus of the umbrella, and then there is the tactical part that I will come to.



The strategic focus of IT governance

1. The strategic focus in IT governance is alignment of organizational and IT strategies.
Does the organization have processes, methods, standards in place to make sure that the business plans align with IT plans and the IT plans align with the business plans?

2. The second element of the strategic focus of IT governance is realization of IT opportunities.
Are we making sure that we are taking advantage of all the opportunities that IT provides?

The tactical focus of IT governance

The tactical element can be divided into four elements.

1. Realization of IT project and operations value.
IT investment is made in terms of individual projects.
Are we making sure that these individual projects are meeting their goals?
Are the projects on time?
Are the projects realizing the value that they were expected to provide?

2. Effective management and responsible use of IT resources.
Are we using our IT resources responsibly?

For example, if we are using different software in the organization, are we making sure that all the software licenses are up to date?

3. Effective management of IT related business risks.

There are significant risks associated with the IT investment portfolio as the Target example illustrated.

Are we making sure that we have evaluated our risks, and have safeguards in place?

4. Are we complying with all the IT related laws and regulations?



Information Systems Management: Key Principles and Practices

Course 1: IS/IT Governance
Module 1.3: IT Alignment

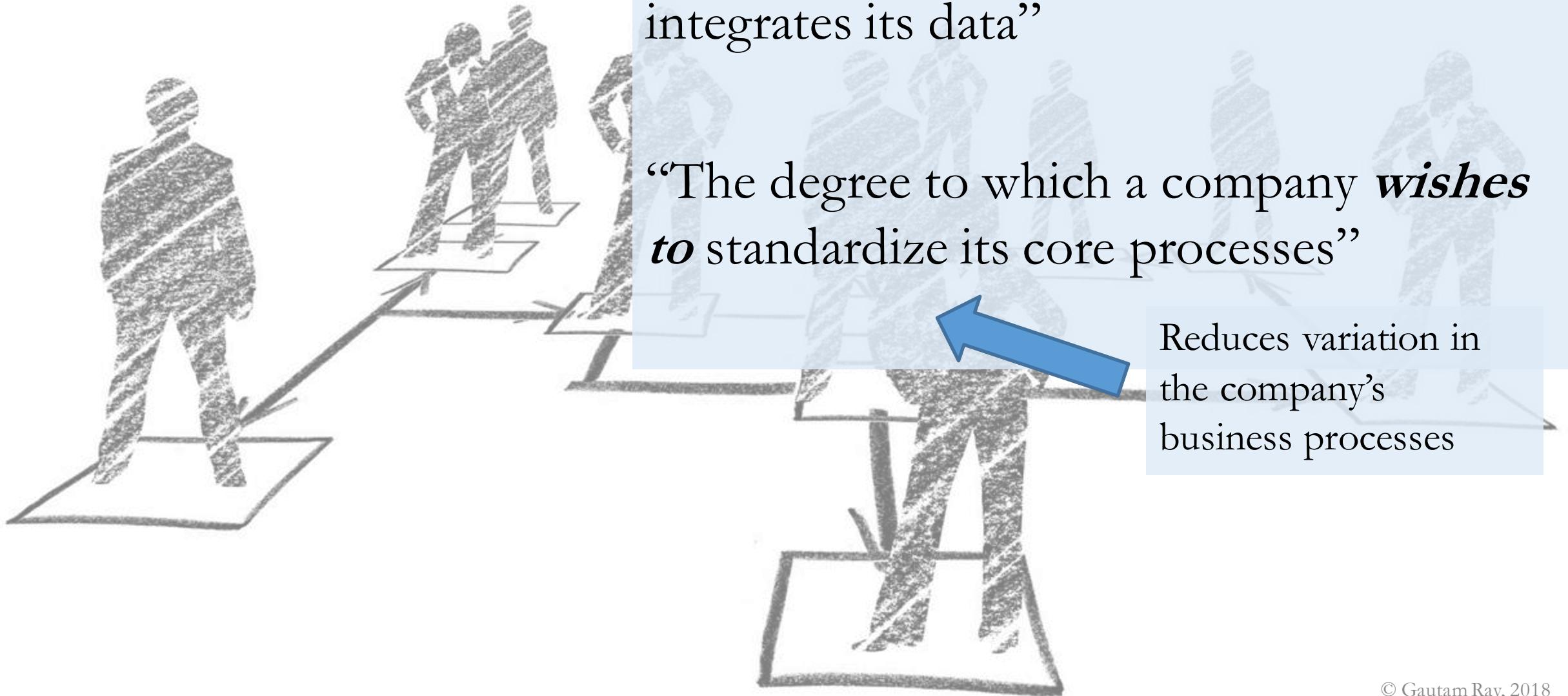
Module Overview and Goals

1. Define IT Governance and Understand its Significance
2. A Framework for IT Governance
3. Operating Model and the Maturity of Enterprise Architecture

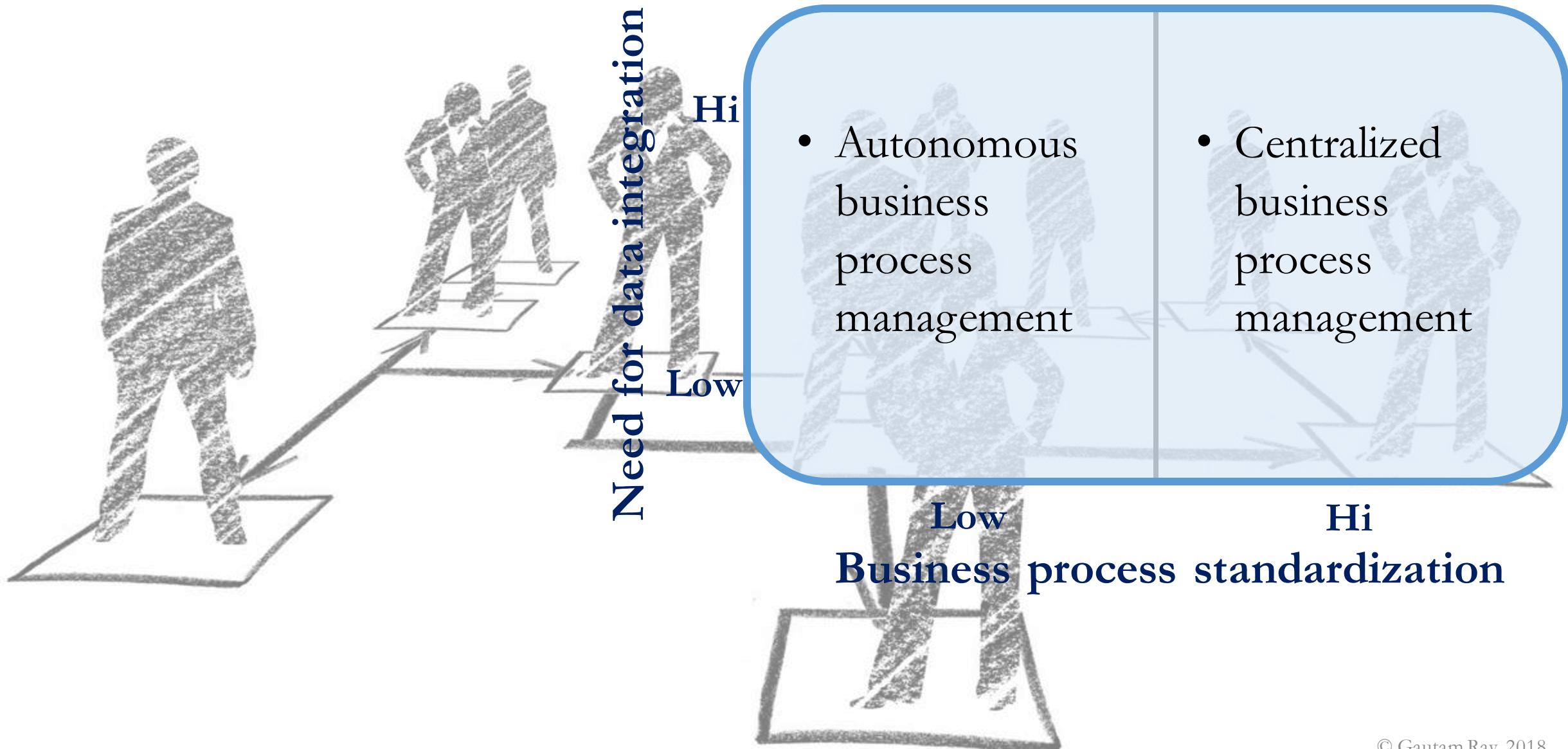
Lesson Overview and Goals

- Operating Model and the Maturity of Enterprise Architecture
 1. Discover the operating model of a company
 2. Explore the relationship between the operating model and enterprise architecture.
 3. Discuss strategic implications of each stage of the enterprise architecture.

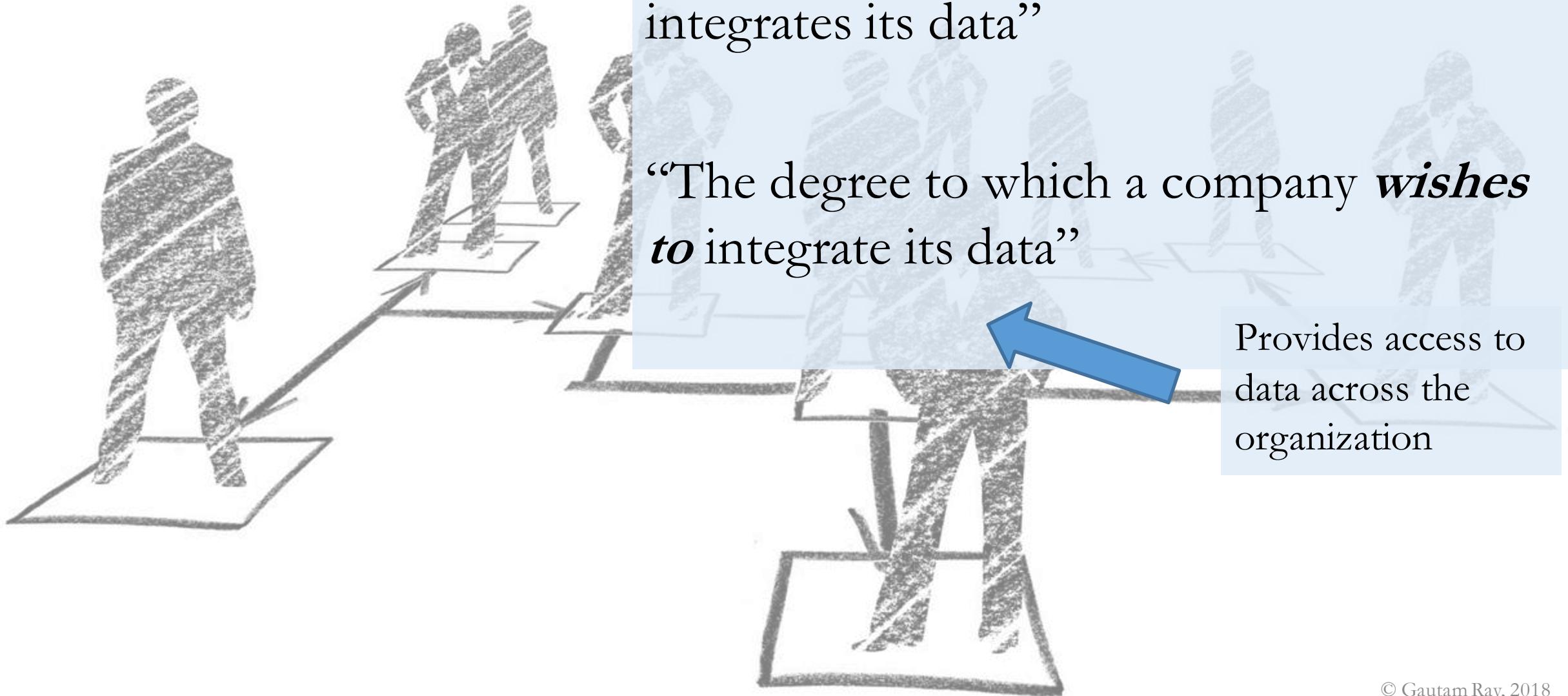
Operating Model



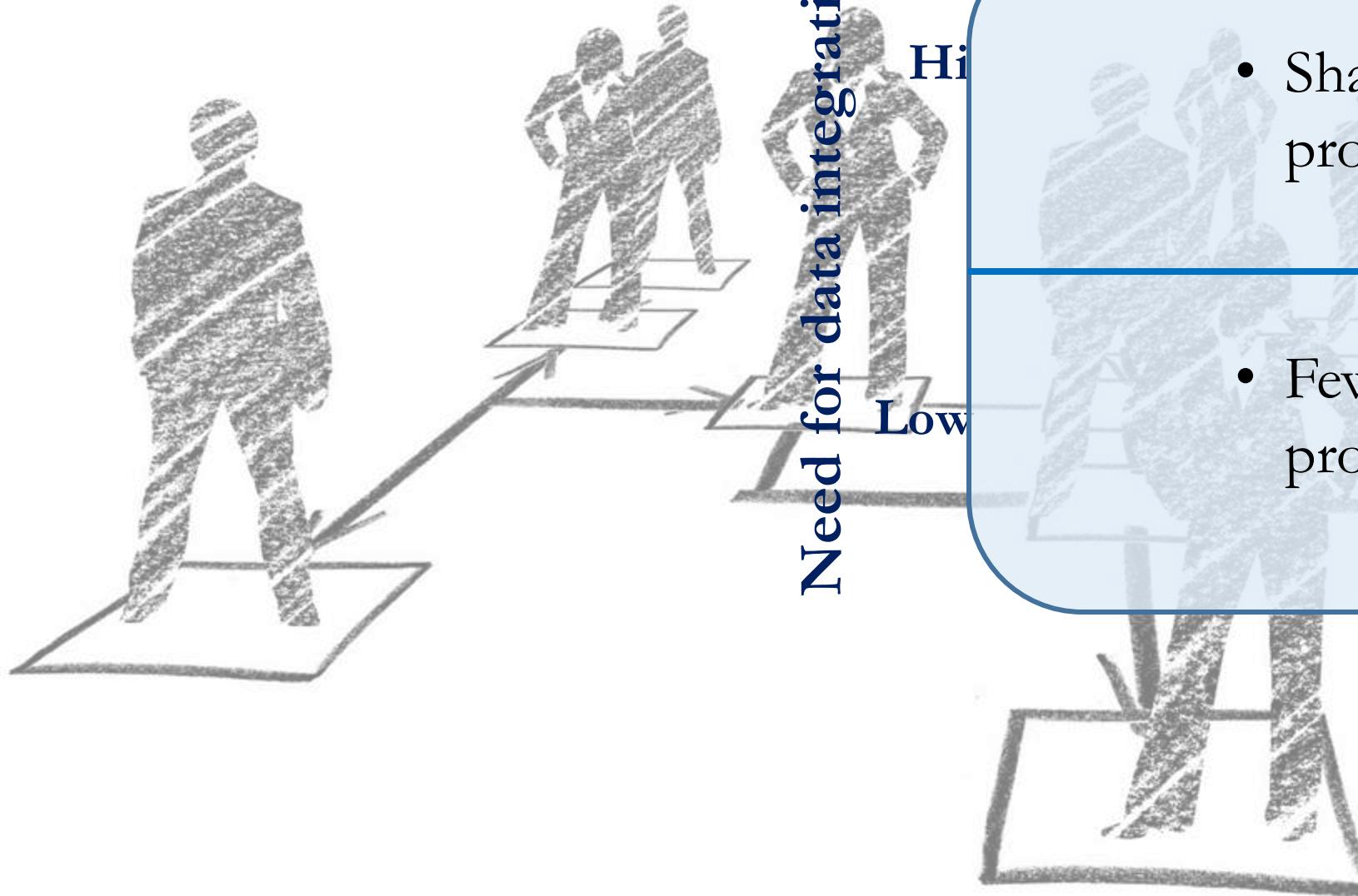
Operating Model



Operating Model



Operating Model



Operating Model



Need for data integration

Hi

Low

Coordination

- Shared customers, products, suppliers
- Business unit control over business process design

Diversification

- Few shared customers, products, suppliers
- Business unit control over business process design

Unification

- Shared customers, products, suppliers
- High level process owners design standard processes

Replication

- Few shared customers, products, suppliers
- Locally owned data
- Shared business processes

Low

Hi

Business process standardization

Operating Model



Need for data integration

Hi

Low

Coordination

- Shared customers, products, suppliers
- Business unit control over business process design

Diversification

- Few shared customers, products, suppliers
- Business unit control over business process design

Unification

- Shared customers, products, suppliers
- High level process owners design standard processes

Replication

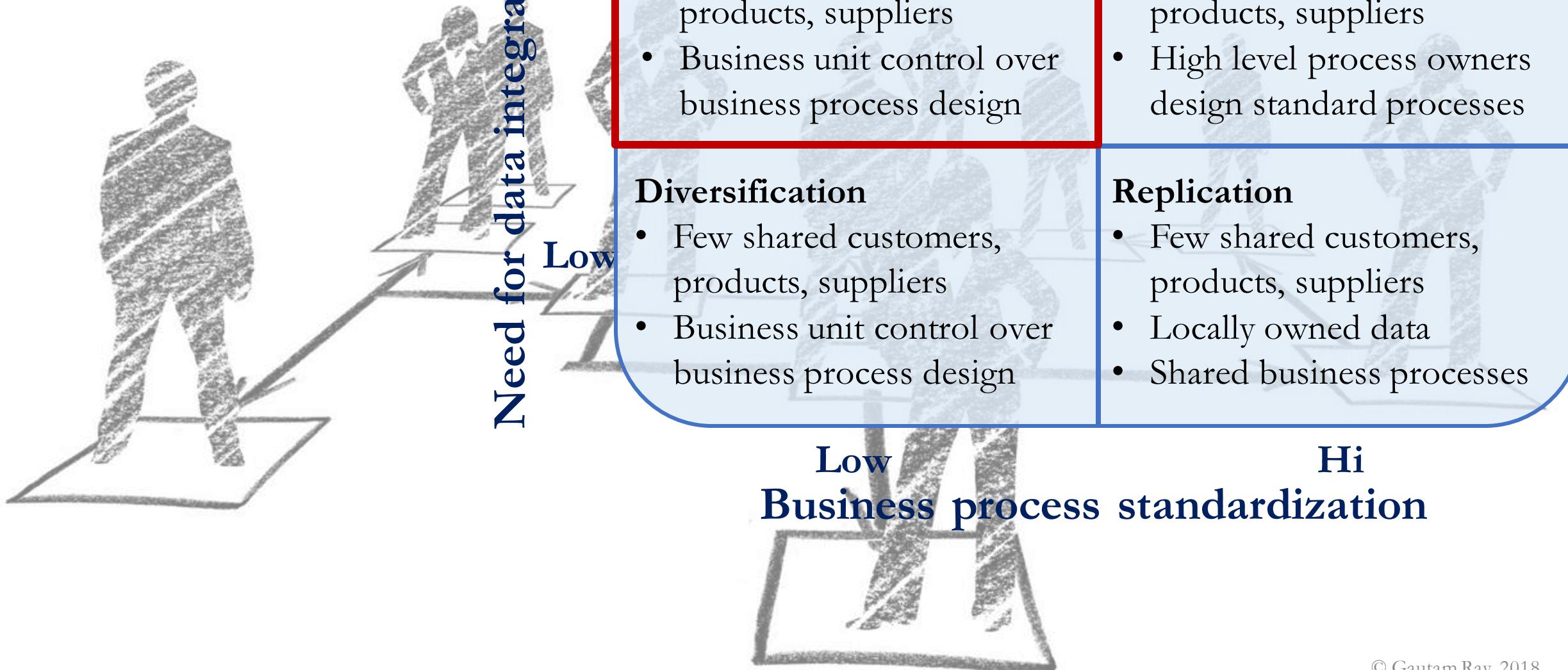
- Few shared customers, products, suppliers
- Locally owned data
- Shared business processes

Low

Hi

Business process standardization

Operating Model



Operating Model



Coordination

- Shared customers, products, suppliers
- Business unit control over business process design

Diversification

- Few shared customers, products, suppliers
- Business unit control over business process design

Unification

- Shared customers, products, suppliers
- High level process owners design standard processes

Replication

- Few shared customers, products, suppliers
- Locally owned data
- Shared business processes

Low Hi
Business process standardization



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Image Credits

<u>Image/Source</u>	<u>Copyright</u>	<u>Locati on</u>	<u>Notes</u>
Operating Model and Enterprise Architecture	Ross, J. W., Weill, P., & Robertson, D. (2006). <i>Enterprise architecture as strategy: Creating a foundation for business execution.</i> Harvard Business Press.	Slide 4-11	
Operating Model and Enterprise Architecture	Ross, J. W. Creating a Strategic IT Architecture Competency: Learning in stages, <i>MIS Quarterly Executive</i> , Volume 2, Number 1, March 2003, pp. 31-43.	Slide 4-11	



Information Systems Management: Key Principles and Practices

Course 1: IS/IT Governance

Module 2.1: Evaluating IT Investments

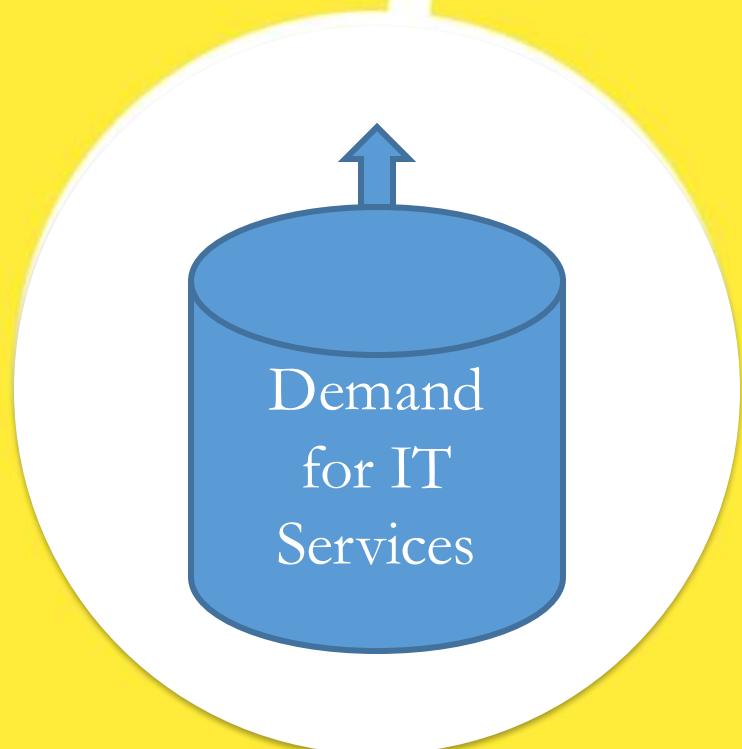
Module Overview and Goals

1. Evaluate Individual IT Investments
2. Incorporating Risk in IT Investment Evaluation

Lesson Overview and Goals

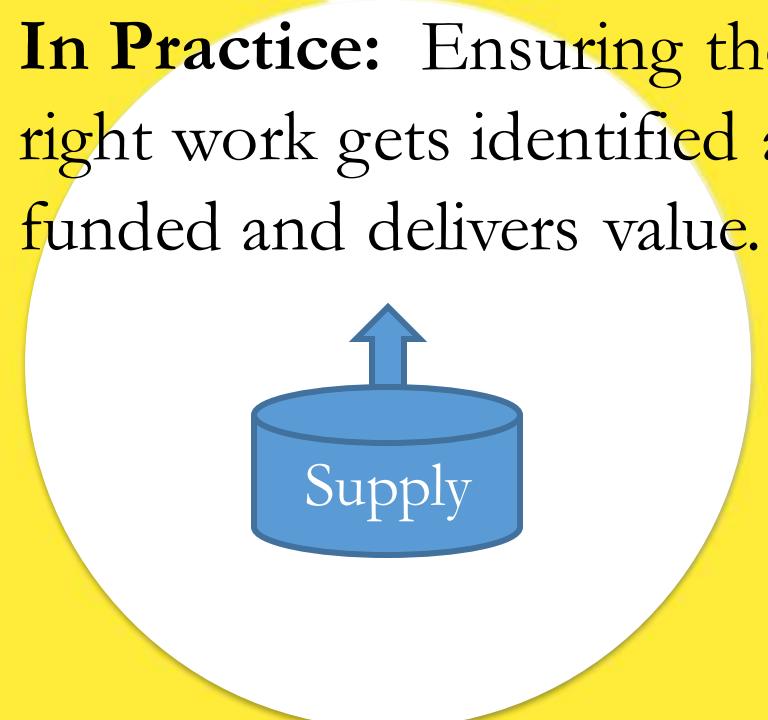
- Evaluate Individual IT Investments
 1. Explore the different kinds of estimating

IT Demand Management



In Theory: Allocating limited resources to the overall benefit of the enterprise.

