

# Common Sense Approaches for IT Excellence<sup>SM</sup>

Cisco Global Supply Chain Management  
November 2008

# Proposed agenda

---

## Yesterday

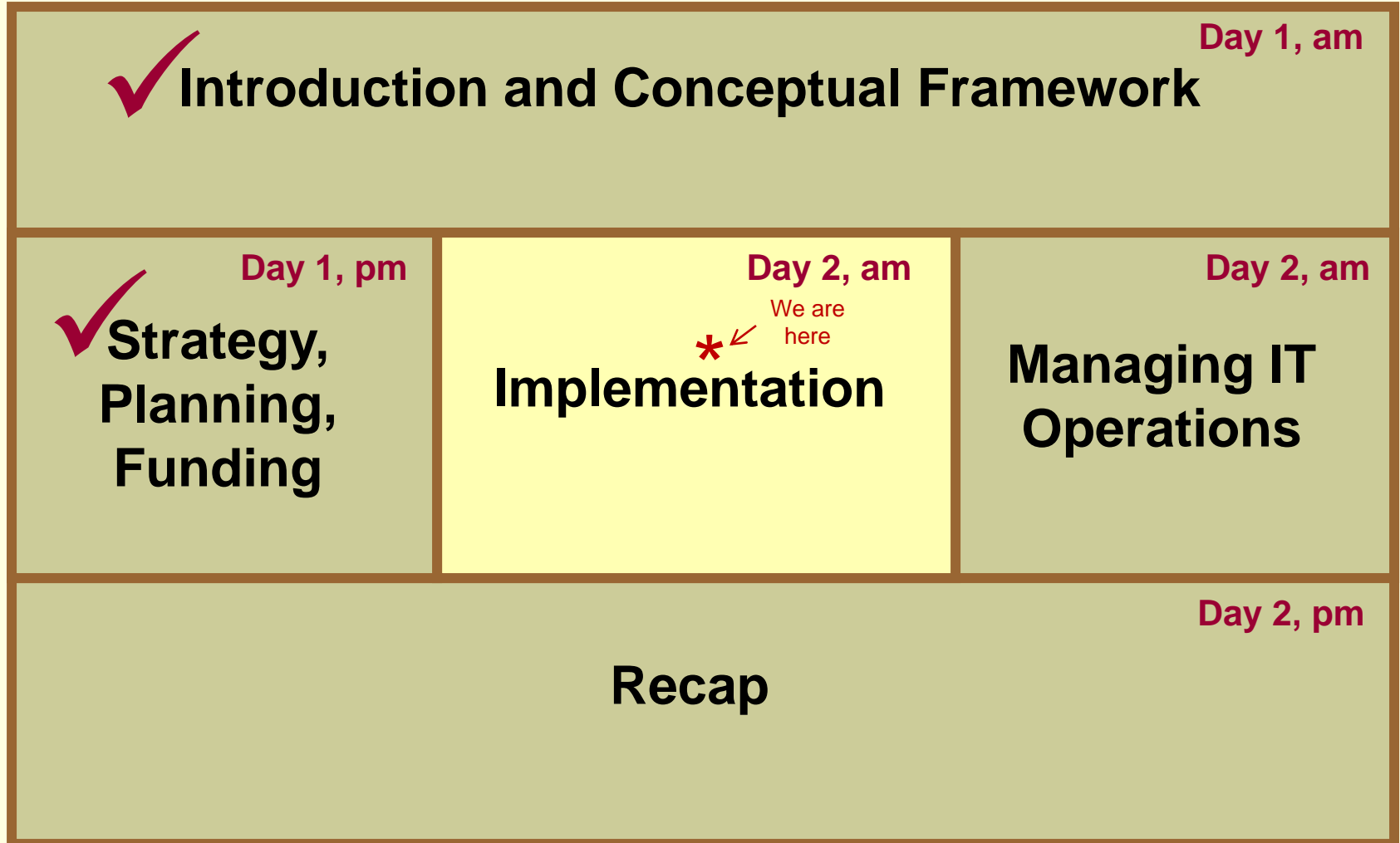
- ✓ Background on this class
- ✓ Introductions
- ✓ Your input
- ✓ IT failures and statistics
- ✓ Effectiveness vs efficiency
- ✓ End-to-end IT management overview
- ✓ Ensuring best use of IT spending
- ✓ Managing projects effectively