In Practice: Ensuring the right work gets identified and funded and delivers value.



Three types of Estimating

	Order of Magnitude	Budget	Definitive
SDLC phase	Proposal	Proposal	Plan
What do you know?	Solution strategy	Deliverables (or general use cases)	Work packages (or definitive use cases)
Estimating method	Top down	Top down	Bottom up
Range of project estimate	-30% to +60%	-15% to +30%	-5% to + 5%

Three types of Estimating

	Order of Magnitude	Budget	Definitive
SDLC phase	Proposal	Proposal	Plan
What do you know?	Solution strategy	Deliverables (or general use cases)	Work packages (or definitive use cases)
Estimating method	Top down	Top down	Bottom up
Range of project estimate	-30% to +60%	-15% to +30%	-5% to + 5%

Three types of Estimating

	Order of Magnitude	Budget	Definitive
SDLC phase	Proposal	Proposal	Plan
What do you know?	Solution strategy	Deliverables (or general use cases)	Work packages (or definitive use cases)
Estimating method	Top down	Top down	Bottom up
Range of project estimate	-30% to +60%	-15% to +30%	-5% to + 5%

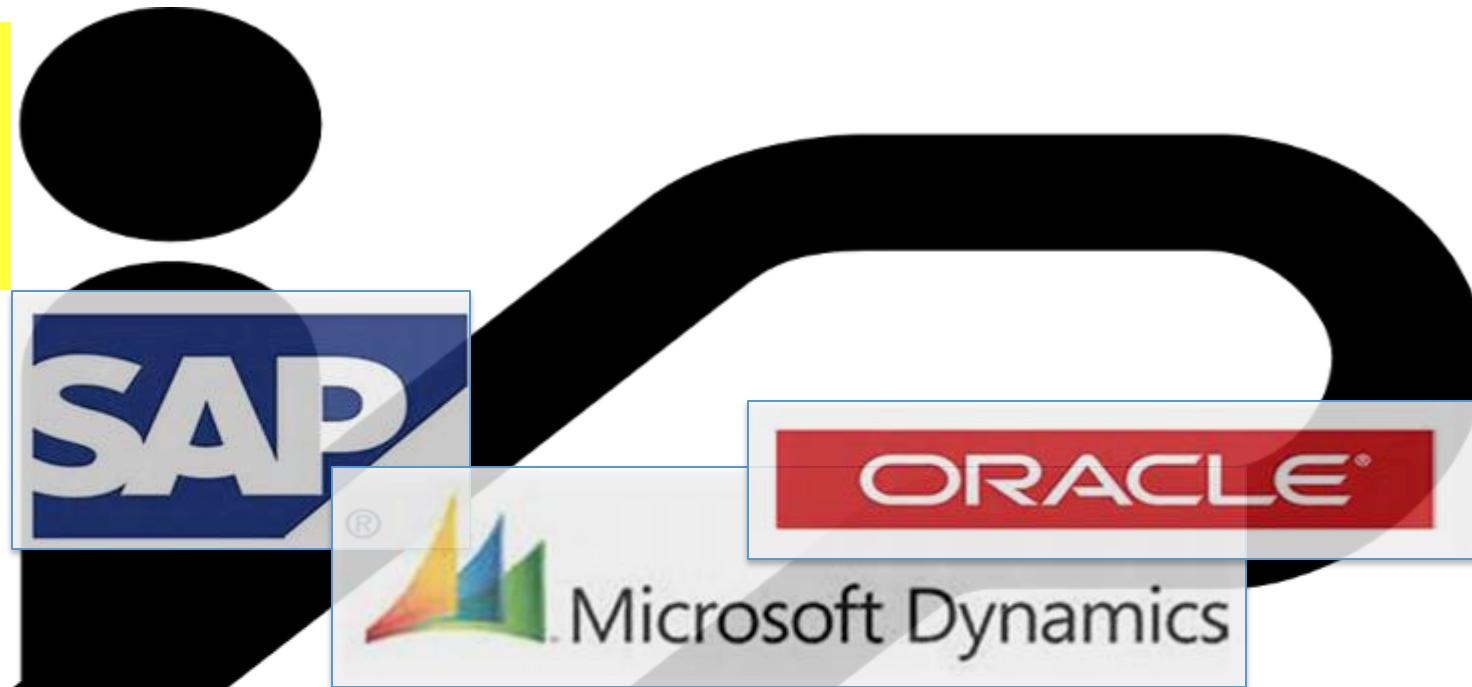
Top Down Approach: Parametric



Top Down Approach: Parametric



What is the cost of implementing an ERP system?



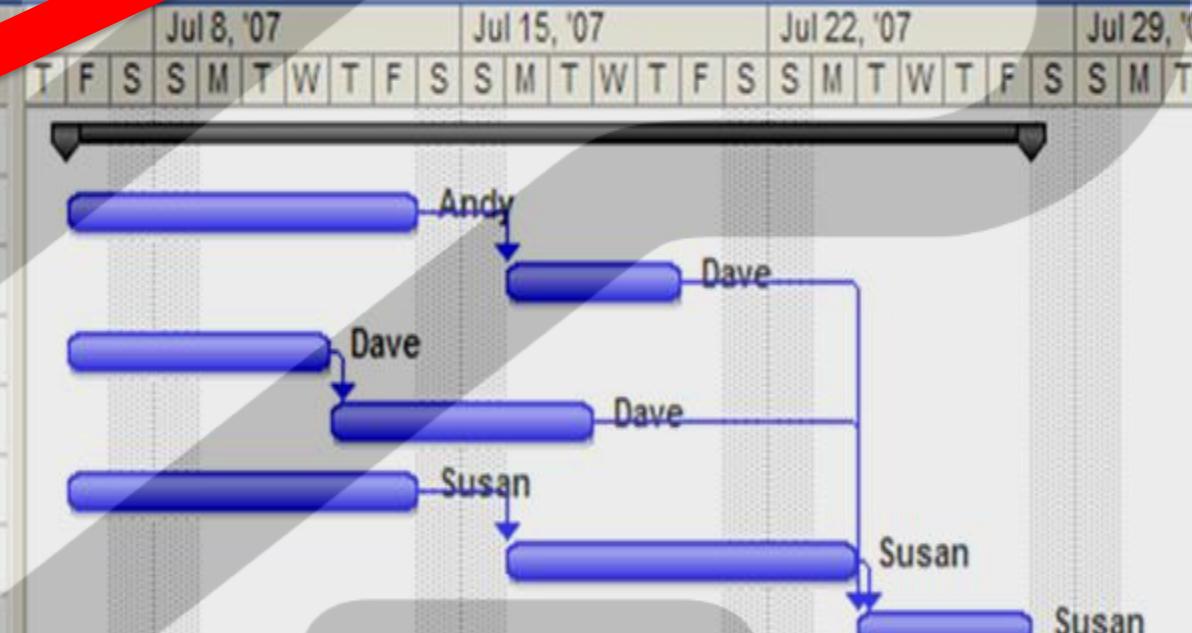
“Expect a project cost of 1% of revenue”

The only function of a ballpark estimate should be to evaluate whether it would be useful to get a more accurate estimate.

Bottom up Estimating: We know the work

Work package level estimates

	Task Name	Duration	Start	Finish	Pred	Resources
1	Employee Survey Proj	16 days	7/6/07	7/27/07		
2	Develop Data Analysis	6 days	7/6/07	7/13/07		Andy
3	Test Data Analysis Met	4 days	7/16/07	7/19/07	2	Dave
4	Develop Web Site	4 days	7/6/07	7/11/07		Dave
5	Test Web Site	4 days	7/12/07	7/17/07	4	Dave
6	Review Survey with M	6 days	7/6/07	7/13/07		Susan
7	Publicize Survey	6 days	7/16/07	7/23/07	6	Susan
8	Oversee Survey	4 days	7/24/07	7/27/07	3,5,7	Susan



Budget Estimate: Total Cost of Ownership



- Acquisition and Procurement
 - Developing bid specifications
 - Hardware and Software cost
- Operations and Maintenance
 - IT Staff
 - Space, furniture, energy.
- End-of-life Management
 - Sanitizing storage media
 - Physical shipping and delivery.

Budget Estimate: Total Cost of Ownership



- Acquisition and Procurement
 - Developing bid specifications
 - Hardware and Software cost
- Operations and Maintenance
 - IT Staff
 - Space, furniture, energy.
- End-of-life Management
 - Sanitizing storage media
 - Physical shipping and delivery.

Budget Estimate: Total Cost of Ownership



- Acquisition and Procurement
 - Developing bid specifications
 - Hardware and Software cost
- Operations and Maintenance
 - IT Staff
 - Space, furniture, energy.
- End-of-life Management
 - Sanitizing storage media
 - Physical shipping and delivery.

An example of a Top Down Budget Estimate: Anesthesia Information Management System



Figure 3. AIMS workstation mounted on an anesthesia machine.

Table 2 Operating cash flow and capital investment

3. Operating cash flow
 - i. anesthetic-related drug costs:
 - reduced anesthetic-related drug costs = 9,200 cases per year $\times \$32$ per case = \$294,400
 - ii. money per case:
 - financial saving per case = \$100 per case \times 9,200 cases per year = \$920,000 per year
 - iii. hospital reimbursement:
 - annual increase: $\$71,910,048 \times 0.4 \times 0.015 = \$431,460$ per year
- Total cost savings per year = \$294,400 + \$920,000 + \$431,460 = \$1,643,860 per year**
4. Capital spending
 - i. number of operating room work stations
 - \$8,000 per operating room work stations \times 14 operating rooms = \$112,000
 - \$10,000 per AIMS and server software \times 14 workstations = \$140,000
 - ii. number of office work stations
 - \$5,000 per recovery room and office workstation \times 3 workstations = \$15,000
 - \$10,000 per AIMS and server software \times 3 workstations = \$30,000
 - iii. estimate of routers/switches/cables
 - routers, switches, and cables = \$15,000
- Capital Investment: \$112,000 + \$140,000 + \$15,000 + \$30,000 + \$15,000 = \$312,000**

AIMS = anesthesia information management systems.



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Image Credits

Image/Source	Copyright	Location	Notes
Top Down Approach: Parametric	Verzuh, E. (2015). <i>The fast forward MBA in project management</i> . John Wiley & Sons.	Slide 8-9	
Budget Estimate: Total Cost of Ownership	McKeen, J. D., & Smith, H. A. (2010). Developments in Practice XXXVII: Total Cost of Ownership. <i>CAIS</i> , 27, 32.	Slides 11-13	
Anesthesia Information Management System	Sinclair, D. R. (2012). Discounted cash flow of anesthesia information management systems. <i>Journal of clinical anesthesia</i> , 24(7), 603-604.	Slide 14	



Information Systems Management: Key Principles and Practices

Course 1: IS/IT Governance

Module 2.1: Evaluating IT Investments

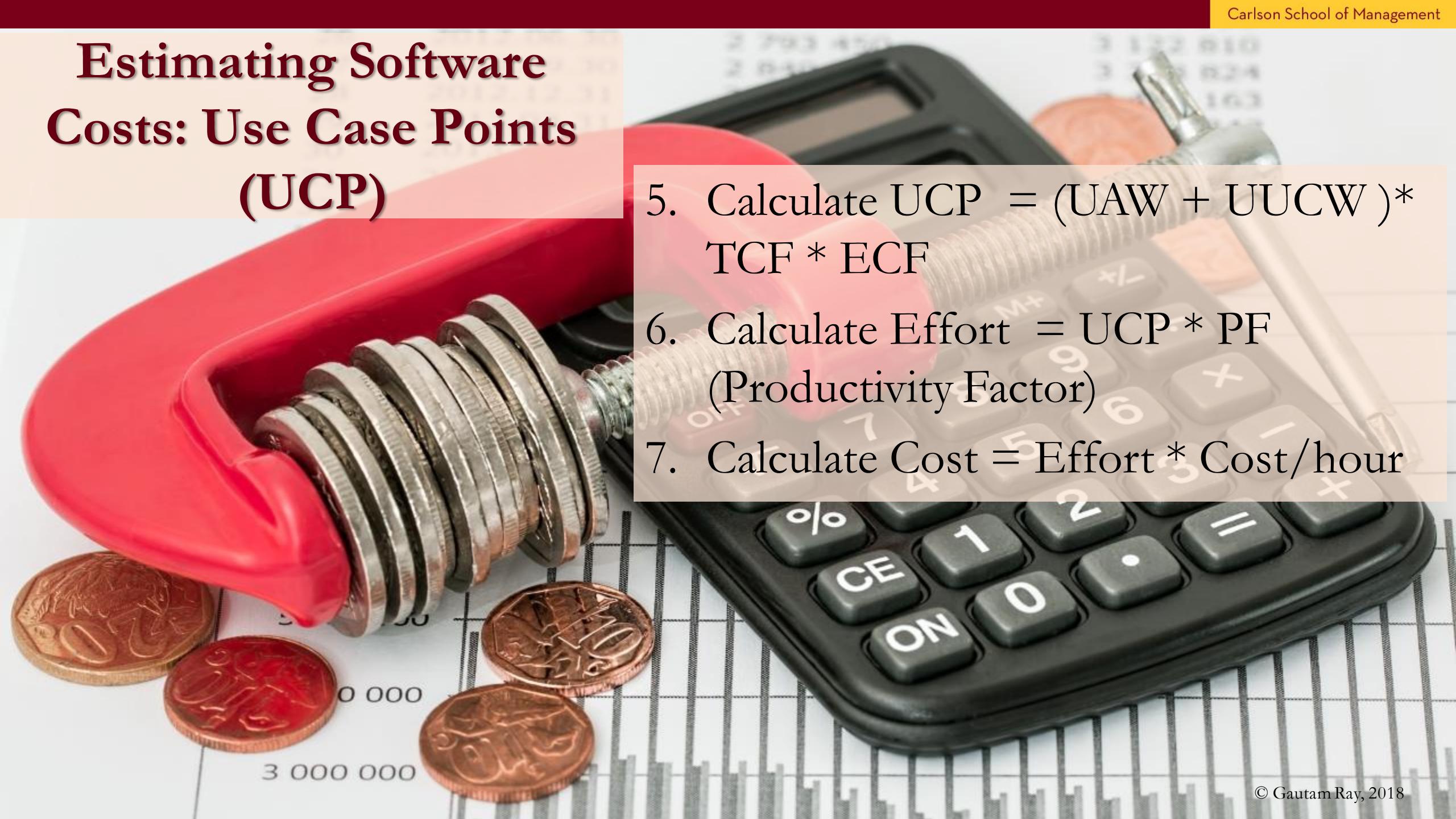
Lesson Overview and Goals

- Evaluate Individual IT Investments
 1. Explore the different kinds of estimating
 2. Discover a formal method to estimate software costs

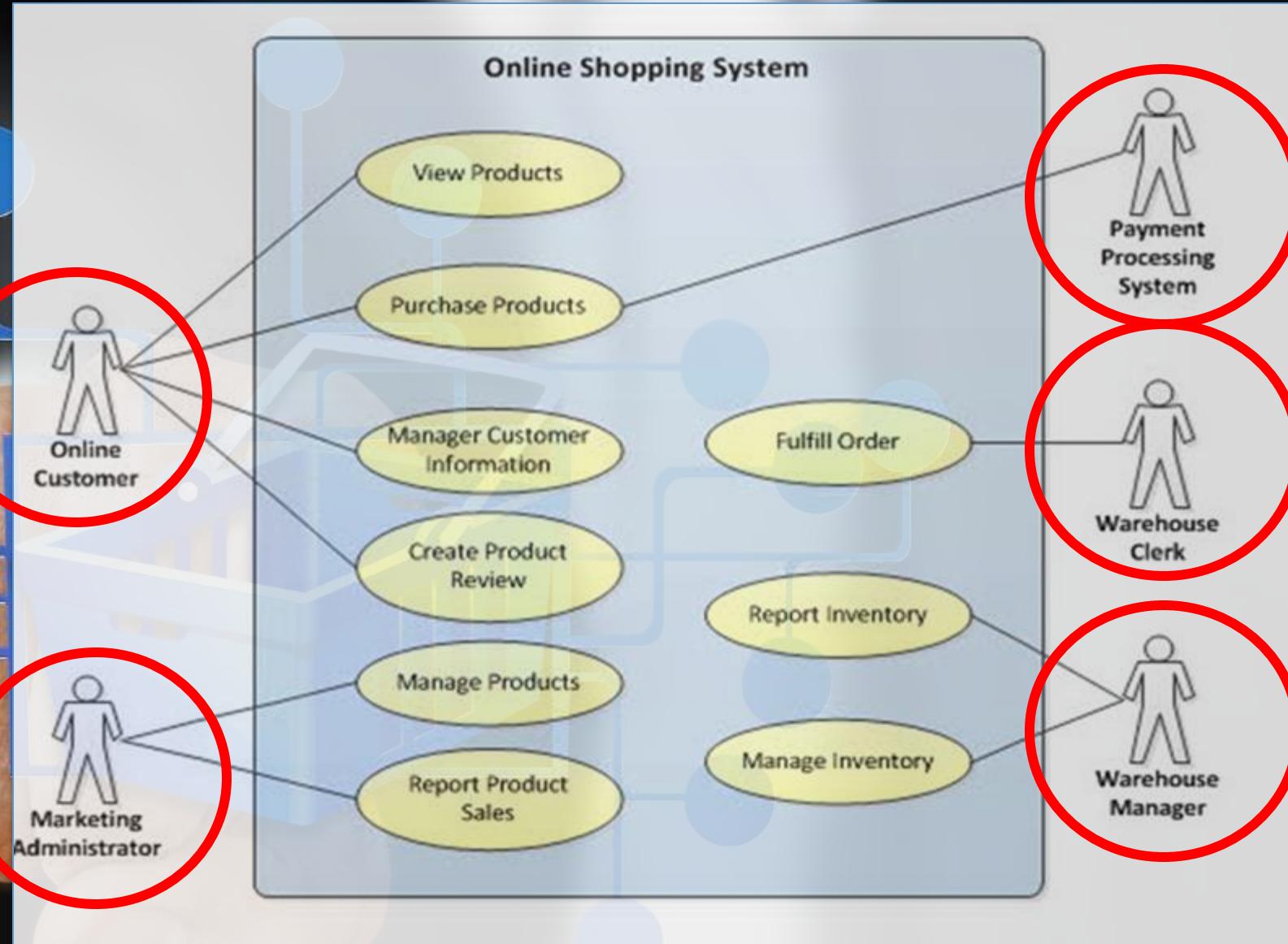
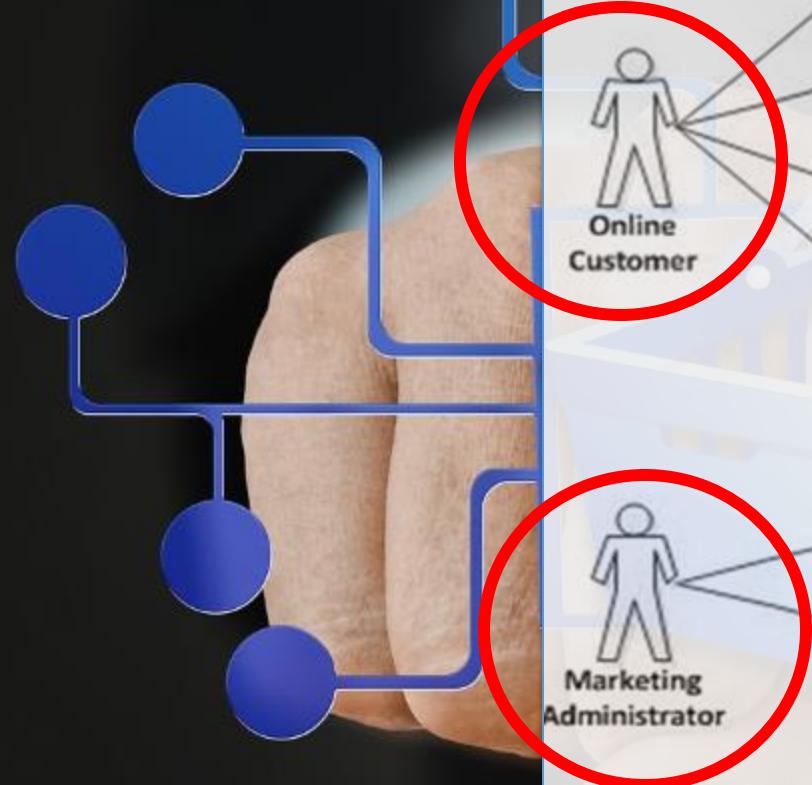
Estimating Software Costs: Use Case Points (UCP)

1. Develop use case diagram and use cases
2. Calculate UAW (unadjusted actor weight)
3. Calculate UUCW (unadjusted use-case weight)
4. Estimate TCF and ECF (technical and environmental complexity factors)

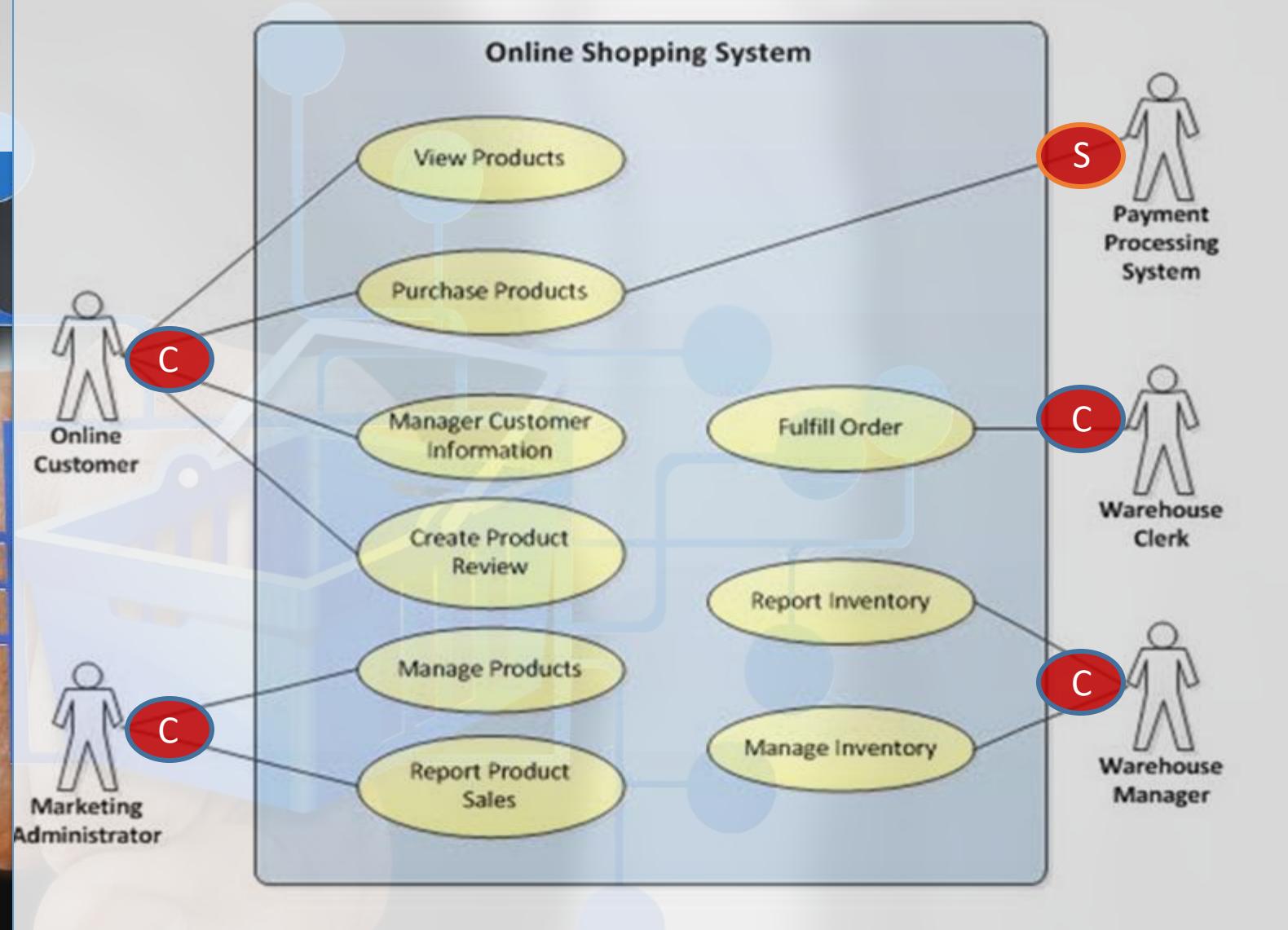
Estimating Software Costs: Use Case Points (UCP)

- 
- A red car key fob with several silver coins attached to its chain, resting on a calculator and some coins.
5. Calculate UCP = $(UAW + UUCW) * TCF * ECF$
 6. Calculate Effort = UCP * PF
(Productivity Factor)
 7. Calculate Cost = Effort * Cost/hour

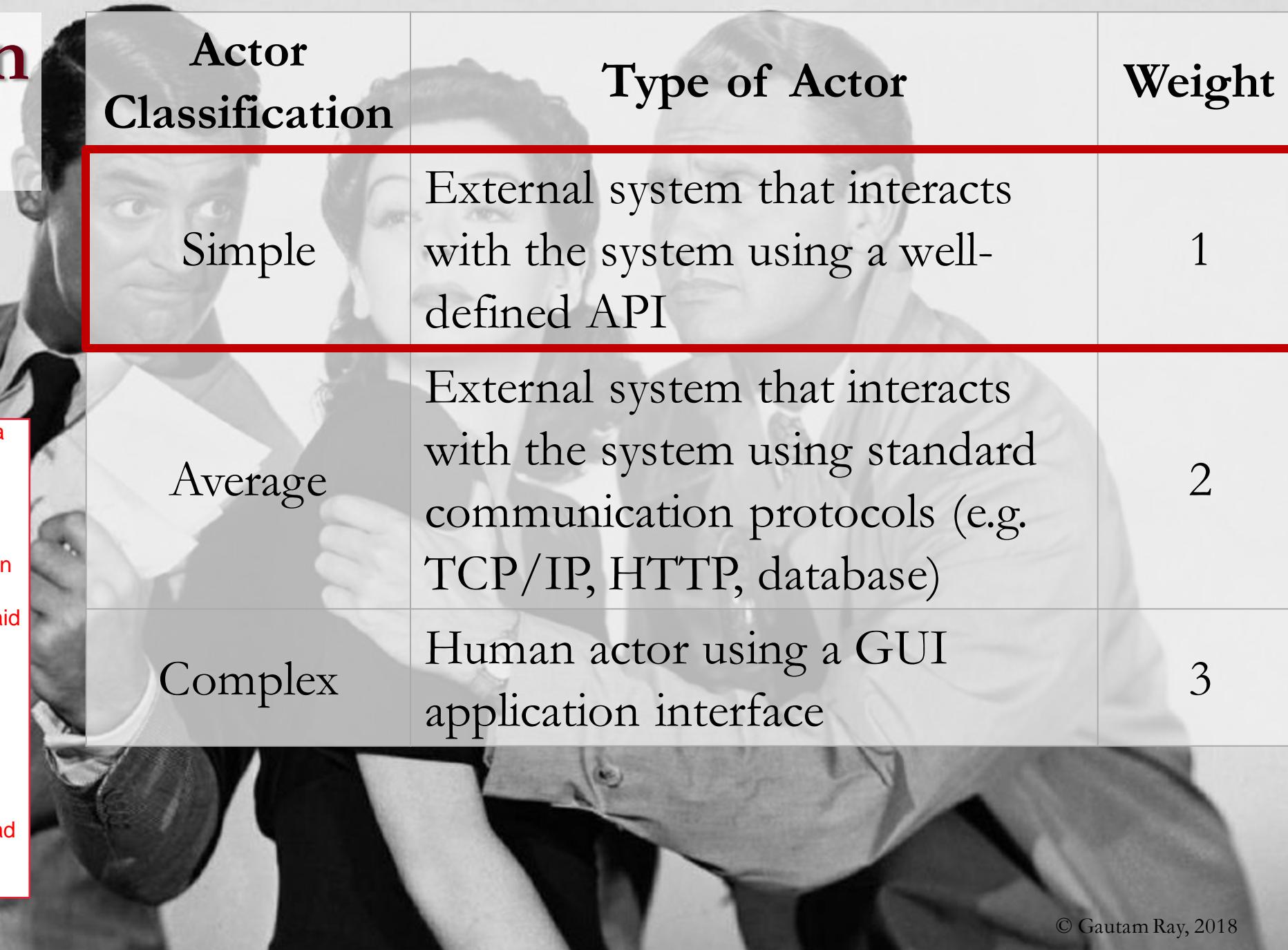
1. Develop UCD and Use Cases



2. Calculate UAW



Classification of Actors



Actor Classification	Type of Actor	Weight
Simple	External system that interacts with the system using a well-defined API	1
Average	External system that interacts with the system using standard communication protocols (e.g. TCP/IP, HTTP, database)	2
Complex	Human actor using a GUI application interface	3

An application programming interface (API) is a connection between computers or between computer programs. It is a type of software interface, offering a service to other pieces of software.^[1] A document or standard that describes how to build or use such a connection or interface is called an API specification. A computer system that meets this standard is said to implement or expose an API. The term API may refer either to the specification or to the implementation.

The graphical user interface (GUI) is a form of user interface that allows users to interact with electronic devices through graphical icons and audio indicator such as primary notation, instead of text-based user interfaces, typed command labels or text navigation.

Classification of Actors

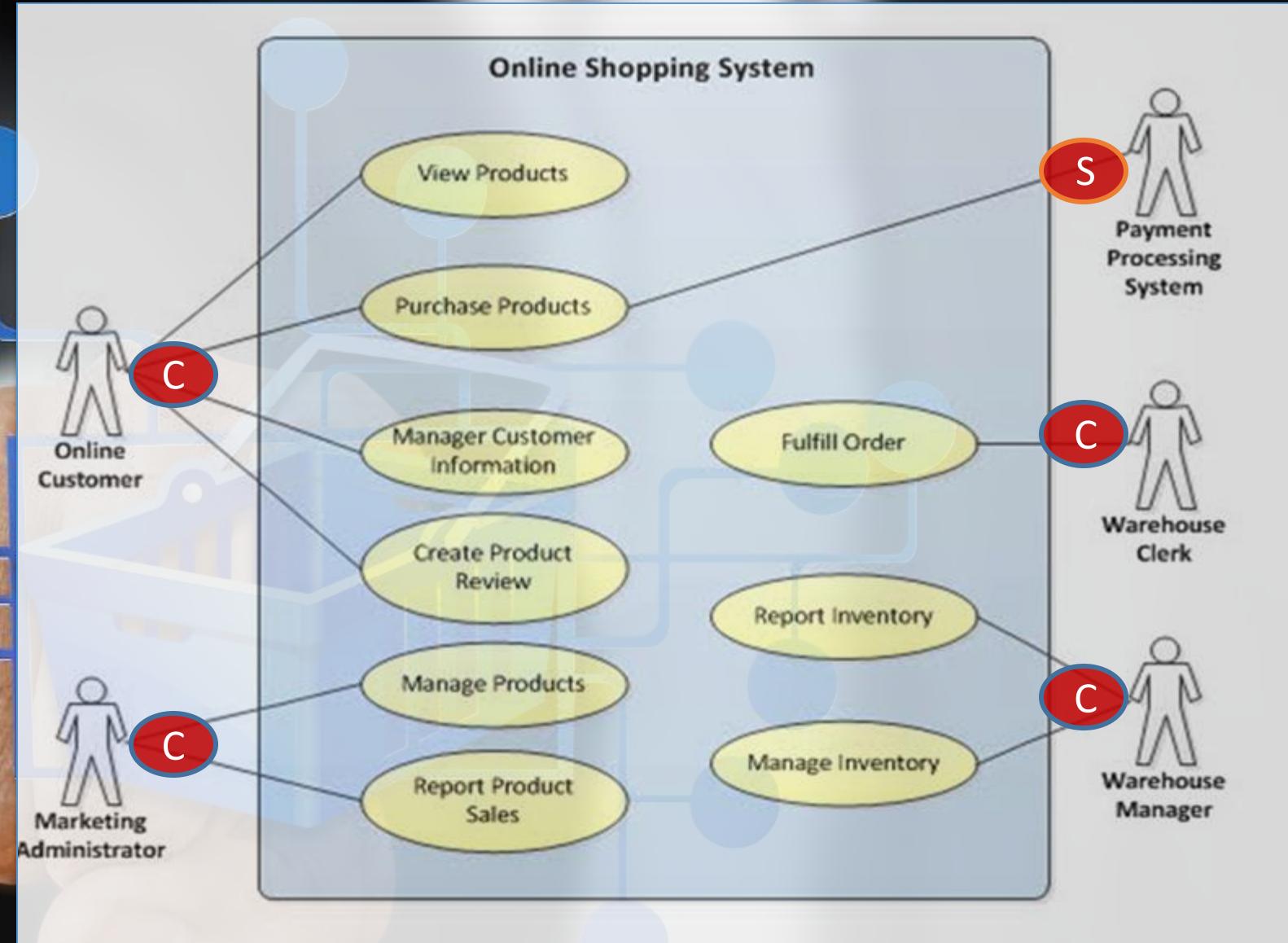
Actor Classification	Type of Actor	Weight
Simple	External system that interacts with the system using a well-defined API	1
Average	External system that interacts with the system using standard communication protocols (e.g. TCP/IP, HTTP, database)	2
Complex	Human actor using a GUI application interface	3

Classification of Actors

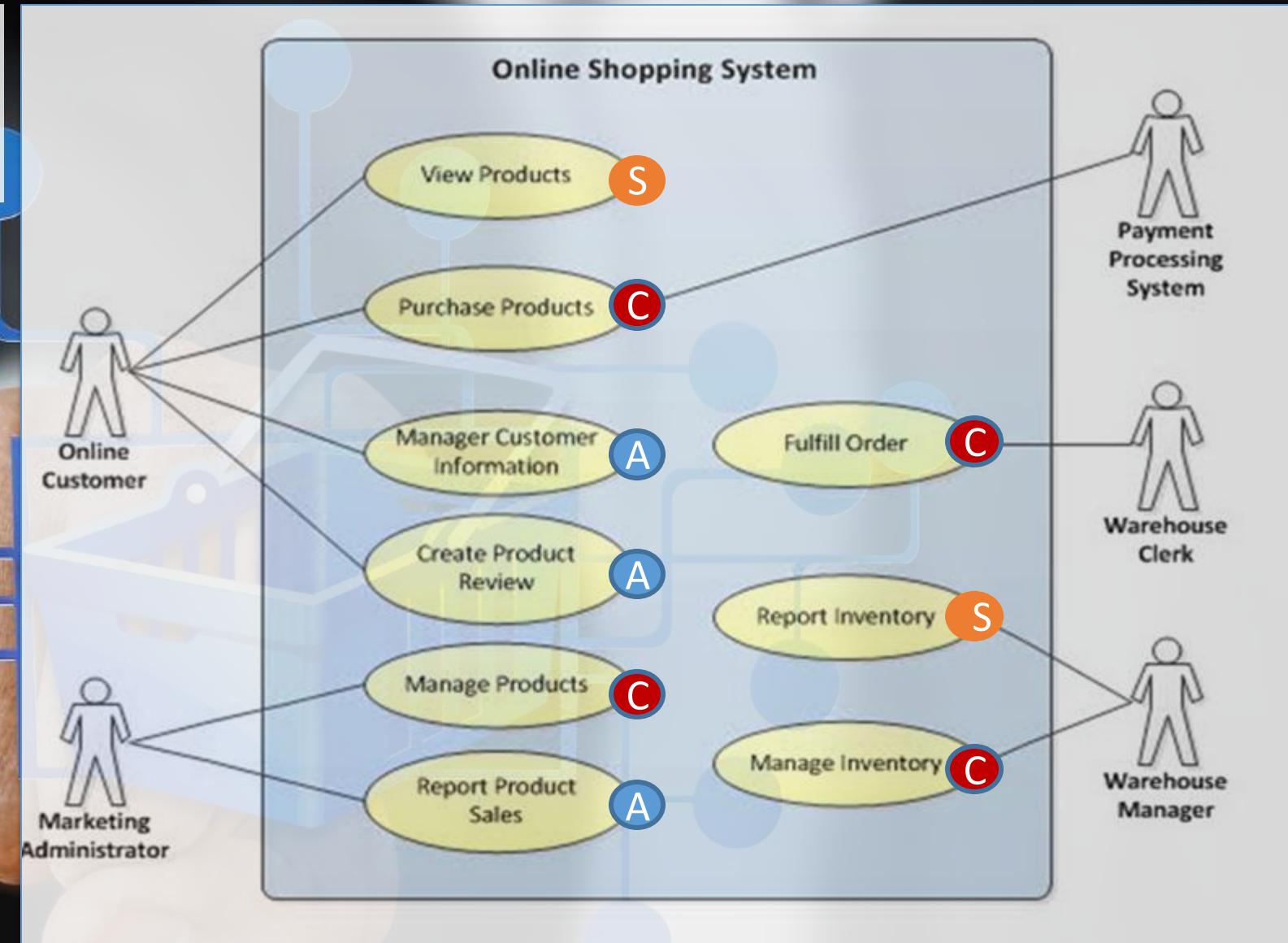
Actor Classification	Type of Actor	Weight
Simple	External system that interacts with the system using a well-defined API	1
Average	External system that interacts with the system using standard communication protocols (e.g. TCP/IP, HTTP, database)	2
Complex	Human actor using a GUI application interface	3

2. Calculate UAW

- Simple = $1*1$
- Average = $0*2$
- Complex = $4*3$
- UAW = $1*1 + 4*3 = 13$



3. Calculate UUCW



Classification of Use Cases

Use Case Classification

Simple

No. of Transactions

1 to 3 transactions

Weight

5

Average

4 to 7 transactions

10

Complex

8 or more transactions

15



Classification of Use Cases

Use Case Classification

Simple

No. of Transactions

1 to 3 transactions

Weight

5

Average

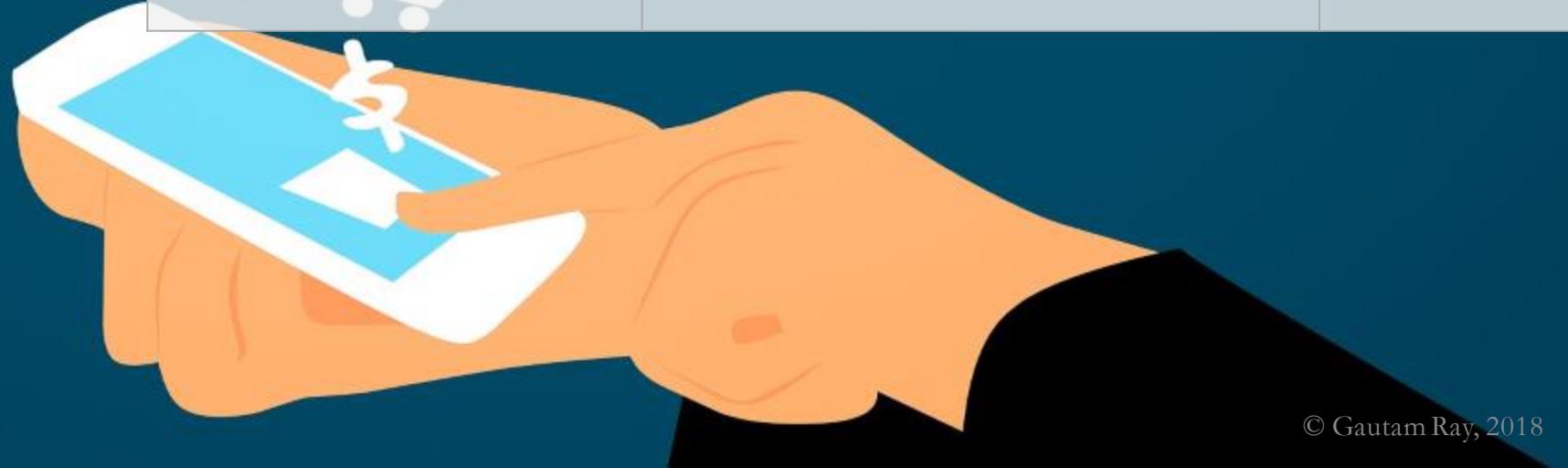
4 to 7 transactions

10

Complex

8 or more transactions

15



Classification of Use Cases

Use Case Classification

Simple

No. of Transactions

Weight

1 to 3 transactions

5

Average

4 to 7 transactions

10

Complex

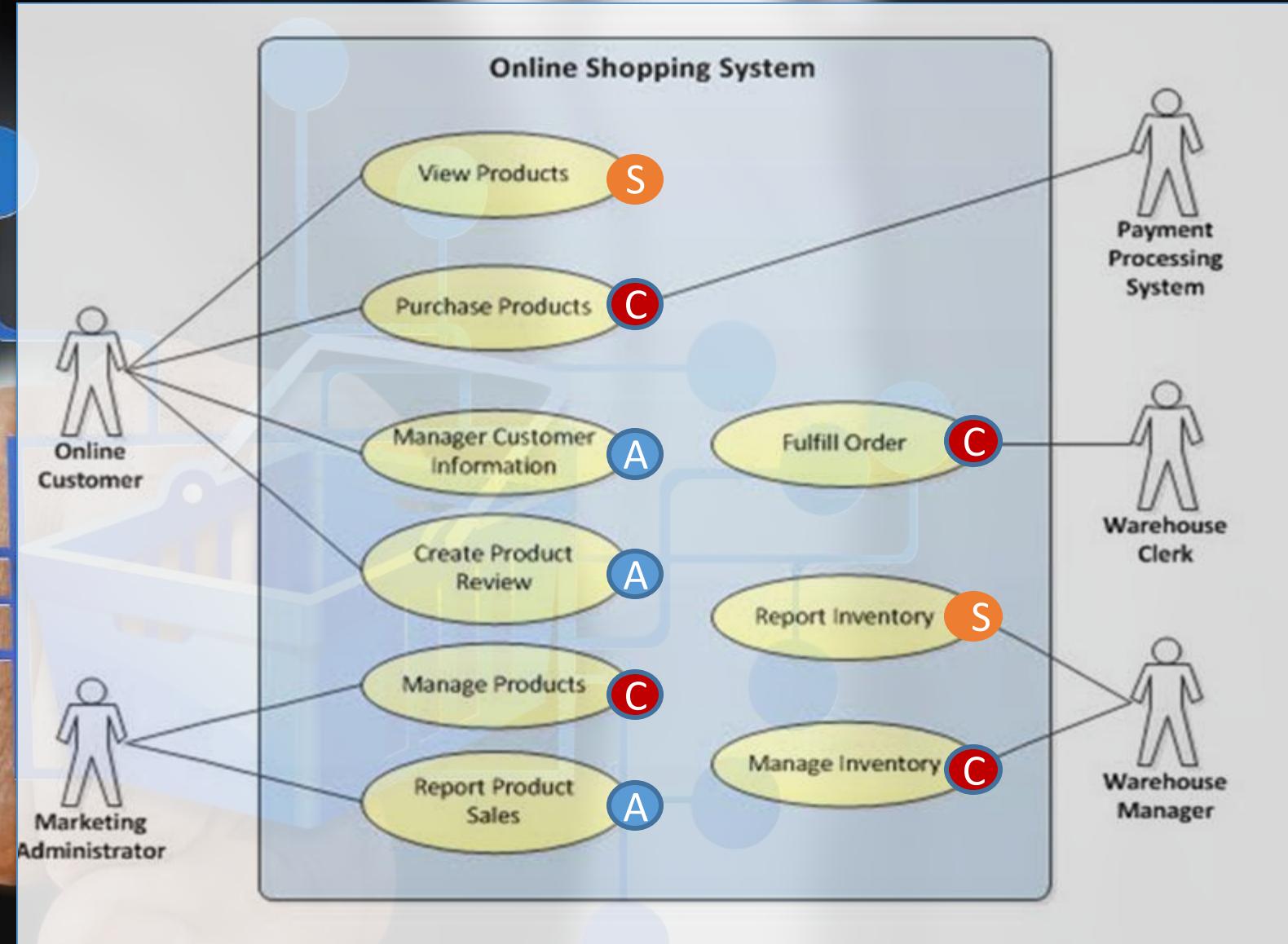
8 or more transactions

15



3. Calculate UUCW

- Simple = $2*5$
- Average = $3*10$
- Complex = $4*15$
- UUCW = $2*5 + 3*10 + 4*15 = 100$





CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Image Credits

Image/Source	Copyright	Location	Notes
Use Case Point Estimation	https://en.wikipedia.org/wiki/Use_Case_Points	Slides 5-15	



Information Systems Management: Key Principles and Practices

Course 1: IS/IT Governance

Module 2.1: Evaluating IT Investments

Lesson Overview and Goals

- Evaluate Individual IT Investments
 1. Explore the different kinds of estimating
 2. Discover a formal method to estimate software costs

Calculating Technical and Environmental Complexity Factor

- Relative Importance of Each Sub-factor

- Assigned-value as specified by the development team
 - Irrelevant: 0
 - Average: 3
 - Strong: 5

4. Estimate Technical Complexity Factor (TCF)

Factor	Description	Weight	Assigned Value	Weight x Assigned Value
T1	Distributed system	2.0	5	10
T2	Response time/performance objectives	1.0	5	5
T3	End-user efficiency	1.0	3	3
T4	Internal processing complexity	1.0	2	2
T5	Code reusability	1.0	3	3
T6	Easy to install	0.5	1	0.5
T7	Easy to use	0.5	5	2.5
T8	Portability to other platforms	2.0	2	4
T9	System maintenance	1.0	2	2
T10	Concurrent/parallel processing	1.0	3	3
T11	Security features	1.0	5	5
T12	Access for third parties	1.0	1	1
T13	End user training	1.0	1	1
Total (TF):				42

4. Estimate Technical Complexity Factor (TCF)

Factor	Description	Weight	Assigned Value	Weight x Assigned Value
T1	Distributed system	2.0	5	10
T2	Response time/performance objectives	1.0	5	5
T3	End-user efficiency	1.0	3	3
T4	Internal processing complexity	1.0	2	2
T5	Code reusability	1.0	3	3
T6	Easy to install	0.5	1	0.5
T7	Easy to use	0.5	5	2.5
T8	Portability to other platforms	2.0	2	4
T9	System maintenance	1.0	2	2
T10	Concurrent/parallel processing	1.0	3	3
T11	Security features	1.0	5	5
T12	Access for third parties	1.0	1	1
T13	End user training	1.0	1	1
Total (TF):				42

4. Estimate Technical Complexity Factor (TCF)

Factor	Description	Weight	Assigned Value	Weight x Assigned Value
T1	Distributed system	2.0	5	10
T2	Response time/performance objectives	1.0	5	5
T3	End-user efficiency	1.0	3	3
T4	Internal processing complexity	1.0	2	2
T5	Code reusability	1.0	3	3
T6	Easy to install	0.5	1	0.5
T7	Easy to use	0.5	5	2.5
T8	Portability to other platforms	2.0	2	4
T9	System maintenance	1.0	2	2
T10	Concurrent/parallel processing	1.0	3	3
T11	Security features	1.0	5	5
T12	Access for third parties	1.0	1	1
T13	End user training	1.0	1	1
Total (TF):				42

4. Estimate Technical Complexity Factor (TCF)

Factor	Description	Weight	Assigned Value	Weight x Assigned Value
T1	Distributed system	2.0	5	10
T2	Response time/performance objectives	1.0	5	5
T3	End-user efficiency	1.0	3	3
T4	Internal processing complexity	1.0	2	2
T5	Code reusability	1.0	3	3
T6	Easy to install	0.5	1	0.5
T7	Easy to use	0.5	5	2.5
T8	Portability to other platforms	2.0	2	4
T9	System maintenance	1.0	2	2
T10	Concurrent/parallel processing	1.0	3	3
T11	Security features	1.0	5	5
T12	Access for third parties	1.0	1	1
T13	End user training	1.0	1	1
Total (TF):				42

4. Estimate Technical Complexity Factor (TCF)

$$\text{TCF} = 0.6 + (\text{TF}/100)$$

$$\text{TCF} = 0.6 + 42/100 = 1.02$$

5. Estimate Environmental Complexity Factor (ECF)

Factor	Description	Weight	Assigned Value	Weightx Assigned Value
E1	Familiarity with development process used	1.5	3	4.5
E2	Application experience	0.5	3	1.5
E3	Object-oriented experience of team	1.0	2	2
E4	Lead analyst capability	0.5	5	2.5
E5	Motivation of the team	1.0	2	2
E6	Stability of requirements	2.0	1	2
E7	Part-time staff	-1.0	0	0
E8	Difficult programming language	-1.0	4	-4
Total (EF):				10.5