## Today

- Building quality solutions: from requirements to solution identification to verification
- Preparing for effective support and maintenance
- Good practices in post-deployment operations
- Summing it up: where to go from here

Throughout the agenda: roles of IT and the business, and how the two can improve communication, collaboration, and results

# Session organization



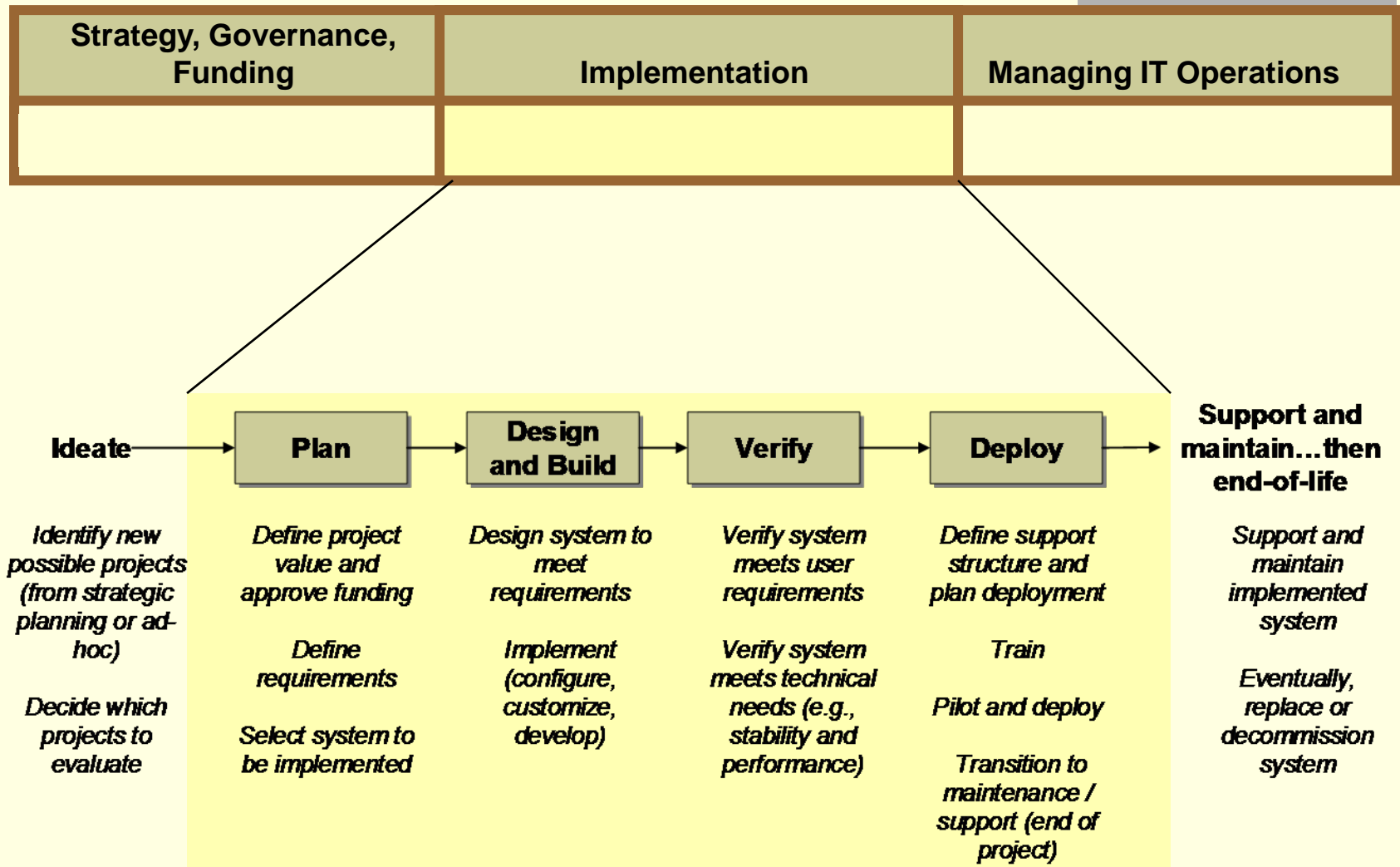
# Business System Implementation

Recap from where we left off yesterday...