5. Estimate Environmental Complexity Factor (ECF)

Factor	Description	Weight	Assigned Value	Weightx Assigned Value
E1	Familiarity with development process used	1.5	3	4.5
E2	Application experience	0.5	3	1.5
E3	Object-oriented experience of team	1.0	2	2
E4	Lead analyst capability	0.5	5	2.5
E5	Motivation of the team	1.0	2	2
E6	Stability of requirements	2.0	1	2
E7	Part-time staff	-1.0	0	0
E8	Difficult programming language	-1.0	4	-4
Total (EF):				10.5

5. Estimate Environmental Complexity Factor (ECF)

Factor	Description	Weight	Assigned Value	Weightx Assigned Value
E1	Familiarity with development process used	1.5	3	4.5
E2	Application experience	0.5	3	1.5
E3	Object-oriented experience of team	1.0	2	2
E4	Lead analyst capability	0.5	5	2.5
E5	Motivation of the team	1.0	2	2
E6	Stability of requirements	2.0	1	2
E7	Part-time staff	-1.0	0	0
E8	Difficult programming language	-1.0	4	-4
Total (EF):				10.5

5. Estimate Environmental Complexity Factor (ECF)

Factor	Description	Weight	Assigned Value	Weightx Assigned Value
E1	Familiarity with development process used	1.5	3	4.5
E2	Application experience	0.5	3	1.5
E3	Object-oriented experience of team	1.0	2	2
E4	Lead analyst capability	0.5	5	2.5
E5	Motivation of the team	1.0	2	2
E6	Stability of requirements	2.0	1	2
E7	Part-time staff	-1.0	0	0
E8	Difficult programming language	-1.0	4	-4
Total (EF):				10.5

4. Estimate Environmental Complexity Factor (ECF)

$$\text{ECF} = 1.4 + (-0.03 * \text{EF})$$

$$\text{ECF} = 1.4 + (-0.03 * 10.5) = 1.085$$

5-7 Calculate UCP, Effort, Cost

Productivity Factor (PF)



$$\begin{aligned} \text{UCP} &= (\text{UUCW} + \text{UAW}) \times \text{TCF} \times \text{ECF} \\ &= (100 + 13) \times 1.02 \times 1.085 \\ &= \mathbf{125.06} \end{aligned}$$

$$\begin{aligned} \text{Effort} &= \text{UCP} \times \text{PF} \text{ (Hours/Use Case Point)} \\ &= 125.06 \times 28 \\ &= \mathbf{3501 \text{ Hours}} \end{aligned}$$

$$\begin{aligned} \text{Cost} &= \text{Effort} * \text{Average Cost/Hour} \\ &= 3501 * \$50/\text{hour} \\ &= \mathbf{\$168,500} \end{aligned}$$



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Image Credits

Image/Source	Copyright	Location	Notes
Use Case Point Estimation	https://en.wikipedia.org/wiki/Use_Case_Points	Slides 3-14	



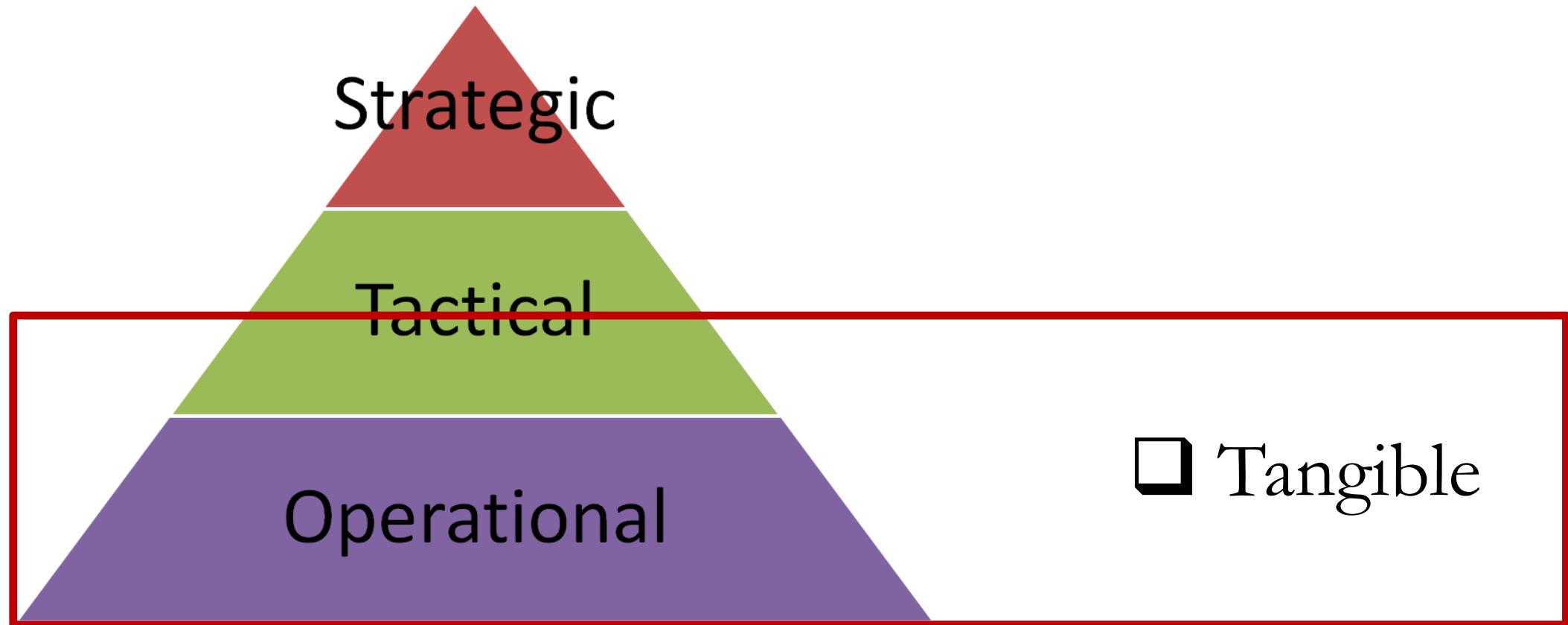
Information Systems Management: Key Principles and Practices

Course 1: IS/IT Governance
Module 2.1: Evaluating IT Investments

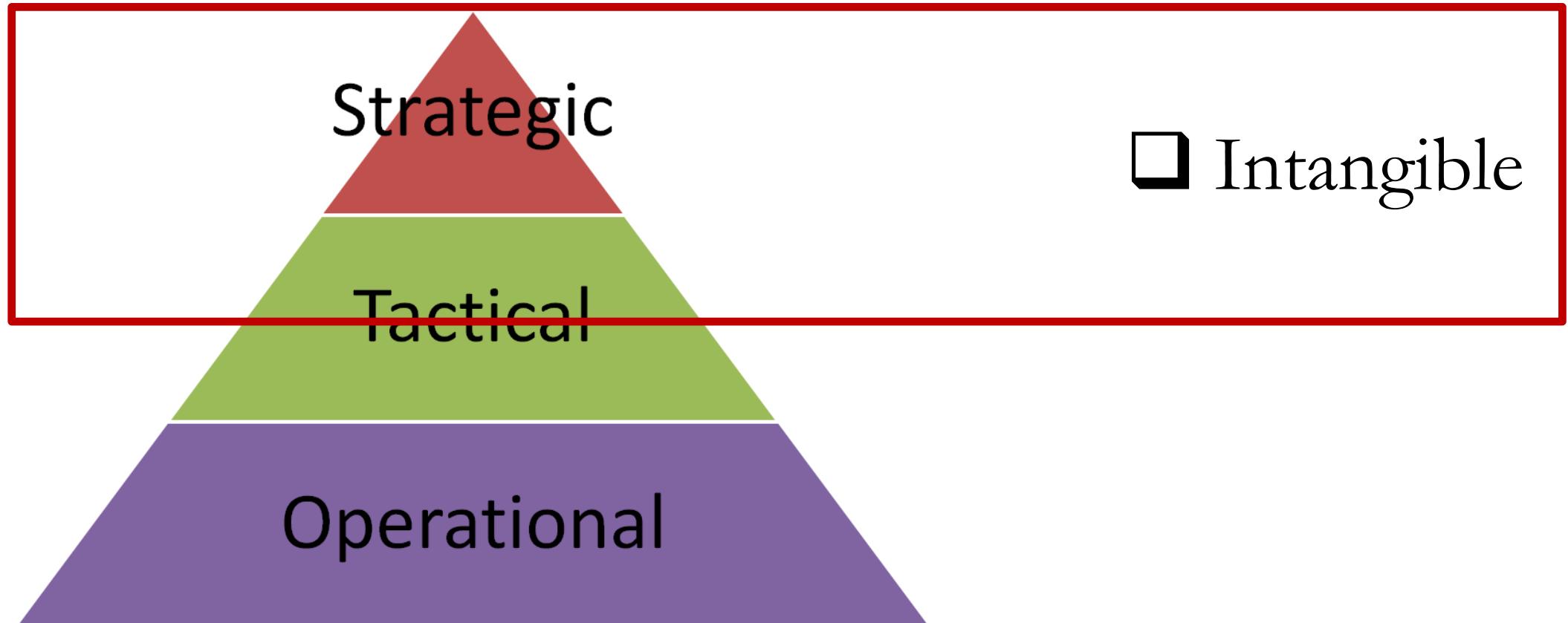
Lesson Overview and Goal

1. Intangible benefits

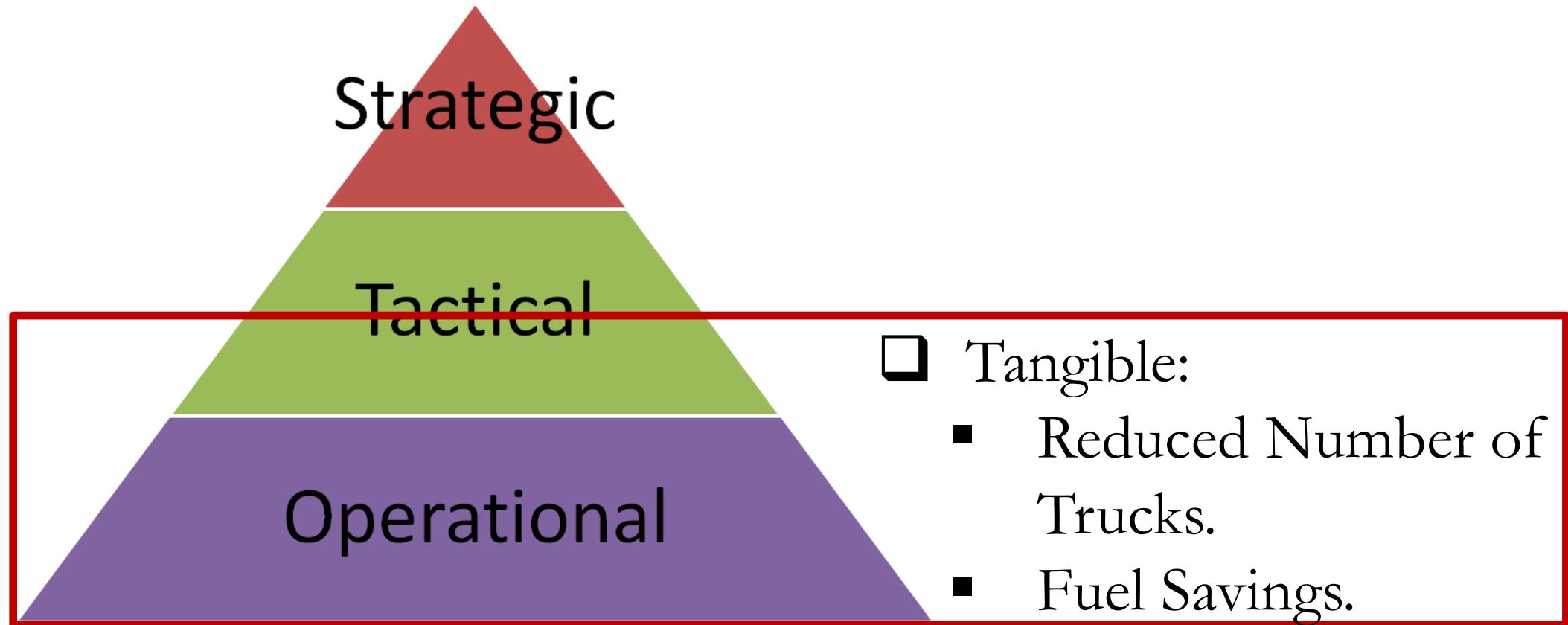
Types of Benefits



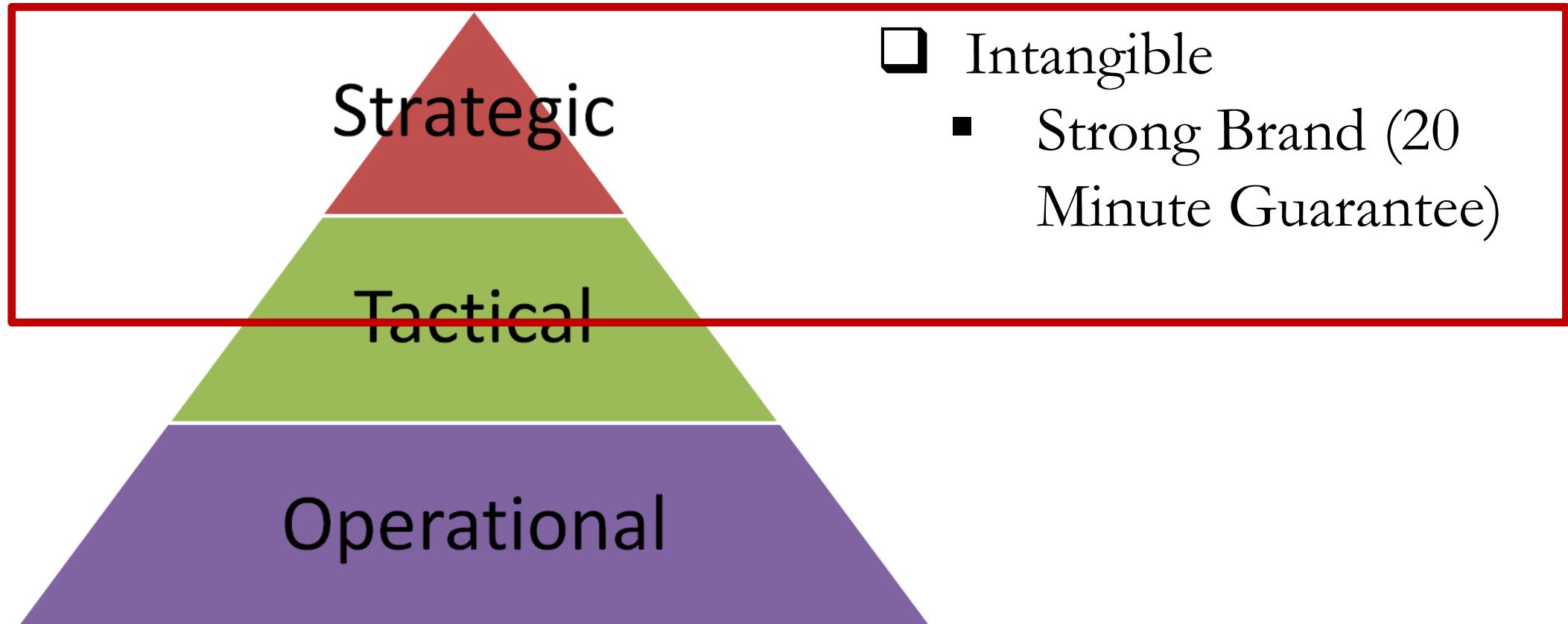
Types of Benefits



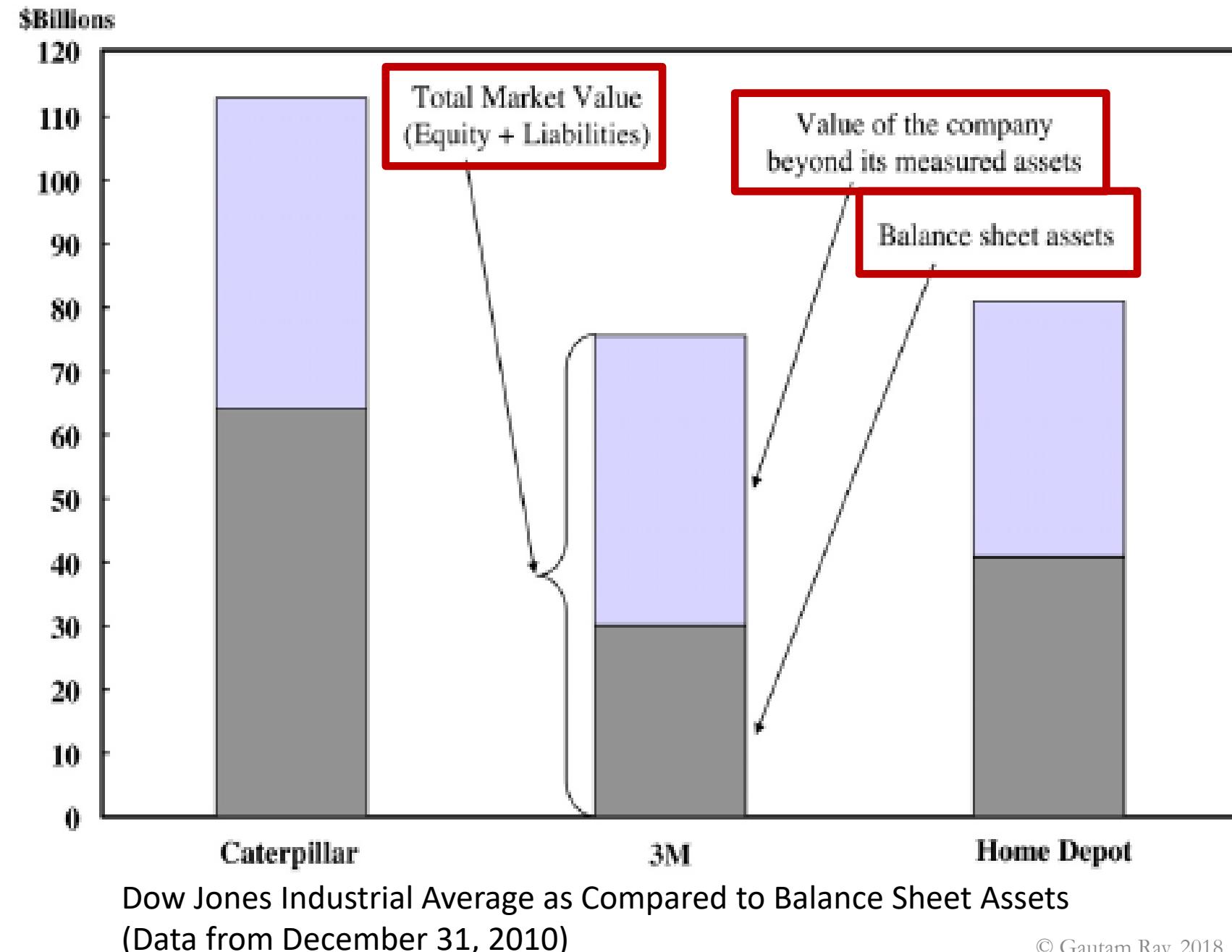
Types of Benefits



Types of Benefits



Market Value of Firms



Intangible Assets

“Assets are claims to future benefits, such as the rents generated by commercial property, interest payments derived from a bond, and cash flows from a production facility. An intangible asset is a claim to future benefits that does not have a physical or financial (a stock or a bond) embodiment. A patent, a brand, and a unique organizational structure (for example, an Internet-based supply chain) that generate cost savings are intangible assets.” – Intangibles: Management, Measurement, and Reporting by Lev, B. (2000)

Market Value as a Function of Assets of the Firm

Table 6. Market Value as a Function of the Assets of the Firm, 2003–2006

	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Information Technology (<i>IT</i>) – Hardware only Purchased Hardware (BHY 2002 measure)	11.31 (3.51)	8.75 (3.89)					
Information Technology (<i>IT</i>) – Capitalized only Capitalized Hardware, Capitalized Software			7.19 (1.39)	5.68 (1.52)	3.61 (1.01)		
Information Technology (<i>IT</i>) – All Hardware, All Software, Other Internal IT Assets, IT Consulting, IT-Related Training						1.53 (.41)	1.15 (.46)
Ordinary Assets (<i>K</i>)	1.20 (.08)	1.48 (.14)	1.22 (.08)	1.48 (.14)	1.48 (.14)	1.20 (.08)	1.48 (.14)
Other Assets (<i>F</i>)	1.12 (.09)	1.80 (.18)	1.09 (.10)	1.81 (.17)	1.81 (.18)	1.12 (.09)	1.80 (.18)
R&D (<i>R</i>)		1.65 (.23)		1.65 (.23)	1.64 (.23)		1.63 (.23)
Brand (<i>B</i>)		2.57 (1.45)		2.48 (1.38)	2.53 (1.41)		2.49 (1.44)
Number of Observations	508	508	508	508	508	508	508
Assumptions	Firm capitalized all purchased hardware only	Firm capitalized all purchased hardware only	Firm capitalized industry average of purchases and payroll for hardware & software	Firm capitalized industry average of purchases and payroll for hardware & software	Firm capitalized 100% of purchases and payroll for hardware & software	Virtually all IT spending included in IT assets (except leased hardware & software)	Virtually all IT spending included in IT assets (except leased hardware & software)

Note: All regressions are GLS, with correction for heteroskedasticity and serial correlation. Controls for sector, year, no R&D, and no advertising are included (where each control is multiplied by total balance sheet assets).

Information Technology Capabilities

HR Capabilities, Management Capability, Internal Communications, Communication with Suppliers, and Internet Capability

Table 4. Distribution of Capabilities (ITC)

Category	Meaning	Number of Observations
ITC_A	95 th percentile and above	24
ITC_B	85 th -95 th percentile	52
ITC_C	15 th -85 th percentile	356
ITC_D	5 th -15 th percentile	52
ITC_F	5 th percentile and below	24
Total		508

Market Value as a Function of Assets of the Firm and ITC

Table 8. Market Value as a Function of the Assets of the Firm and ITC

	(1)	(2)
Information Technology (IT) – All Hardware, All Software, Other Internal Assets, IT Consulting, IT-Related Training	.67 (.36)	.71 (.36)
Ordinary Assets (K)	1.58 (.13)	1.51 (.13)
Other Assets (F)	1.94 (.17)	1.91 (.16)
R&D (R)	1.56 (.22)	1.58 (.23)
Brand (B)	3.24 (1.26)	3.58 (1.34)
ITC_A * IT		14.63 (5.09)
ITC_B * IT		9.77 (1.94)
ITC_D * IT		-.75 (1.67)
ITC_F * IT		-5.21 (3.93)
ITC_A * Employment	119.81 (57.26)	
ITC_B * Employment	78.11 (15.10)	
ITC_D * Employment	-4.66 (22.44)	
ITC_F * Employment	-171.89 (49.62)	
Number of Observations	508	508

Note: All regressions are GLS, with correction for heteroskedasticity and serial correlation. Controls for sector, year, no R&D, and no advertising are included (where each control is multiplied by total balance sheet assets). Employment levels are reported in thousands.

Market Value as a Function of Assets of the Firm and ITC

Table 8. Market Value as a Function of the Assets of the Firm and ITC

	(1)	(2)
Information Technology (IT) – All Hardware, All Software, Other Internal Assets, IT Consulting, IT-Related Training	.67 (.36)	.71 (.36)
Ordinary Assets (K)	1.58 (.13)	1.51 (.13)
Other Assets (F)	1.94 (.17)	1.91 (.16)
R&D (R)	1.56 (.22)	1.58 (.23)
Brand (B)	3.24 (1.26)	3.58 (1.34)
ITC_A * IT		14.63 (5.09)
ITC_B * IT		9.77 (1.94)
ITC_D * IT		-.75 (1.67)
ITC_F * IT		-5.21 (3.93)
ITC_A * Employment	119.81 (57.26)	
ITC_B * Employment	78.11 (15.10)	
ITC_D * Employment	-4.66 (22.44)	
ITC_F * Employment	-171.89 (49.62)	
Number of Observations	508	508

Note: All regressions are GLS, with correction for heteroskedasticity and serial correlation. Controls for sector, year, no R&D, and no advertising are included (where each control is multiplied by total balance sheet assets). Employment levels are reported in thousands.



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Image Credits

<u>Image/Source</u>	<u>Copyright</u>	<u>Location</u>	<u>Notes</u>
Information Technology and Intangible Assets	Saunders, A., & Brynjolfsson, E. (2016). Valuing Information Technology Related Intangible Assets. <i>MIS Quarterly</i> , 40(1) pp. 83-110.	Slides 3-7	
Intangible Assets	Lev, B. (2000). <i>Intangibles: Management, measurement, and reporting</i> . Brookings Institution Press.	Slide 8	
Table 6: Market Value as a Function of the Assets of the Firm 2003-06	Saunders, A., & Brynjolfsson, E. (2016). Valuing Information Technology Related Intangible Assets. <i>MIS Quarterly</i> , 40(1) pp. 83-110.	Slides 9-12	



Information Systems Management: Key Principles and Practices

Course 1: IS/IT Governance
Module 2.2: Evaluating IT Investments

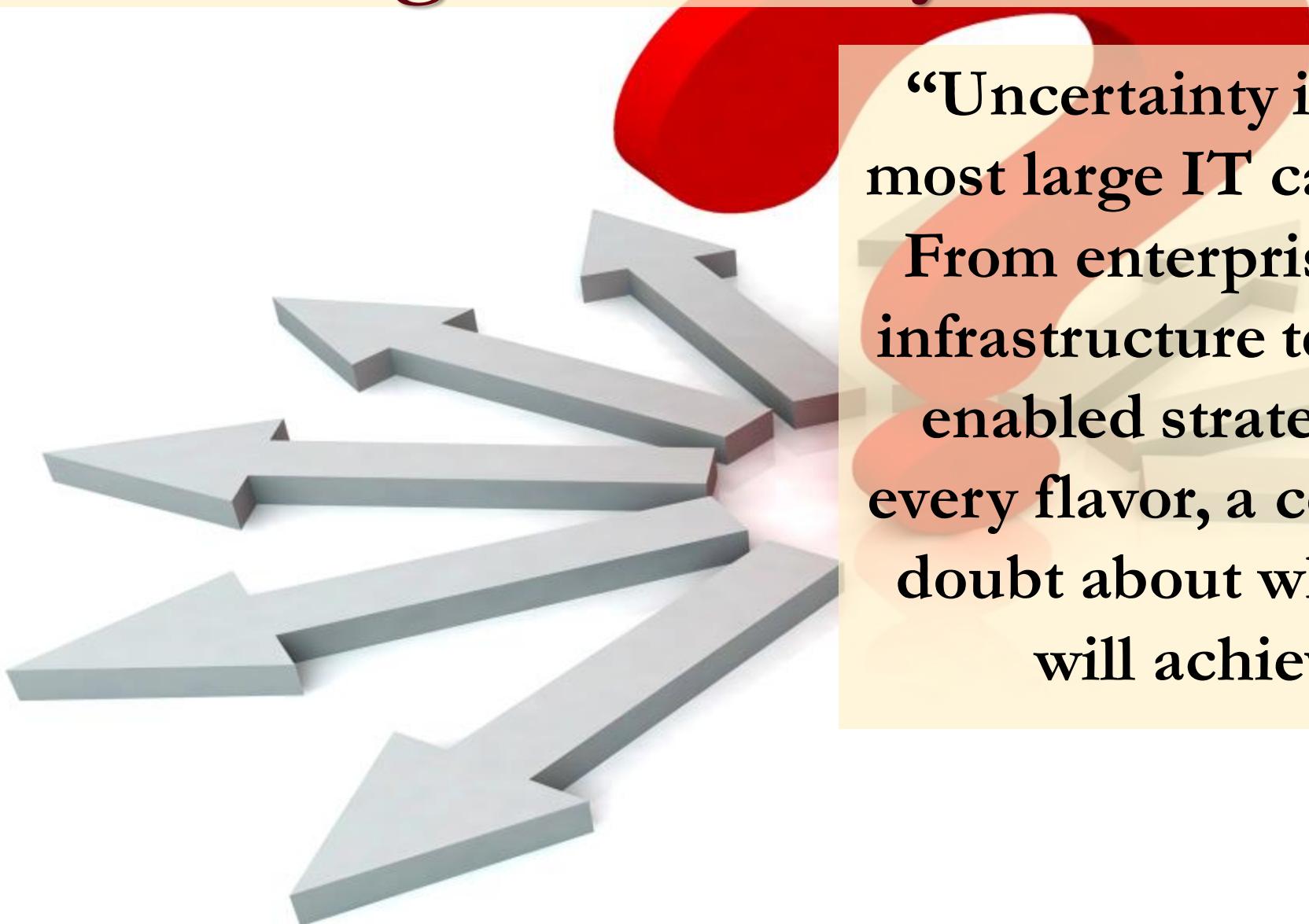
Module Overview and Goals

1. Evaluate Individual IT Investments
2. Incorporating Risk in IT Investment Evaluation

Lesson Overview and Goals

- Incorporating Risk in IT Investment Evaluation
 1. Discover different types of IT project risks

Taking uncertainty into the equation



“Uncertainty is a fact of life for most large IT capital investments. From enterprise applications to infrastructure technologies to IT-enabled strategic initiatives of every flavor, a common element is doubt about whether the project will achieve its goals.”

Risks in IT Projects

- Development Risk:** Can the IS unit develop the proposed system.
 - Knowledge of the domain, Maturity of the technology platform, the ability to implement the requirements in the available technical platform.
- Organizational Risk:** Can the organization assimilate the organizational changes.
 - Ability to specify requirements, magnitude of and the readiness to implement change, Top management support.
- Market Risk:** How the changes in the environment will affect organizational returns from the new system.
 - Competitor preemption or imitation, customer/supplier acceptance, Emergence of superior technological alternatives.

Risks in IT Projects

- Development Risk:** Can the IS unit develop the proposed system.
 - Knowledge of the domain, Maturity of the technology platform, the ability to implement the requirements in the available technical platform.
- Organizational Risk:** Can the organization assimilate the organizational changes.
 - Ability to specify requirements, magnitude of and the readiness to implement change, Top management support.
- Market Risk:** How the changes in the environment will affect organizational returns from the new system.
 - Competitor preemption or imitation, customer/supplier acceptance, Emergence of superior technological alternatives.

Risks in IT Projects

- Development Risk:** Can the IS unit develop the proposed system.
 - Knowledge of the domain, Maturity of the technology platform, the ability to implement the requirements in the available technical platform.
- Organizational Risk:** Can the organization assimilate the organizational changes.
 - Ability to specify requirements, magnitude of and the readiness to implement change, Top management support.
- Market Risk:** How the changes in the environment will affect organizational returns from the new system.
 - Competitor preemption or imitation, customer/supplier acceptance, Emergence of superior technological alternatives.

Taking risk into the equation - What do companies do?

- Downplay the level of risk
- Penalizing IT projects with a high hurdle rate
- Treat setbacks on IT projects as inadequacies of the project team

Limitations of the NPV Method

- The NPV rule makes some debatable assumptions:
 - Investment is reversible, it can be undone if conditions change.
 - It is a now-or-never decision.
- Most investments are irreversible, but can be delayed.
- Investment opportunities are *Options*: Rights but not obligations to take some action in the future.



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Image Credits

<u>Image/Source</u>	<u>Copyright</u>	<u>Location</u>	<u>Notes</u>
Limitations of the NPV Method	Dixit, A. K.; Pindyck, R. S. The Options Approach to Capital Investment. <i>Harvard Business Review</i> . 73, 3, 105-115, May 1995.	Slide 9	



Information Systems Management: Key Principles and Practices

Course 1: IS/IT Governance
Module 2.2: Evaluating IT Investments

Lesson Overview and Goals

- Incorporating Risk in IT Investment Evaluation
 1. Discover different types of IT project risks
 2. Real Options Approach to IT Investment

Financial Options

- *HiTech* stock is selling at \$20/share. Price is expected to rise significantly in the near future because of the demand for the company's innovative products. However, there is also market uncertainty that indicates a possible sharp drop in the stock price.
- As an investor, you can
 - Purchase shares of the stock today
 - Purchase a call (buy) option for \$2 that gives you the right to buy the stock one year from now at \$25

Options Approach to IT Investment

- Essential Idea: There is uncertainty and it can be resolved in one of two ways:
 - By waiting.
 - A small experiment.
- Option holder can participate in the upside, but limit their losses to the cost of acquiring the option.
 - Higher uncertainty means higher upside.
 - However, the downside is capped at zero.

Options Approach to IT Investment

Type of Option	Definition
Defer	A decision to invest can be put off for some period without reducing the benefits.
Stage	A project can be divided into distinct phases where pursuit of each stage is contingent on reassessing costs and benefits as the preceding stage is completed.
Growth	An initial baseline investment opens the door to pursue a variety of potential follow-on projects.
Abandon	A project can be terminated midstream and remaining project resources redeployed.

Options Approach to IT Investment

Type of Option	Definition
Defer	A decision to invest can be put off for some period without reducing the benefits.
Stage	A project can be divided into distinct phases where pursuit of each stage is contingent on reassessing costs and benefits as the preceding stage is completed.
Growth	An initial baseline investment opens the door to pursue a variety of potential follow-on projects.
Abandon	A project can be terminated midstream and remaining project resources redeployed.

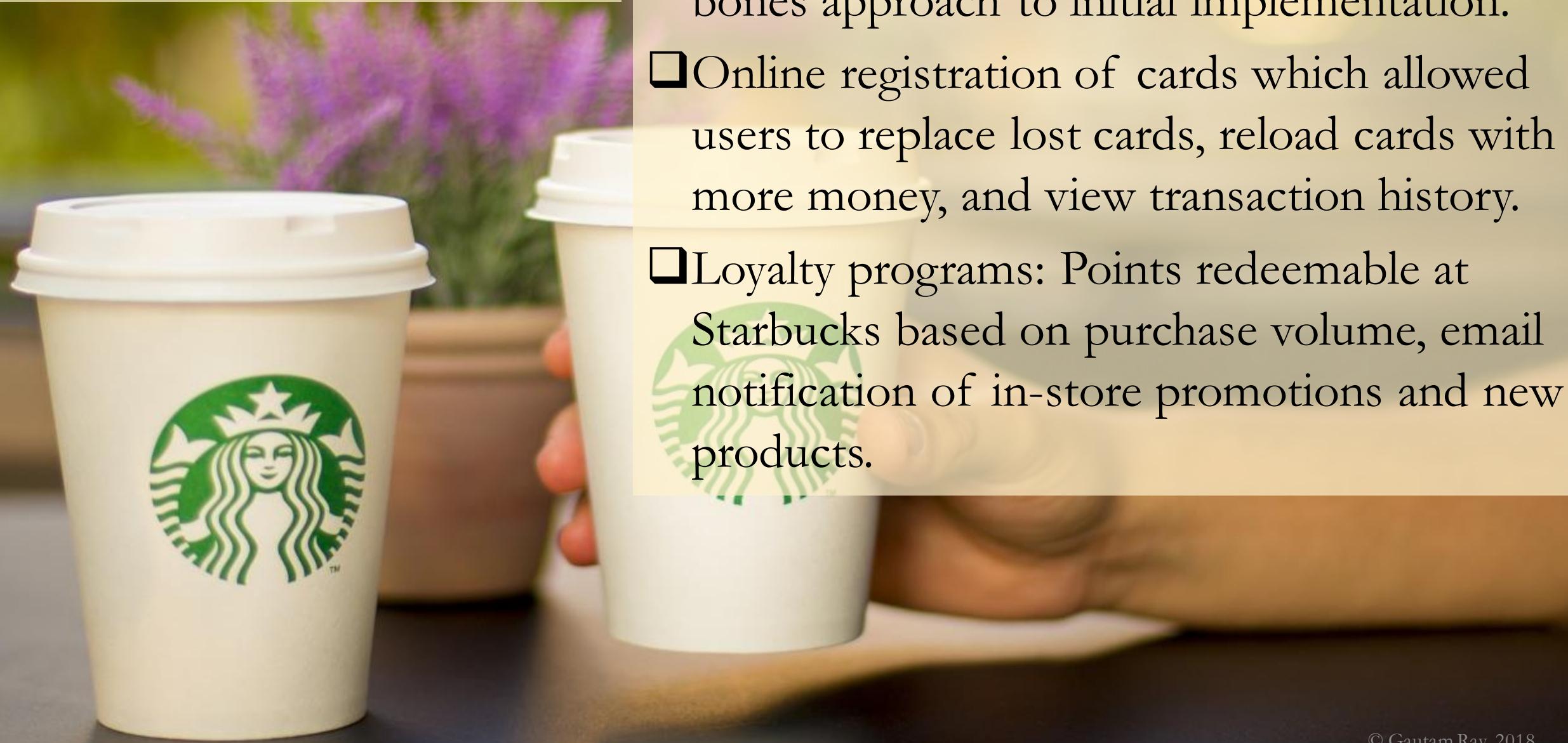
Options Approach to IT Investment

Type of Option	Definition
Defer	A decision to invest can be put off for some period without reducing the benefits.
Stage	A project can be divided into distinct phases where pursuit of each stage is contingent on reassessing costs and benefits as the preceding stage is completed.
Growth	An initial baseline investment opens the door to pursue a variety of potential follow-on projects.
Abandon	A project can be terminated midstream and remaining project resources redeployed.

Options Approach to IT Investment

Type of Option	Definition
Defer	A decision to invest can be put off for some period without reducing the benefits.
Stage	A project can be divided into distinct phases where pursuit of each stage is contingent on reassessing costs and benefits as the preceding stage is completed.
Growth	An initial baseline investment opens the door to pursue a variety of potential follow-on projects.
Abandon	A project can be terminated midstream and remaining project resources redeployed.

Starbucks Example



- Pre-paid card to speed checkout. Bare-bones approach to initial implementation.
- Online registration of cards which allowed users to replace lost cards, reload cards with more money, and view transaction history.
- Loyalty programs: Points redeemable at Starbucks based on purchase volume, email notification of in-store promotions and new products.

```
  "timestamp": "2017-06-03T18:42:18.018", "deltastartMillis": "0", "method": "POST", "sizeChars": "5022", "message": "Duration Log", "durationMillis": "36", "webParams": "null", "class": "com.orgmanager.handlers.RequestHandler", "durationInMillis": "36"}, {"timestamp": "2017-06-03T18:43:335.036", "deltastartMillis": "0", "method": "GET", "sizeChars": "4845", "message": "Duration Log", "durationMillis": "7"}, {"timestamp": "2017-06-03T18:46:921.000", "deltastartMillis": "0", "method": "POST", "sizeChars": "10190", "message": "Duration Log", "durationMillis": "23"}, {"timestamp": "2017-06-03T18:42:18.018", "deltastartMillis": "0", "method": "POST", "sizeChars": "5022", "message": "Duration Log", "durationMillis": "36"}, {"timestamp": "2017-06-03T18:43:335.036", "deltastartMillis": "0", "method": "GET", "sizeChars": "4845", "message": "Duration Log", "durationInMillis": "36"}, {"timestamp": "2017-06-03T18:46:921.000", "deltastartMillis": "0", "method": "POST", "sizeChars": "10190", "message": "Duration Log", "durationMillis": "23"}]
```



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Image Credits

<u>Image/Source</u>	<u>Copyright</u>	<u>Location</u>	<u>Notes</u>
Options Thinking	Fichman, R. G., Keil, M., & Tiwana, A. (2005). Beyond valuation:“Options thinking” in IT project management. <i>California Management Review</i> , 47(2), 74-96.	Slides 5-8	
Starbucks Example	Howe, R. (2003). At Starbucks, the future is in plastic. <i>Business 2.0</i> , 4(7), 56-56.	Slide 9	



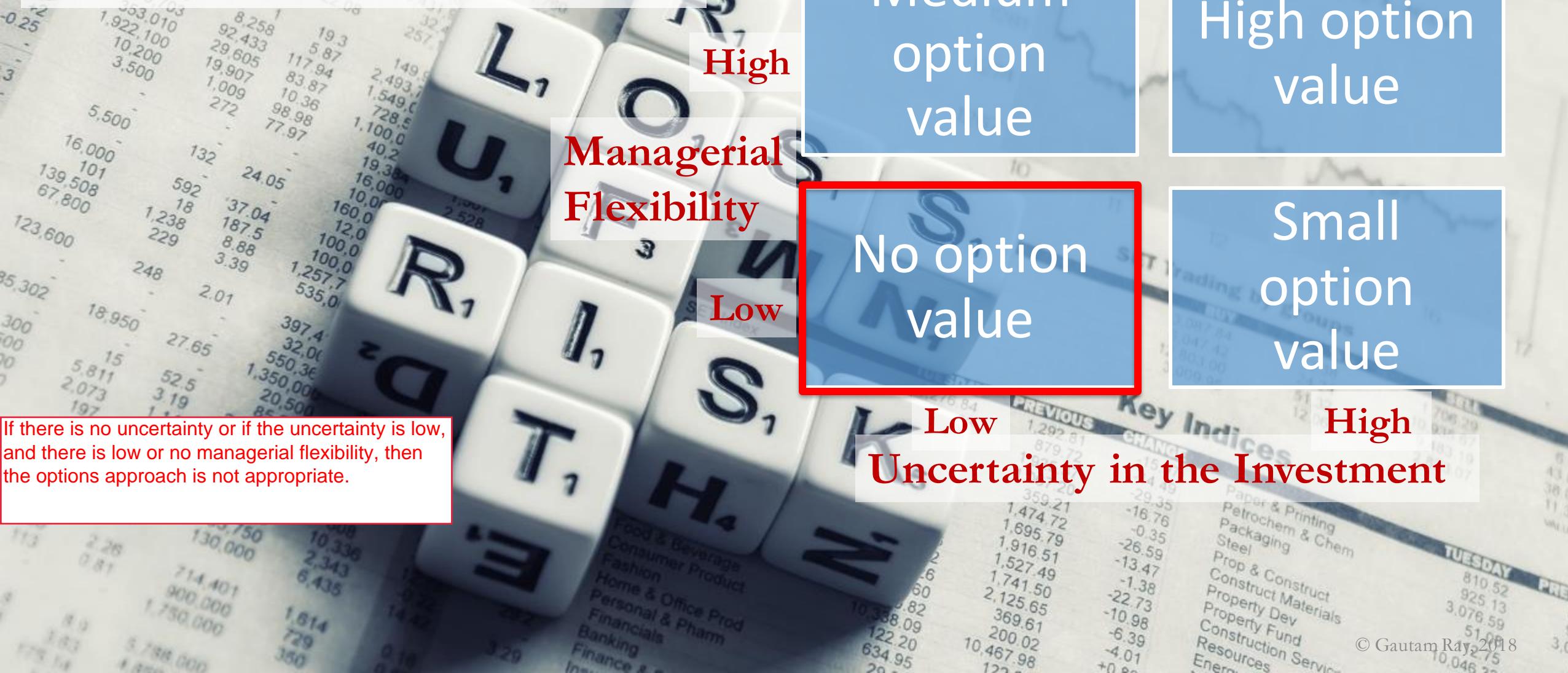
Information Systems Management: Key Principles and Practices

Course 1: IS/IT Governance
Module 2.2: Evaluating IT Investments

Lesson Overview and Goals

- Incorporating Risk in IT Investment Evaluation
 1. Discover different types of IT project risks
 2. Real Options Approach to IT Investment

When does real options provide the most value?



When does real options provide the most value?

If an investment has high uncertainty associated with it and it also has high managerial flexibility associated with it (i.e it can be deferred or it can be done in stages), then the options approach is more appropriate to evaluate an investment.

High
Managerial
Flexibility

Low

Medium
option
value

No option
value

Low

High option
value

Small
option
value

High

Uncertainty in the Investment

A framework for Evaluating IT Investments

Technology Scope

Business Solution

Shared Infrastructure

Process Improvement

Experiments

Renewal

Transformation

Short-term profitability

Long-term growth

Strategic Objective

Renewal

- Upgrade infrastructure to improve reliability, expand functionality, and reduce maintenance costs.

- NPV Based Business Case

In a renewal investment, the company is trying to upgrade the infrastructure to improve reliability, expand functionality, and reduce maintenance costs. So, the objective of this investment is to improve profitability right now. That makes the NPV based business case the more appropriate method to evaluate the investment.

A framework for Evaluating IT Investments

Technology Scope

Business Solution

Shared Infrastructure

Process Improvement

Experiments

Renewal

Transformation

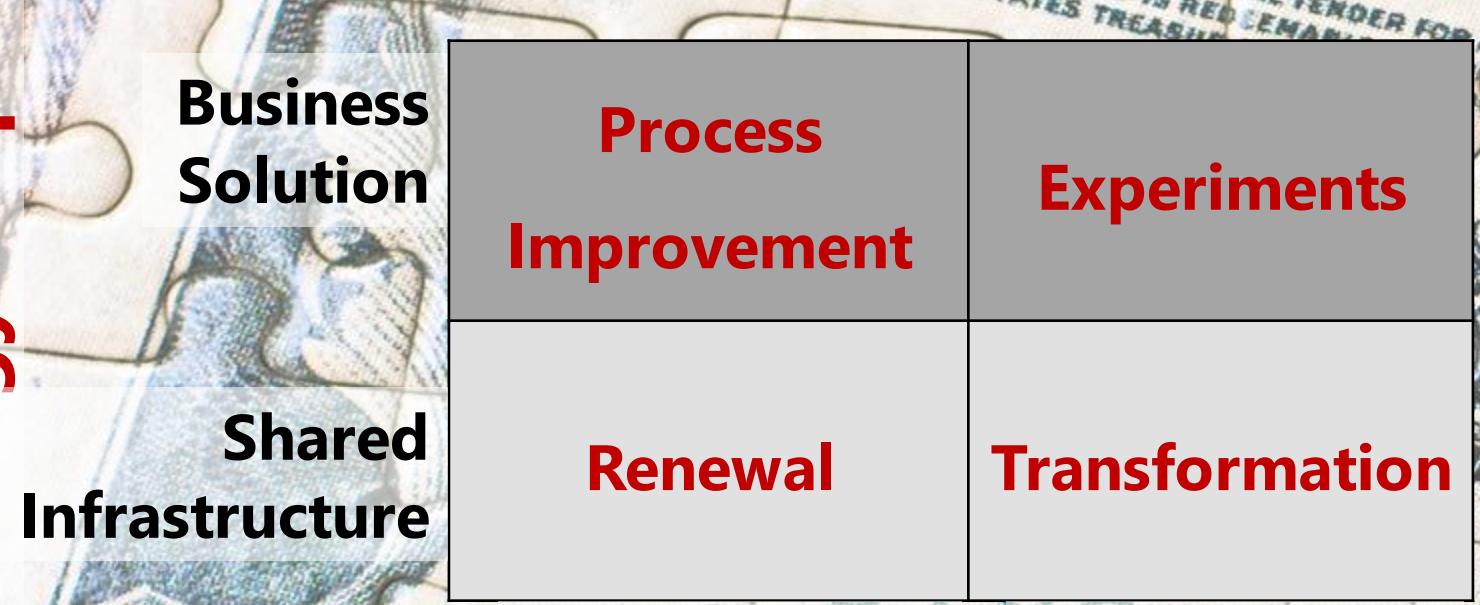
Short-term profitability
Long-term growth
Strategic Objective

- Process Improvement
 - New Application to serve specific business need.
 - NPV Based Business Case

In a process improvement project, the client is trying to implement a new application to serve a specific business name. Since the objective is to improve profitability right now, the NPV based business case is the right way to evaluate the investment.

A framework for Evaluating IT Investments

Technology Scope



Short-term profitability
Long-term growth
Strategic Objective

- Transformation
 - Next generation infrastructure to shape competition in future
 - Real Options analysis.

A transformation project is a next generation infrastructure to shape competition in the future. So, this is an investment that has a long-term effect, which makes the investment very uncertain. Thus, the real options approach is more appropriate to evaluate this investment

A framework for Evaluating IT Investments

Technology Scope

Business Solution

Shared Infrastructure

Process Improvement

Renewal

Experiments

Transformation

Short-term profitability
Long-term growth
Strategic Objective

- Experiments
 - Next generation technologies to shape competition in future
 - Real Options analysis.

Experiments are projects about the next generation technologies to shape competition in the future. Again, this project has a long-term orientation with an uncertain outcome which makes the real options approach more appropriate method to evaluate the investment.

Pitfalls of Termination

What happens if you terminate a project midway? The first point is that, it is difficult to terminate a project midway because of what is referred to as escalation of commitment. That is, once a project is started, often projects take a life of their own and once started can't be stopped. Similarly, termination may hurt the morale of employees, specifically team members who are involved in the project. Finally, termination may affect the credibility of team members because outsiders may not be aware of why the project was terminated. They may confuse two things. That is, they may wonder whether the project was terminated because of some factors beyond the control of the project team or the project was terminated because of the failure of the project team.