# Recap: IT Management Framework

Strategy, Planning, Funding	Implementation	Managing IT Operations
<b>Goal</b> <i>Ensure delivery to critical business needs and best use of IT resources</i>	<i>Ensure effective and efficient project execution and that projects meet intended goals</i>	<i>Ensure system resiliency (short- and long-term) and respond quickly to issues</i>
<b>Key Questions</b> <ul style="list-style-type: none"><li>• What are we trying to accomplish?</li><li>• What do we need to do to accomplish those things?</li><li>• What other attractive opportunities exist?</li><li>• What resources (time, money, people, equipment) are required for each?</li><li>• What resources do we have?</li><li>• What are the best uses of our limited resources?</li><li>• How shall we manage our activities?</li></ul>	<ul style="list-style-type: none"><li>• What are the project's goals and target metrics?</li><li>• What are the business requirements?</li><li>• What will our business processes look like?</li><li>• Who and what do we need?</li><li>• How will we manage the project?</li><li>• How will the solution be used? How will we verify the functionality?</li><li>• How will we implement it?</li><li>• Where do we get the data?</li><li>• How will it interact with other systems?</li><li>• How will it be supported?</li></ul>	<ul style="list-style-type: none"><li>• How will we manage change?</li><li>• Are we providing the support that is needed? Are our customers satisfied?</li><li>• Are we still meeting business needs, or have they evolved?</li><li>• Are we meeting SLA targets?</li><li>• What are the key problems we're seeing? What do we need to do to address them?</li><li>• Do we have the capacity to meet longer-term needs? What else is needed?</li><li>• Are we keeping the systems up-to-date?</li></ul>

# Recap: Generic implementation lifecycle



# Recap: Example activities

## Plan

- Define goals & objectives, key requirements, success measures
- Evaluate project value and secure initial funding
- Develop RFP, including scenarios through which vendor evaluations will occur
- Complete the analysis of candidate systems
- Confirm system selection
- Define project plan, key milestones, success measures, team structure, resource requirements (level and timing)
- Secure funding for next phase

## Design and Build

- Define to-be business processes and system interaction
- Design user interfaces and functional flows (may include prototypes, storyboards, screen designs, early prototypes, system-under-construction, etc.)
- Design system interfaces (may include data flow definition, integration into cross-system storyboard flows, etc.)
- Configure system to meet requirements (may be completed as iteration on above items)
- If applicable, design plan to convert legacy data into new system and implement
- Implement system interfaces
- Test components of the system as configured (unit testing)
- Evaluate project progress-to-goals and current value assessment, define plan for next phase, and secure funding

## Verify

- Test system to functional requirements and designs (system testing)
- Test functionality, usability, completeness with user community (user acceptance and/or usability testing)
- Test end-to-end process flows, including interfaces (end-to-end testing)
- Verify data conversion processes and tools
- Verify performance and stability
- Evaluate project progress-to-goals and current value assessment, define plan for next phase, and secure funding

## Deploy

- Provide training
- Implement and test support structure
- Implement and verify production data conversion
- Transition to production environment
- Go-live
- Provide close monitoring, high level of support, rapid response for immediate post-deployment period
- Transition to normal operations, maintenance, and support

# Business System Implementation

Translating needs to implementation –  
Requirements analysis, usage, and user  
involvement