- Escalation of commitment: Projects take a life of their own, once started can't be stopped.
- Terminating projects may hurt morale.
- Loss of credibility owing to ambiguity about termination due to failure of project team or factors beyond control.

Pitfalls of Deferral

When you defer a project, you expose yourself to some risks, primarily the risk of competitive preemption. A client may defer a project because of risk. However, this deferral exposes the client to competitors who may not be as deliberate and go ahead and execute a risky project and are successful. Finally, deferral can be problematic when direct experience with the innovation is needed to resolve the uncertainty. That is, you defer a project because you are uncertain, but the only way to resolve the uncertainty may be direct experience with the innovation. In other words, deferral may lead to a loss of innovation ability.

- Benefits may erode with time. For example due to competitive preemption.
- Sometime firms may need direct experience with innovation to be able to resolve the uncertainty. Deferral may lead to loss of innovation ability.



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Image Credits

<u>Image/Source</u>	<u>Copyright</u>	<u>Location</u>	<u>Notes</u>
A framework for Evaluating IT Investments	Ross, J. W., & Beath, C. M. (2002). Beyond the business case: New approaches to IT investment. <i>MIT Sloan management review</i> , 43(2), 51.	Slides 5-8	
Pitfalls of Termination and Deferral	Fichman, R. G., Keil, M., & Tiwana, A. (2005). Beyond valuation: “Options thinking” in IT project management. <i>California Management Review</i> , 47(2), 74-96.	Slides 9-10	



Information Systems Management: Key Principles and Practices

Course 1: IS/IT Governance
Module 3.1: Evaluating IT
Investments (continued)

Module Overview and Goals

1. Evaluate a Portfolio of Individual IT Investments
2. IT Chargeback

Lesson Overview and Goals

1. Definition of IT investment portfolio
2. Characteristics of Different IT Asset Classes
3. Strategic Orientation and IT Investment Portfolio Management

Information Technology Portfolio Management

- Select the right set of IT projects that are aligned with the organization's strategy and improve returns on IT investments.

IT portfolio management is about selecting the right set of IT projects that are aligned with the organization's strategy and improve returns on IT investments, that is, we want to not only do projects right, but select and do the right projects.

Information Technology Portfolio Management (ITPM)

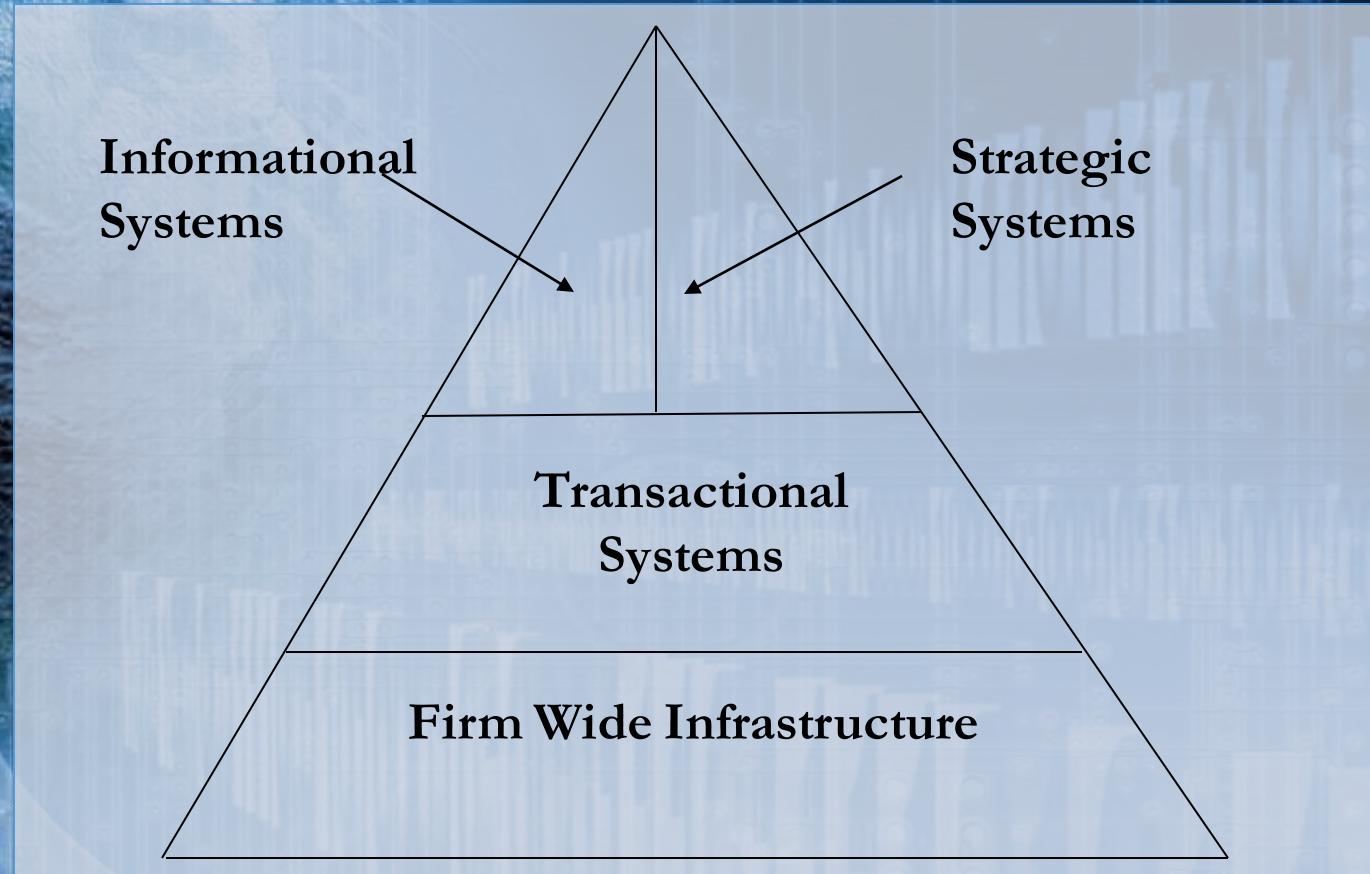
- Not only do projects right but select and do the right projects. Flawless project execution is meaningless if the right projects are not being tackled.
- Project management ensures that projects are done right, while ITPM ensures that the right projects are done.
- Focus on all IT projects across an organization providing a unified view of IT spending (value and risks) in its entirety.



IT Investment Portfolio

The IT investment portfolio consists of four elements:

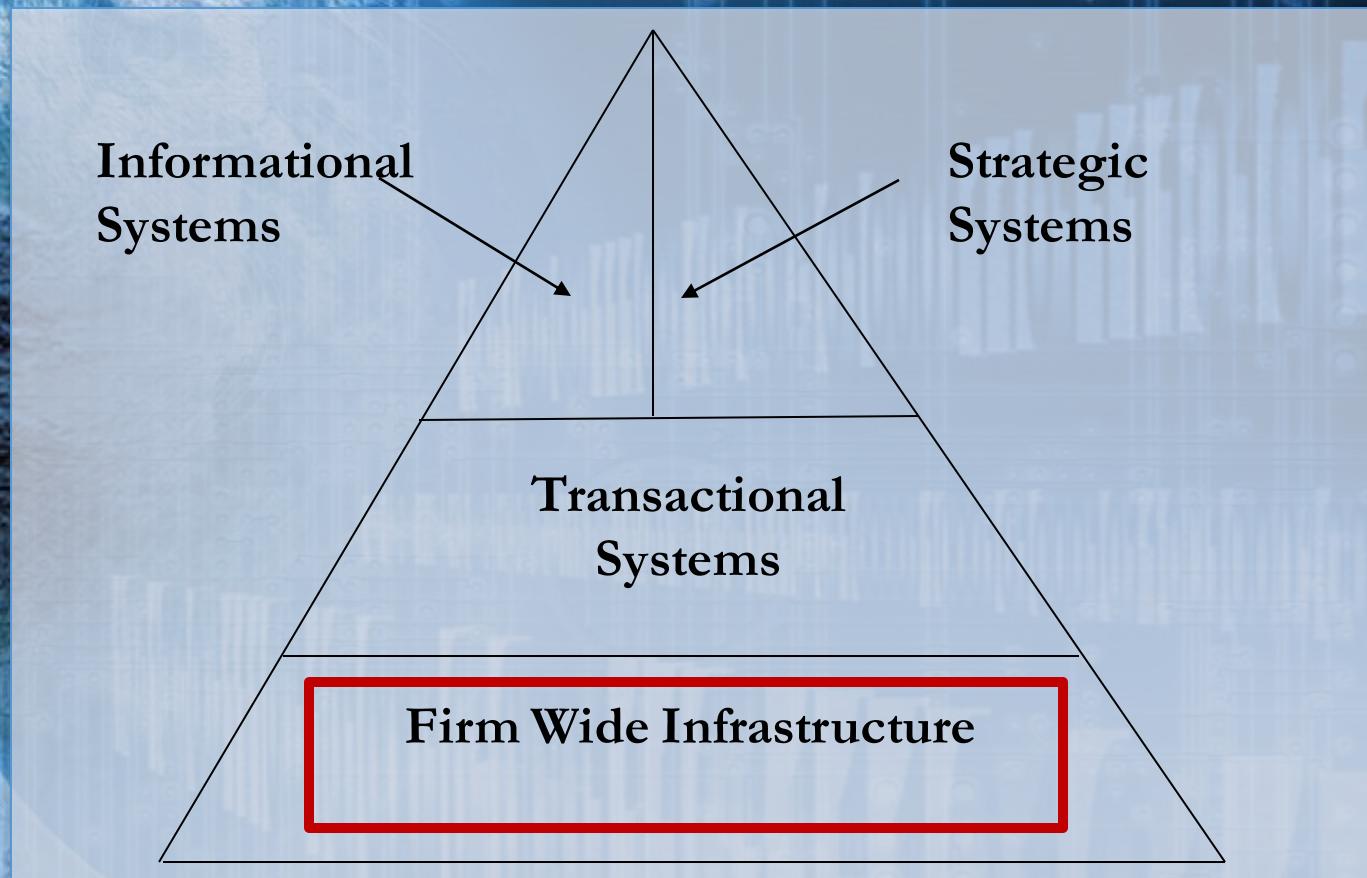
- 1- firm wide infrastructure
- 2- transactional systems
- 3- informational systems
- 4- strategic systems.



IT Investment Portfolio

Firm wide infrastructure is the foundation of network, computing, and database services on which IT applications run. The goal of firm wide infrastructure is

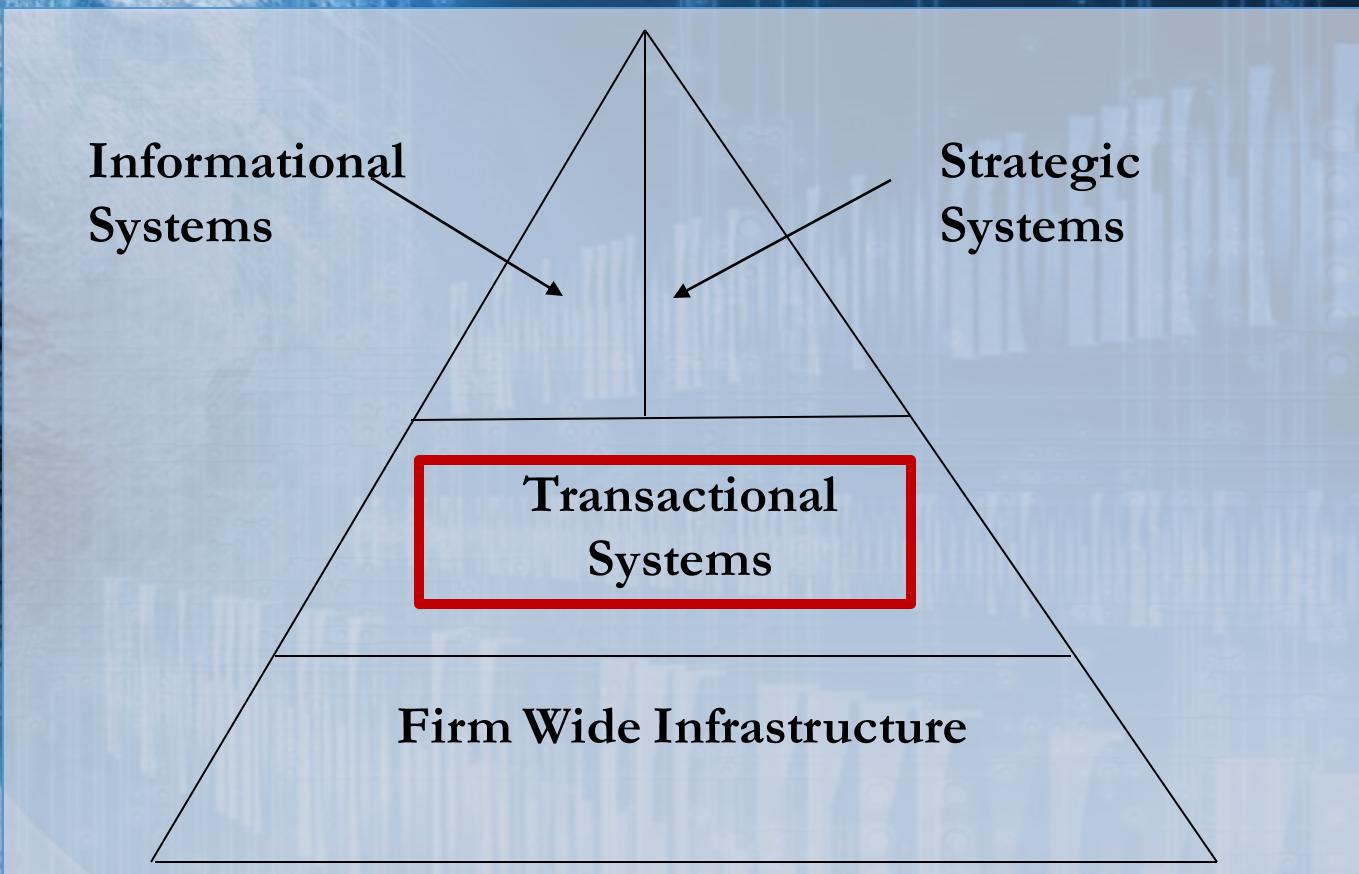
- integration, that is, allow different applications to be integrated
- flexibility, that is, provide the flexibility so that applications can be developed and implemented faster, and
- reduced IT cost through standardization of different elements of the infrastructure.



IT Investment Portfolio

Transactional systems process basic business transactions cost effectively. So, for an airline, reservations, ticketing, checking-in, aircraft and crew scheduling, and aircraft maintenance, these are transactions that happen repeatedly.

The goal of transactional systems is to process basic transactions cost effectively, that is, reduce costs through increased throughput.

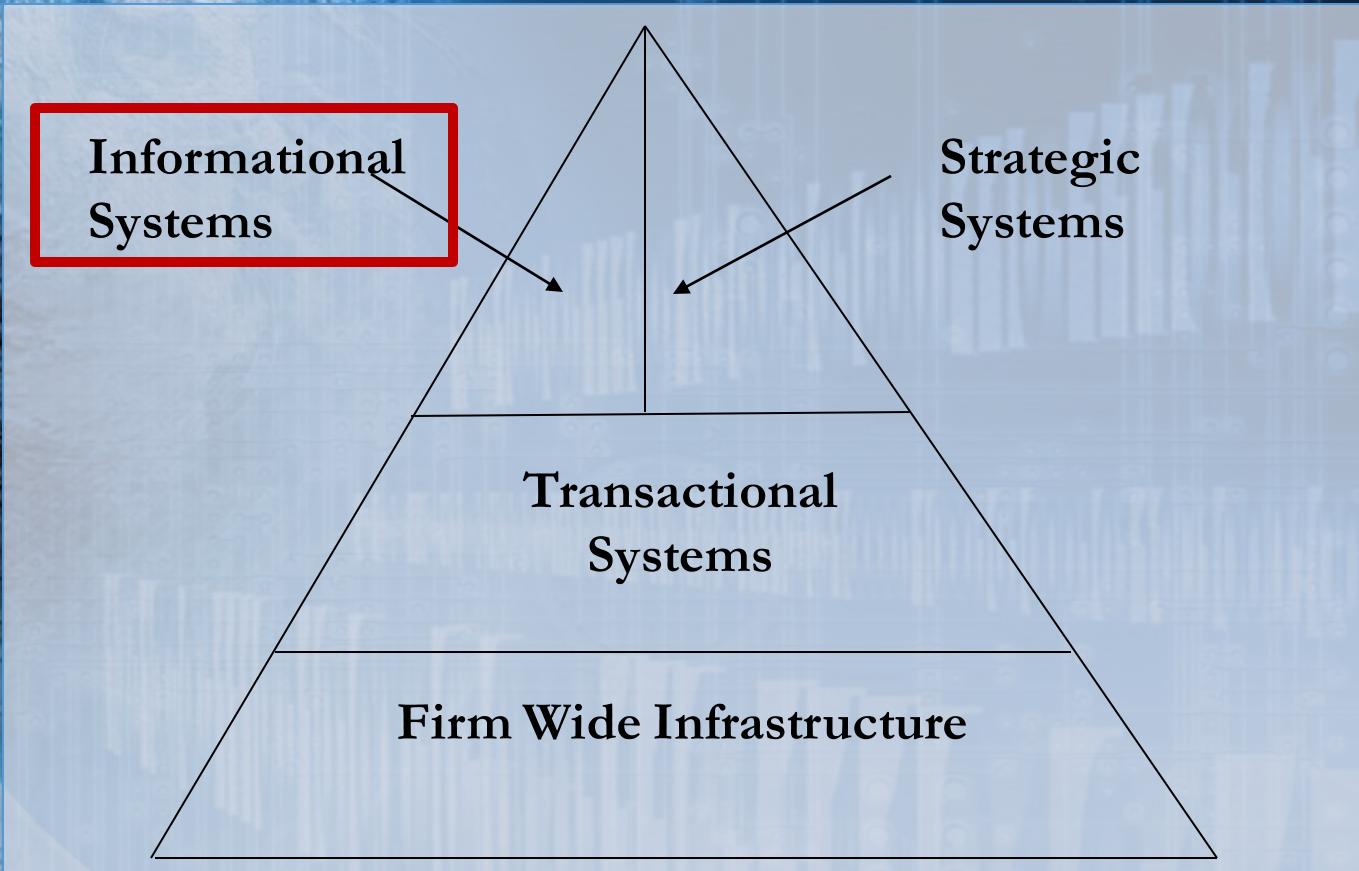


IT Investment Portfolio

Informational systems provide aggregated information, that is, information aggregated from transactional systems to monitor and control the performance of the organization. For example, for an airline, applications that identify which routes have high occupancy and need more flights versus which routes have low occupancy and need fewer flights is an example of an informational system.

The goal of informational system is:

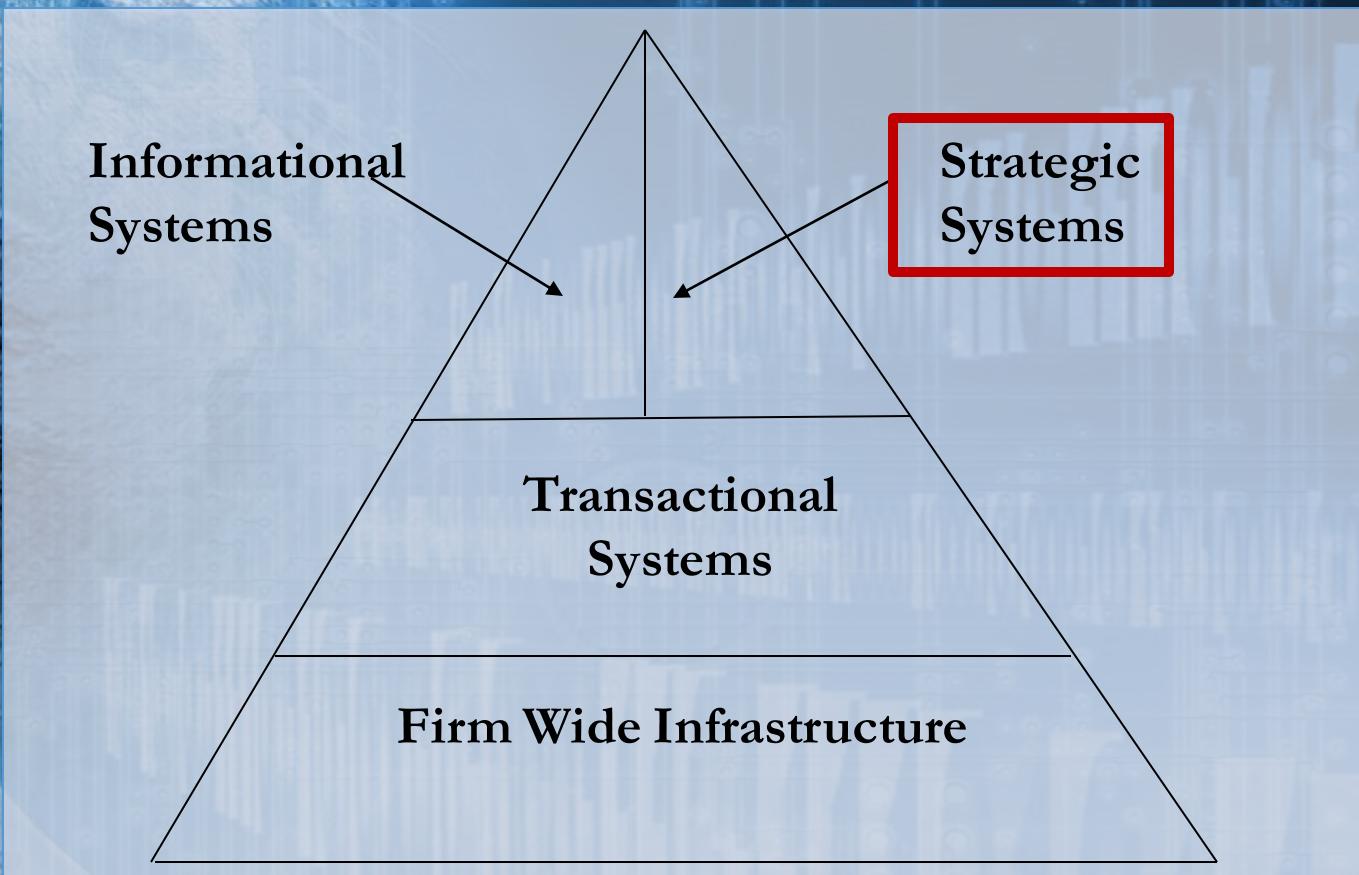
- better information,
- increased control, and
- improved quality.



IT Investment Portfolio

Strategic Systems are applications that provide a competitive advantage. They either provide a competitive advantage, or they reduce a competitive disadvantage. For an airline, applications to identify valuable customers so that they can be provided with differentiated service to improve their loyalty is an example of strategic systems.

The goal of strategic systems is to increase sales, improve competitive advantage, or reduce competitive disadvantage, that is, improve the market positioning of the firm.



Firm Wide Infrastructure



- The foundation of network, computing, and database services on which IT applications (specified below) run.
 - **Goals:** Integration, flexibility, reduced IT costs through standardization.

Transactional Systems



- ❑ Process basic business transactions cost effectively.
 - **Example:** Reservations, ticketing, checking-in, aircraft and crew scheduling, and aircraft maintenance.
 - **Goal:** Reduced costs through increased throughput.

Informational Systems

- ❑ Provide aggregated information (aggregated from the transactional systems) to monitor and control the performance of the organization.
 - **Example:** Which routes have high (low) occupancy and need more (fewer) flights.
 - **Goal:** Better information, increased control, improved quality.

Strategic Systems

- Applications that provide a competitive advantage.
 - **Example:** Applications to identify valuable customers, so that they can be provided with differentiated services to improve their loyalty.
 - **Goal:** Increases sales, competitive advantage, market positioning.



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Image Credits

Image/Source	Copyright	Location
Information technology portfolio management	Lane, D. (2011). <i>The Chief Information Officer's Body of Knowledge: People, Process, and Technology</i> (Vol. 571). John Wiley & Sons.	Slides 4-5
IT Investment Portfolio	Weill, P., & Broadbent, M. (1998). <i>Leveraging the new infrastructure: how market leaders capitalize on information technology</i> . Harvard Business Press.	Slides 6-14



Information Systems Management: Key Principles and Practices

Course 1: IS/IT Governance
Module 3.1: Evaluating IT
Investments (continued)

Lesson Overview and Goals

1. Definition of IT investment portfolio
2. Characteristics of Different IT Asset Classes
3. Strategic Orientation and Portfolio Management

IT Investment Portfolio

- ❑ Why do two firms with the same amount of IT capital perform differently?
 - It is likely that they are investing in different types of technology with different goals.

IT Investment Portfolio

- Data from 147 large publicly traded US firms from 1999-2002
- Examine the relationship between the intensity of each class of IT assets and four measures of performance:
 - Profitability (ROA)
 - Market Valuation (Tobin's Q)
 - Operational Performance (Net Margin, Cost of Goods Sold)
 - Innovation (Sales from New and Modified products)

Firm Wide Infrastructure

(First IT Asset Class)



- Investments made in anticipation of future business needs.
- Expect to be associated with short-term costs, lower short-term profitability, and higher profitability and performance in the long-run.

Firm Wide Infrastructure

(First IT Asset Class)

Firm Wide Infrastructure investments made in 2001 are negatively associated with the return on assets (ROA) in 2001, and net margin in 2001 and 2002. That is, Firm Wide Infrastructure reduces profitability in the short run.

They also find that Firm Wide Infrastructure disrupts short-term efforts at product innovation as measured as revenue from modified products in 2001, that is, Firm Wide Infrastructure causes some disruption that affects innovation in the short term.

However, they find that Firm Wide Infrastructure investments are positively related with Tobin's Q, that is Firm Wide Infrastructure is an investment that is rewarded by the market. So, in some sense, Firm Wide Infrastructure has long-term payoff.

- Investments made in 2001 are negatively associated with ROA in 2001 and net margin in 2001 and 2002.
- Infrastructure also disrupts short-term efforts at product innovation as measured as revenue from Modified products in 2001.
- The association with Tobin's Q is positive in 2001.

Transactional Systems

(Second IT Asset Class)

- Automated processes to reduce costs of standard repetitive processes.
- Transactional systems are associated with lower cost of goods sold but not with more innovation.

Informational Systems

(Third IT Asset Class)

- Tighten reporting and control functions to reduce costs, identify new opportunities for revenue generation, and improve profitability.

- Informational systems are positively related with ROA and net margin in 2001 and 2002.

Strategic Systems

(Fourth IT Asset Class)

- ❑ Investments made to engender innovation.
- ❑ Strategic systems are associated with revenues from modified products but not lower costs.
 - Strategic systems are like R&D investments.

In summary, this study suggests that firms make different kinds of IT investments that have different payoffs.



CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Image Credits

Image/Source	Copyright	Location
Characteristics of Different Class of IT Assets	Aral, S., & Weill, P. (2007). IT assets, organizational capabilities, and firm performance: How resource allocations and organizational differences explain performance variation. <i>Organization Science</i> , 18(5), 763-780.	Slides 3-9



Information Systems Management: Key Principles and Practices

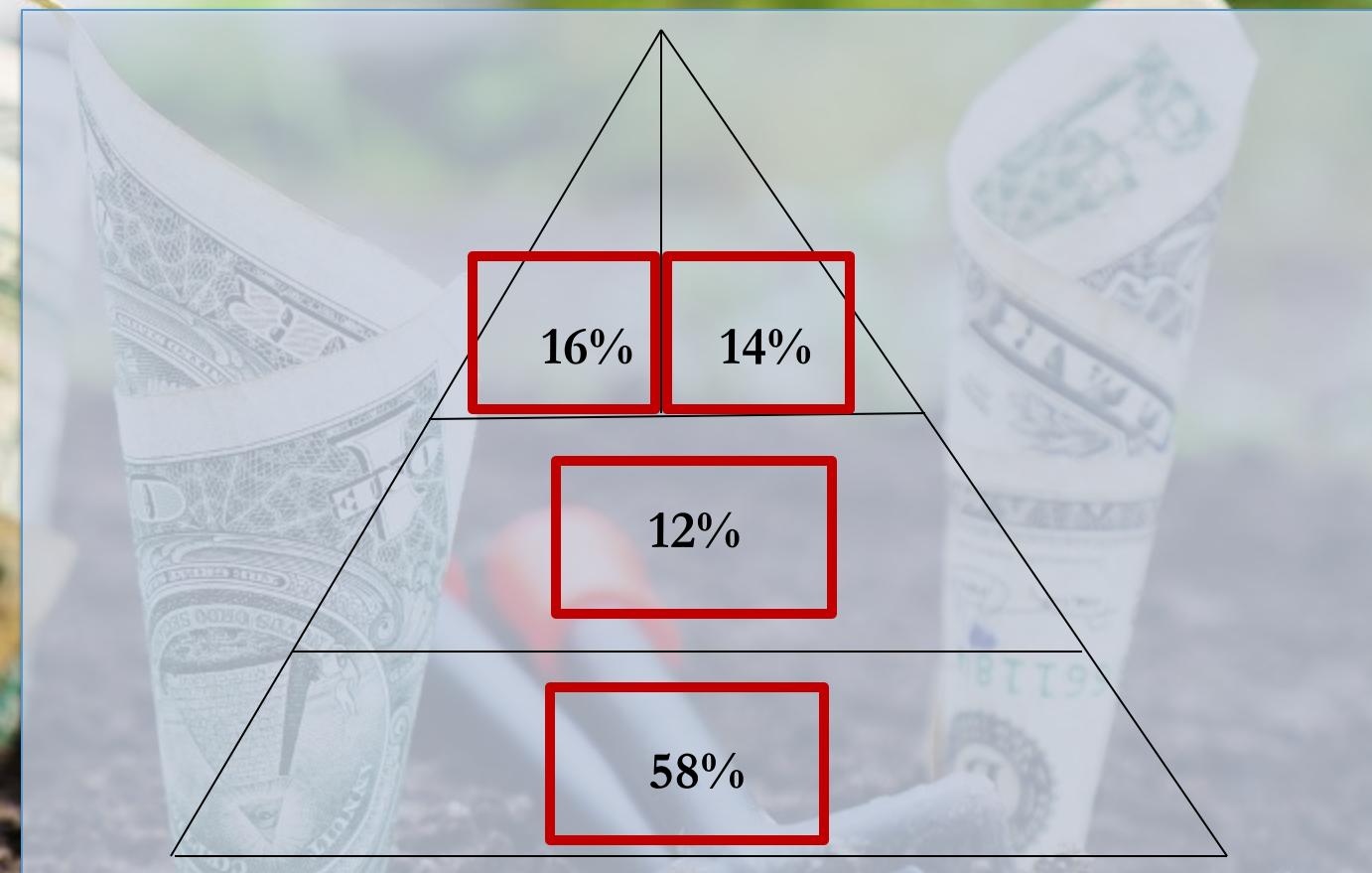
Course 1: IS/IT Governance
Module 3.1: Evaluating IT
Investments (continued)

Lesson Overview and Goals

1. Definition of IT investment portfolio
2. Characteristics of Different IT Asset Classes
3. Strategic Orientation and Portfolio Management

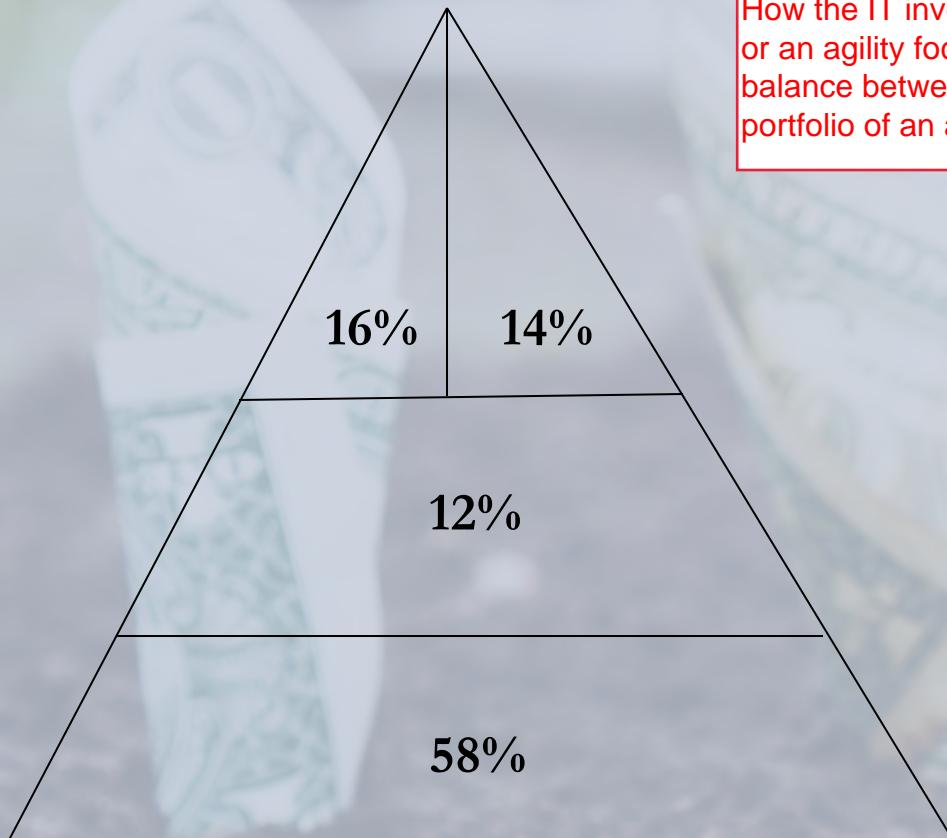
IT Investment Portfolio of an Average Firm

While and Broadband studied a large number of firms across different industries and came up with this characterization of what is an average IT investment portfolio look like. Their study found that on average, firms spend about 58% of the IT dollars on infrastructure, 12% on transactional systems, 16% on information systems, and 14% on strategic systems.

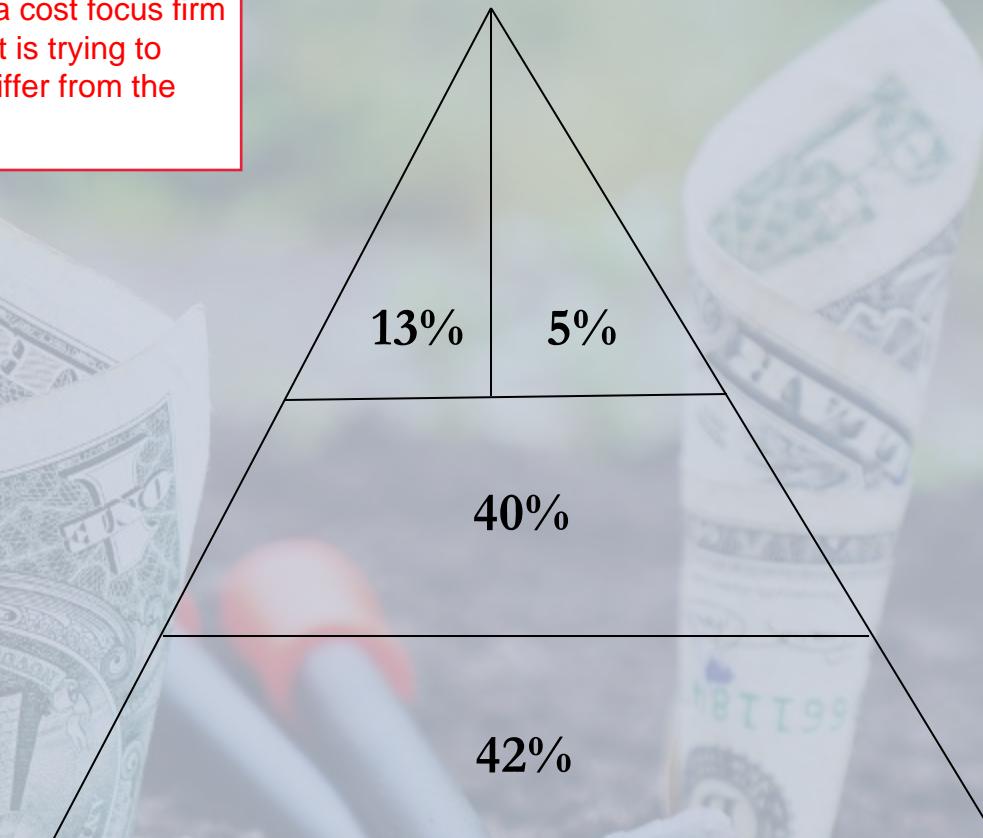


Average Firm

Cost Focused Firm



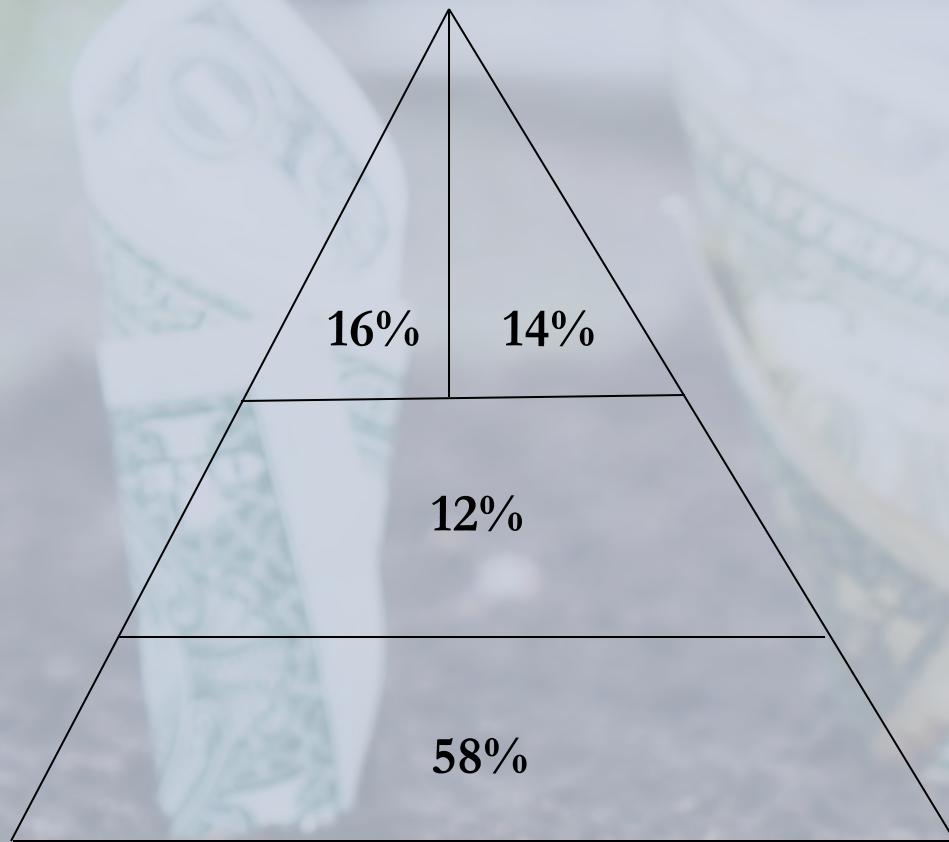
How the IT investment portfolio of a cost focus firm or an agility focus firm or a firm that is trying to balance between cost and agility differ from the portfolio of an average firm?



How does the portfolio of a cost focused firm differ?

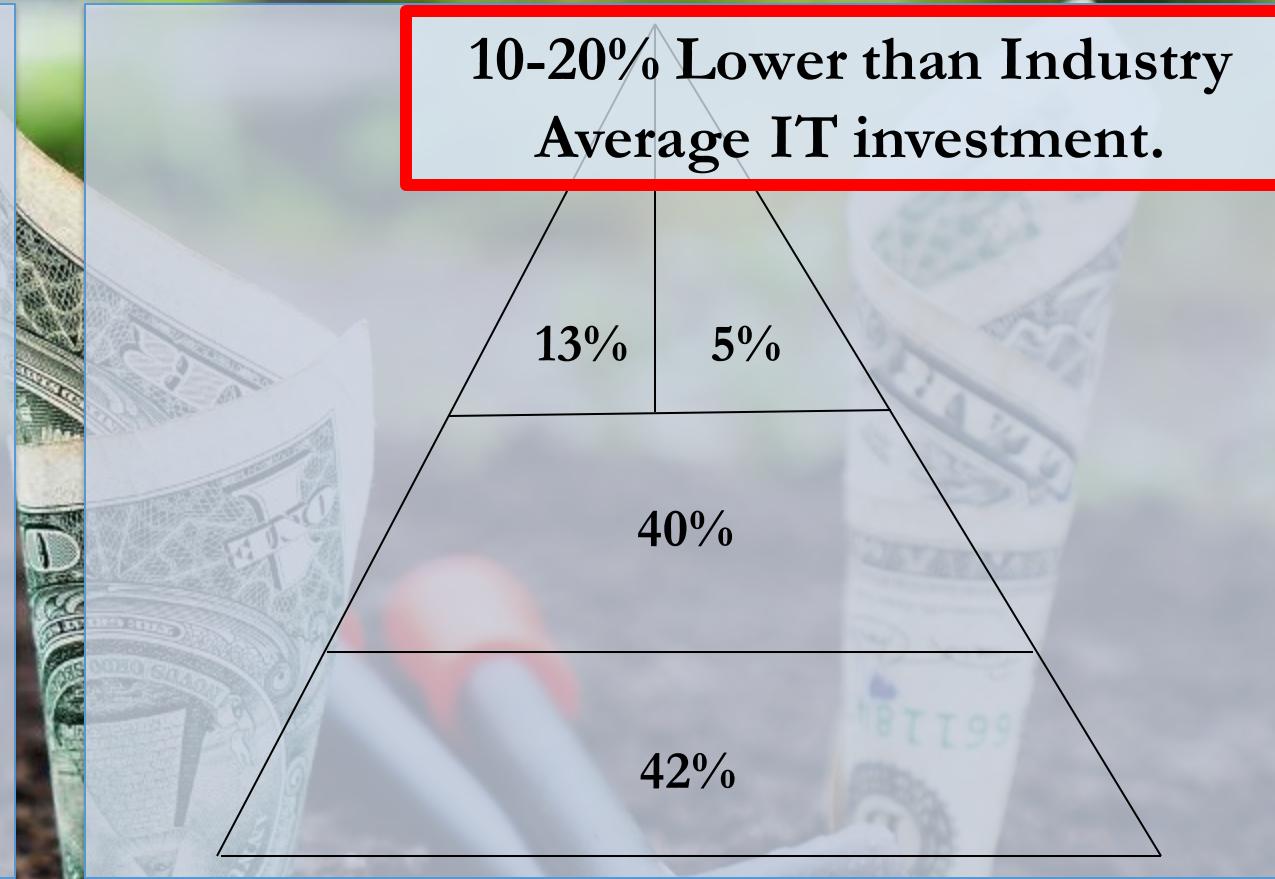
- 1- On average, a cost focused firm has an investment that is 10-20% lower than the industry average.
- 2- A cost focused firm spends more than the average firm on transactional systems, but spends less than the average firm on infrastructure and strategic systems.

Average Firm

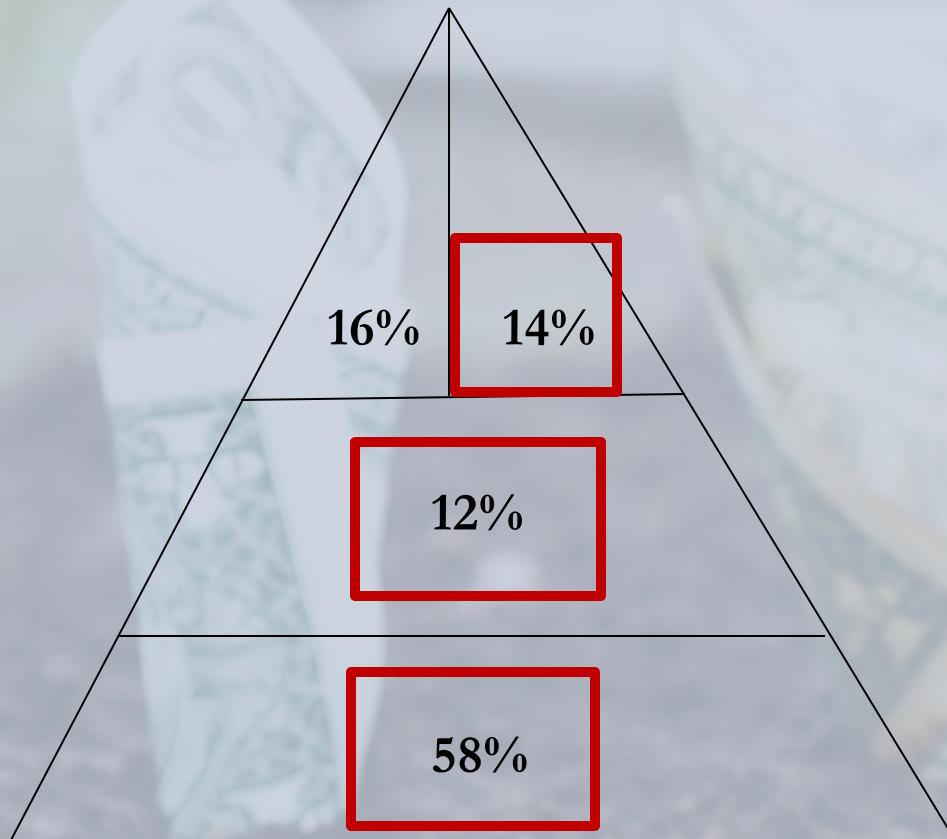


Cost Focused Firm

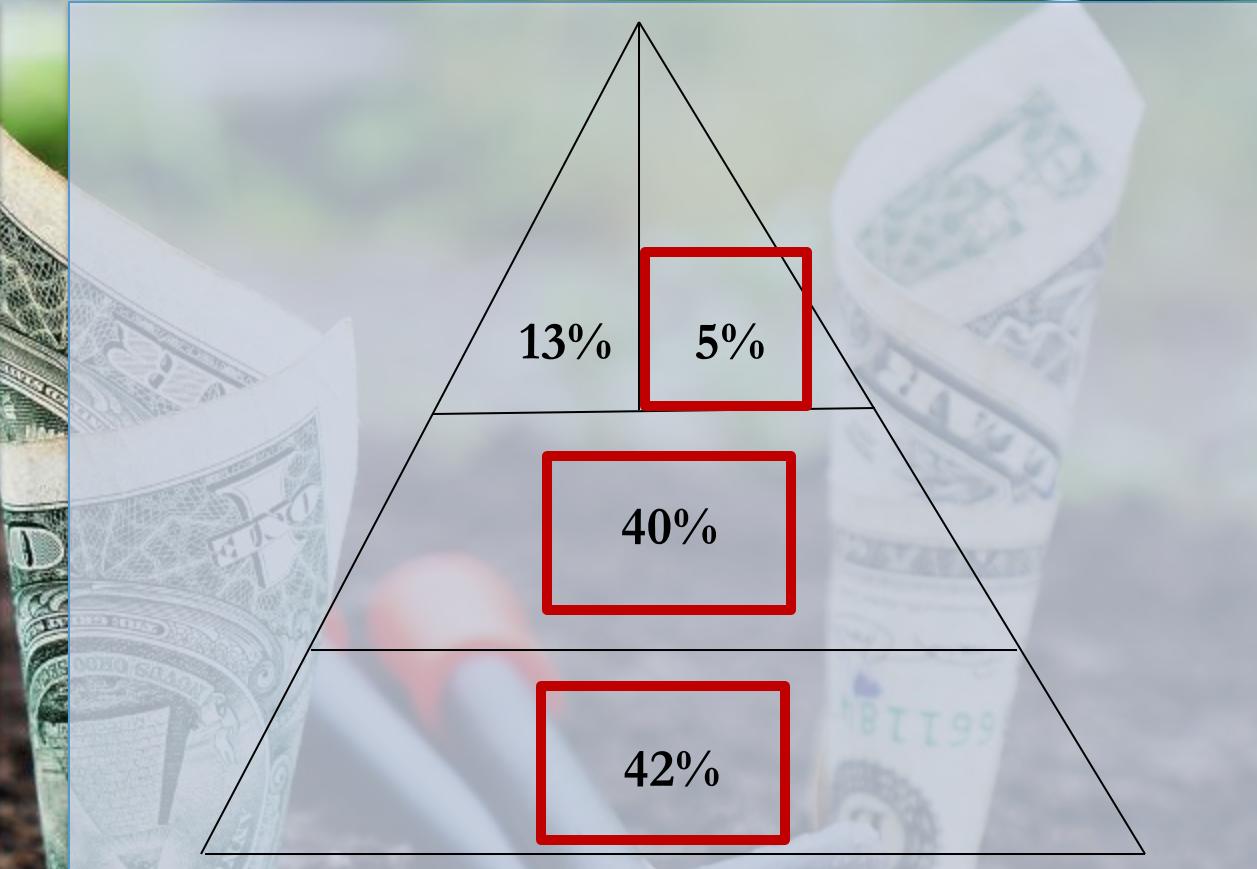
10-20% Lower than Industry
Average IT investment.



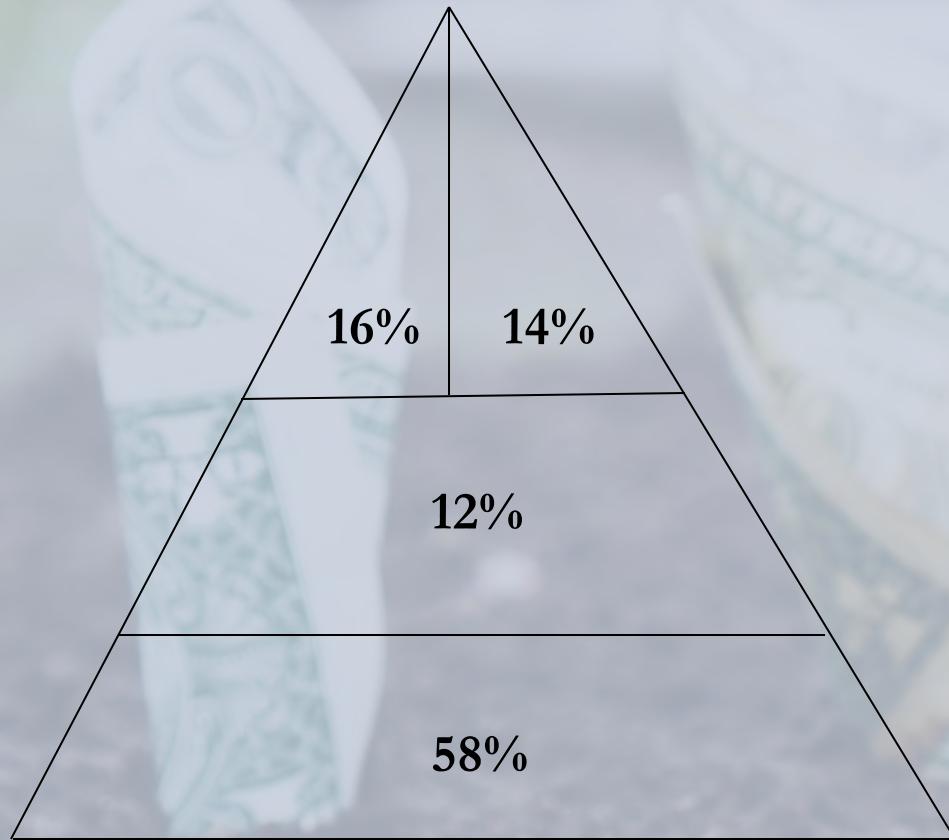
Average Firm



Cost Focused Firm

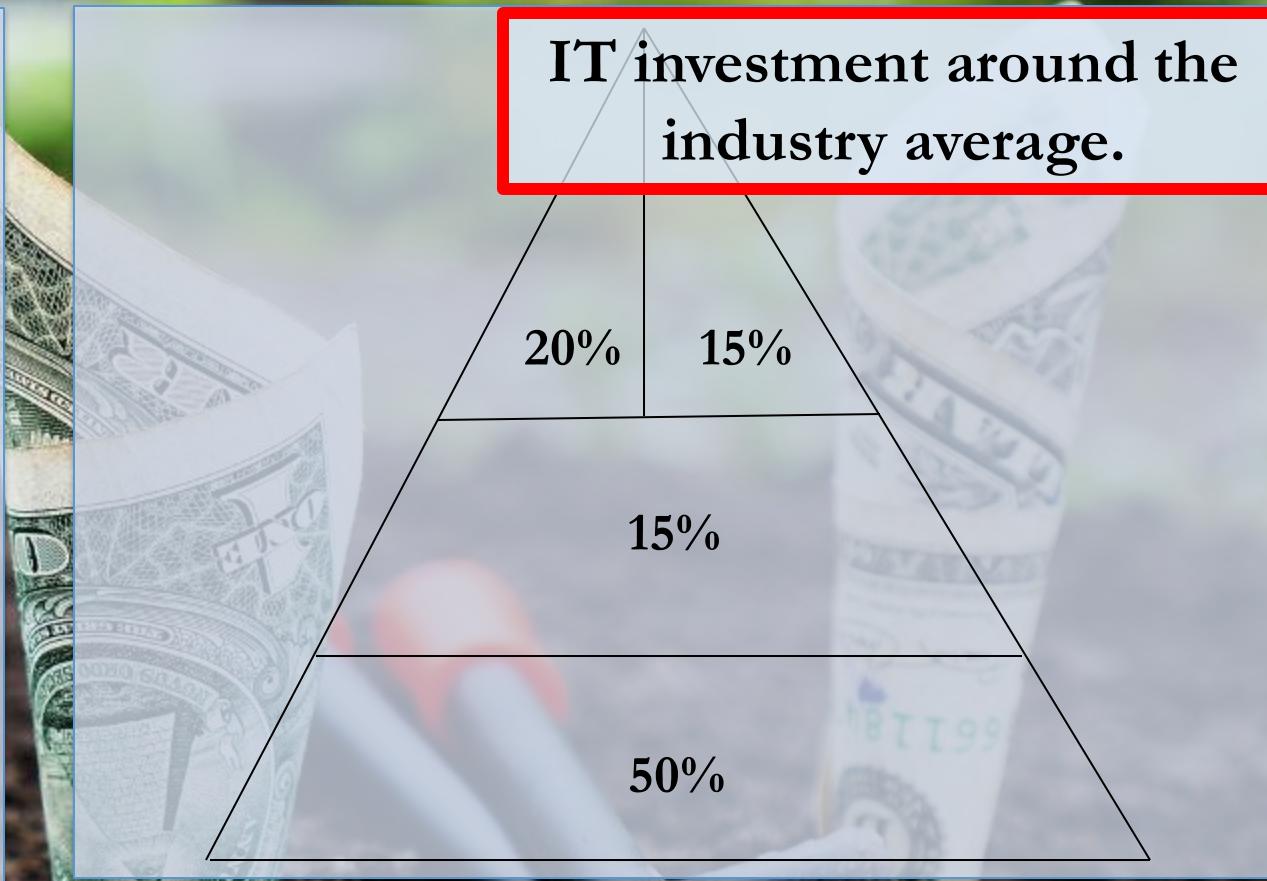


Average Firm

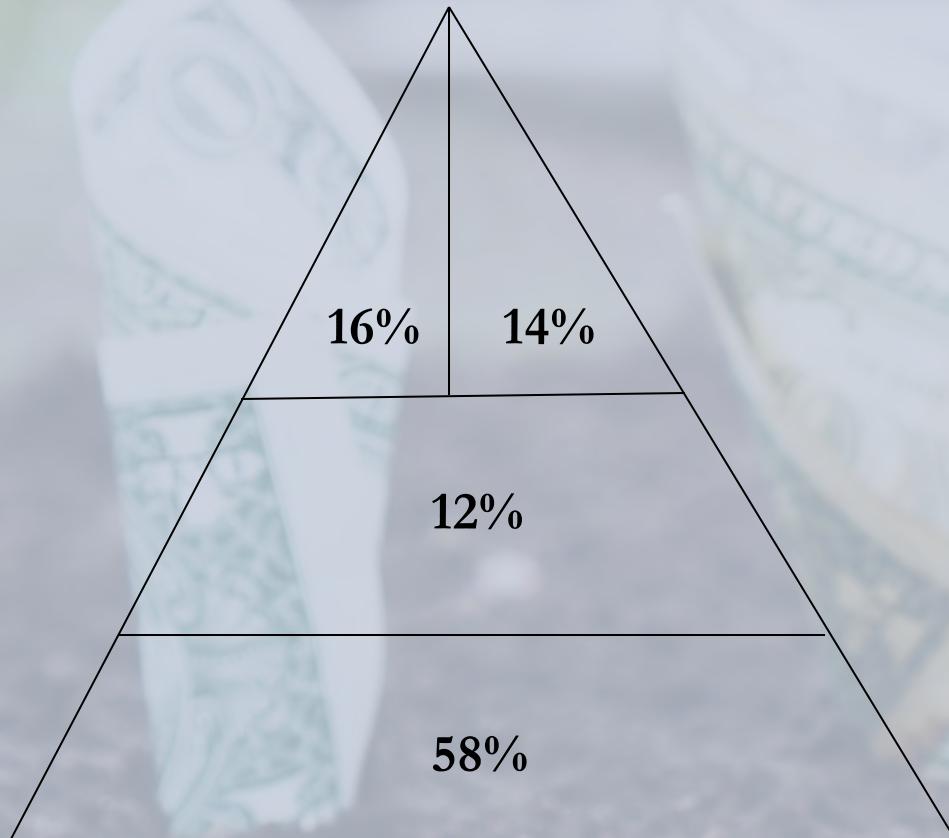


Balance of Cost and Agility Focused Firm

IT investment around the industry average.

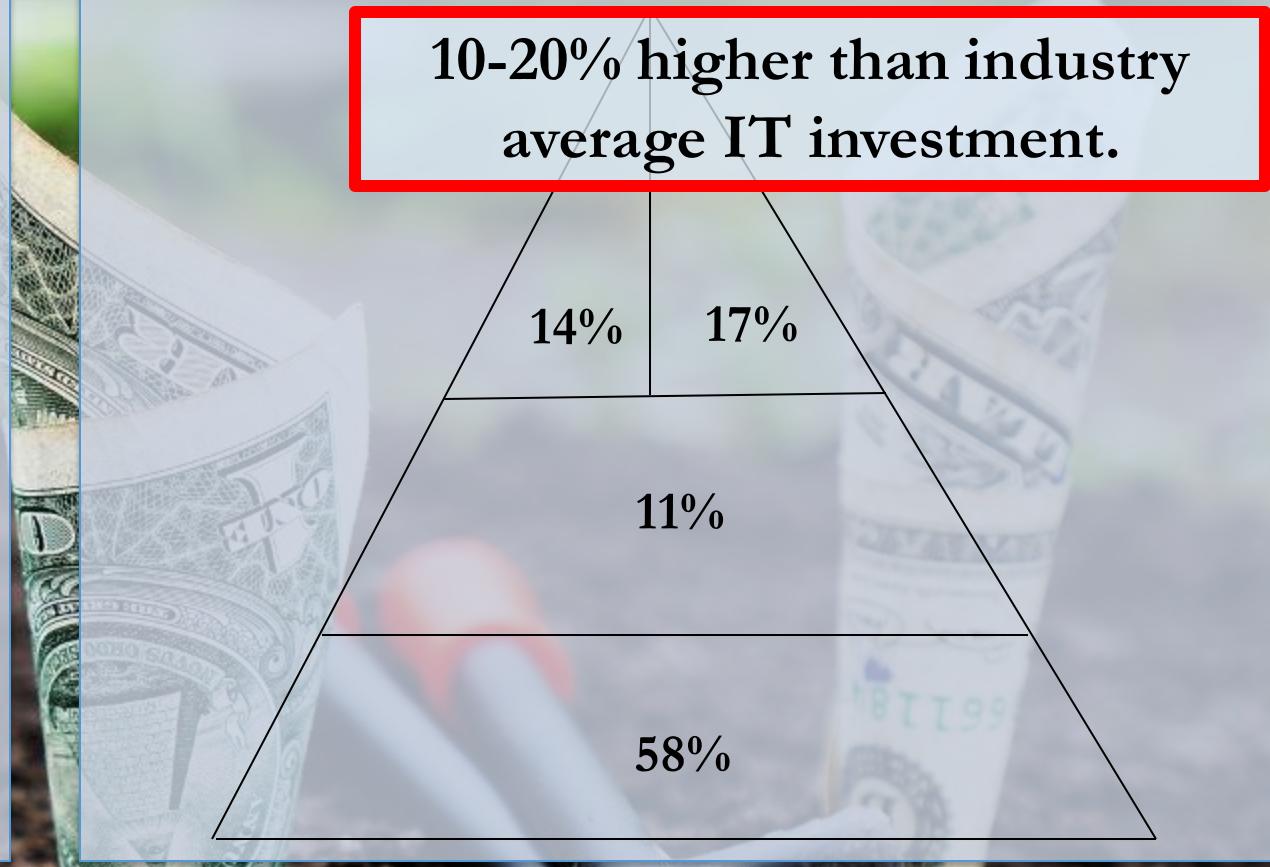


Average Firm

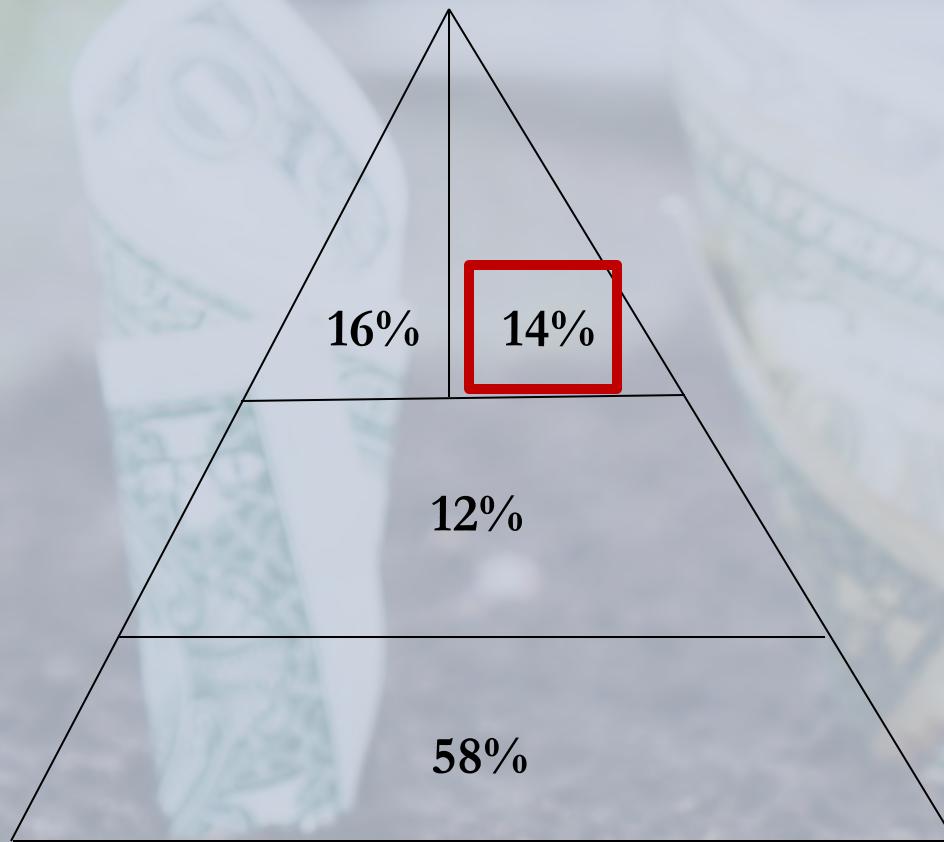


Agility Focused Firm

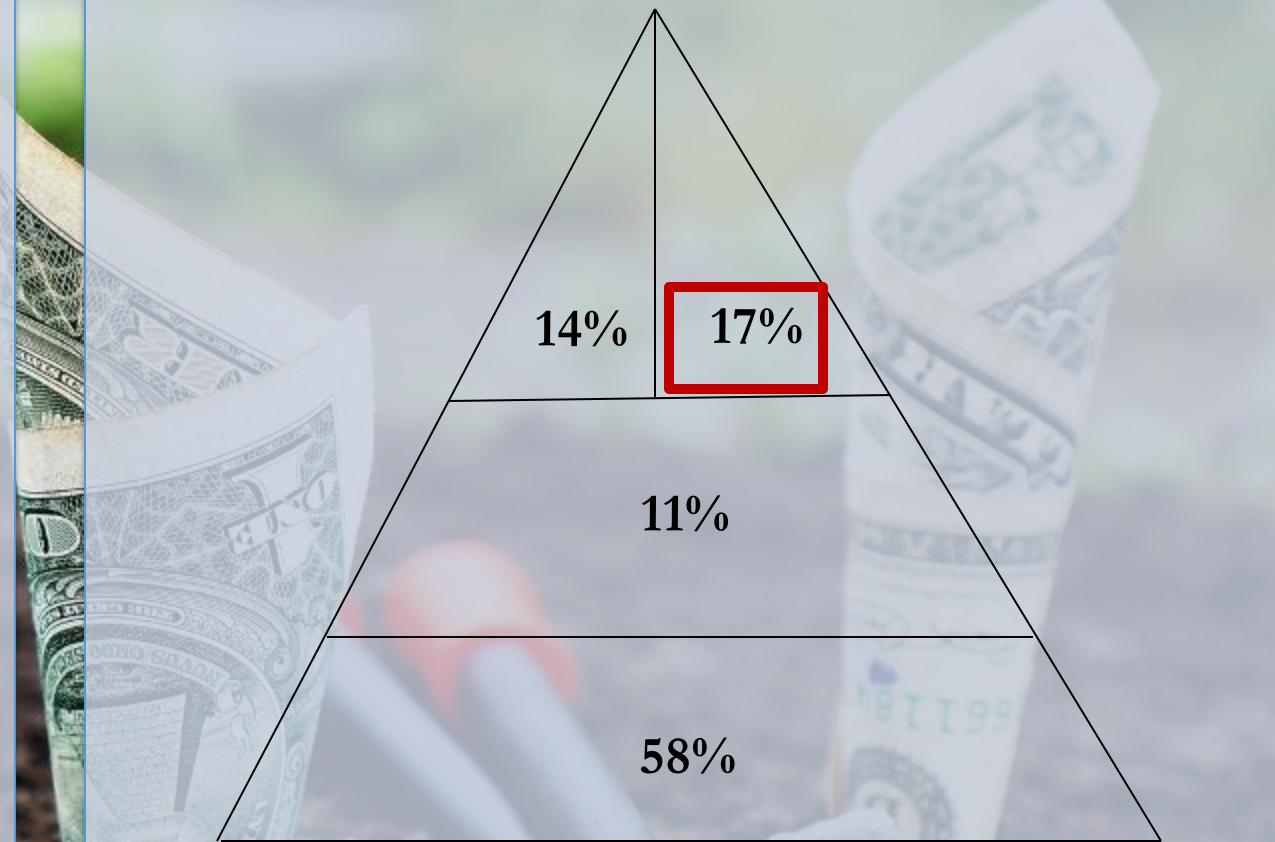
10-20% higher than industry average IT investment.



Average Firm



Agility Focused Firm



Information Technology Portfolio Management Process

What is the process a firm can follow to manage its IT investment portfolio?

1- Inventory of all IT projects. That is, identify all significant IT investments, both current and planned.

2- Ranking and selection of projects. Rank them according to alignment with corporate strategy, technical architecture, evaluation of technology risk, and regulatory compliance.

3- Managing the Project Portfolio

I. Inventory of All IT Projects

- All significant IT investments, both current and planned.
- Project description
- What strategic goal or high-level business objective the project aligns with
- Expected resource requirements
- Project costs, milestones, and timeline
- Financial analysis

Information Technology Portfolio Management Process

II. Ranking and Selection of Projects

- **Alignment with Corporate Strategy:** The alignment of the project with the organization's business goals and objectives.
- **Technical Architecture:** The integration, scalability, and reliability of system components (databases, operating systems, applications, and/or networks) for the project.

Information Technology Portfolio Management Process

- **Technology Risk:** The identification of the proposed projects' exposure to failure or underachievement.
- **Regulatory Compliance:** The determination of the necessity to change because of a change in law, rules, or regulations.

Information Technology Portfolio Management Process

III. Managing the Project Portfolio

- Managing change as business priorities change.
- A continuous reviews of the entire portfolio to ensure that projects remain aligned with the organization.

Information Technology Portfolio Management Process

- This means being willing to restructure, delay, or even terminate projects with performance deficiencies or because of changing business requirements.





CARLSON SCHOOL
OF MANAGEMENT

UNIVERSITY OF MINNESOTA

Image Credits

Image/Source	Copyright	Location
IT Investment Portfolio	Weill, P., & Broadbent, M. (1998). <i>Leveraging the new infrastructure: how market leaders capitalize on information technology</i> . Harvard Business Press.	Slides 3-9
Information technology portfolio management	Lane, D. (2011). <i>The Chief Information Officer's Body of Knowledge: People, Process, and Technology</i> (Vol. 571). John Wiley & Sons.	Slides 10-14

Organizational Readiness Overview

If you are a people manager, you are responsible for making sure your organization is able to accomplish its goals today, and in the future. You've already invested time and effort into hiring the best people; setting goals that will achieve the organization's results; giving your staff the feedback, coaching and development they need to achieve their highest potential; and evaluating your team's year end results.

But, what about the future? What will you do if something changes in your workforce? It might be a staff retirement or someone taking an opportunity in another role. It might be an external change that shifts the focus of your operations. Or maybe the strategic direction of your location shifts due to new opportunities? The nature of such change is frequently out of your control. However, you can anticipate what might change and prepare for those possible future outcomes. And you can drive change in the direction you want by developing a strategic plan to achieve the desired outcomes of the future. This focused strategic planning and follow through is what we call Organizational Readiness.

If you're about to click off this page, consider a little data.

Based on past history, when a UC staff member reaches 19.65 years of service and is 60+ years of age, he or she is "likely to retire." Current UC workforce data indicate that within the next 5-10 years, about 36% of our workforce will be "likely to retire."

- ✓ Do you have staff who might fall into this category?
- ✓ What would you do if one of your senior staff members announced their plans to retire?
- ✓ Have you developed others to be ready to step into the soon-to-be-open role?
- ✓ Are you counting on hiring an external candidate to bring in a different perspective?
- ✓ How will your other staff members respond to being passed over?
- ✓ Will they choose to become the more attractive external hire somewhere else?

These are the questions that keep people managers up at night.

Whether you are responding to a positive organizational goal like UC Merced's goal to grow to 10,000 students by the year 2020, or trying to head off a negative future situation, it makes sense to invest time and energy into preparing your organization to be ready.

Think of it as "business continuity" planning for your workforce needs. You probably have a business continuity program for your IT systems to continue doing business after a disaster. Another way to think of Organizational Readiness is as "business continuity" for your human system. What will you do if a key role becomes vacant? How will you mitigate the negative impact of such a disruption?

We've come up with a 7 Step process to help you do this that will lead you all the way through the process. Before you get overwhelmed by the 7 Steps, though, consider that you may have already done some of these, or may be able to focus on one or two steps to solve the most urgent need at hand, without starting at the beginning of the full process. And, this model can be used at the highest levels,

locally or systemwide, and also at an individual leader's level. We recommend it always be used with the assistance of your local Human Resources expertise.

We are including a similar process for individuals to use to manage their own career readiness, so they'll be ready to take advantage of career opportunities as they come up. Working together with your staff, you can direct the professional development investments you make in your staff to also fill your future organizational needs.

Here are the 7 Steps for Organizational Readiness:

Step 1: Vision of the Future

As with any strategic planning process, it starts with a vision of the desired outcome.

- What do you want the organization to be able to do in 3 to 5 years?
- What significant changes do you anticipate?

Your organization may already have a vision and strategies for implementing it. Most strategies don't get into the details of what staff members are accomplishing which elements of the strategy. Now is the time for you to do a staffing or workforce strategy to accomplish your organizational strategy.

- Who will deliver on the strategic goals?

Step 2: Structure and Key Roles

To bring that vision and strategy alive, the organization structure must be designed to deliver that vision.

- How might work be done differently?
- By different functions or staff?

Identify what roles are critical to the success of the organization, the ones you don't want vacant for long periods of time.

- What roles are critical to accomplish the vision?

Step 3: Competencies for Key Roles

Within the structure, what competencies will need to be demonstrated in those critical roles?

- Today?
- For the future?

Step 4: Assessment of Current Bench Strength – Talent Review

Take a look at your current inventory of talent.

- What is your current bench strength in these critical roles, and do they have the needed competencies?
- Are your individual staff members performing their current roles well?
- Do the individuals have potential for growth?

Step 5: Gap Analysis between Current Skills and Future Needs

Analyze the gap between your current staff and what you'll need for the future.

Step 6: Gap Resolution – Develop, Retain, Acquire

Articulating the gap will inform you what to do next.

- Develop your high performing, high potential staff for future roles and responsibilities, and create a pipeline that makes filling those positions with internal candidates easier.
- Manage your low performers.
- Find out what will retain your steady performers and put those things into play.
- Connect recruiting efforts with internal source pools of candidates ready for their next opportunity.

Step 7: Sustainability and Successes

After select milestones, measure your efforts and determine if you are hitting your objectives, what's giving the best results, and where adjustments need to be made to make the process ongoing. This is a complex process with many components, appropriate since you're addressing the staffing needs of the organization in the moment and in the future.