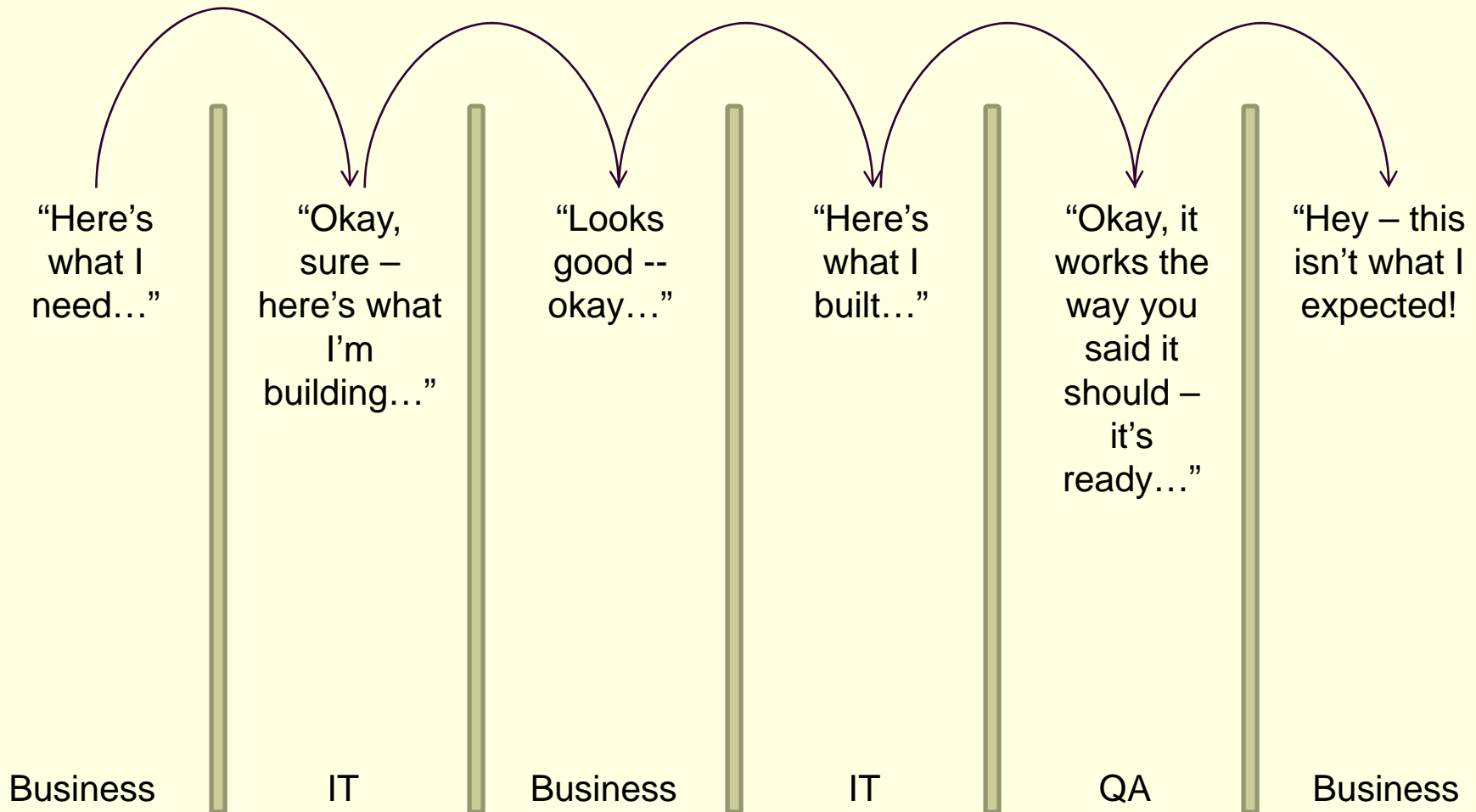


# Problems with the translation

---

- Results at end different from expectations
  - Not what you asked for, or what you thought was documented, or in a way you can use it effectively
  - Has technical limitations you were not aware of
- Does not match performance needs, based on business process, e.g.,:
  - Screen used 40 times per day takes 10 minutes to load
  - Takes twenty clicks to perform a common task
- IT: Get “solutions” from business instead of “needs”
  - No idea of what problem they are trying to solve, nor how they would use a solution to help them in their daily life
- Examples to add?

# “Throw-it-over-the-wall” development



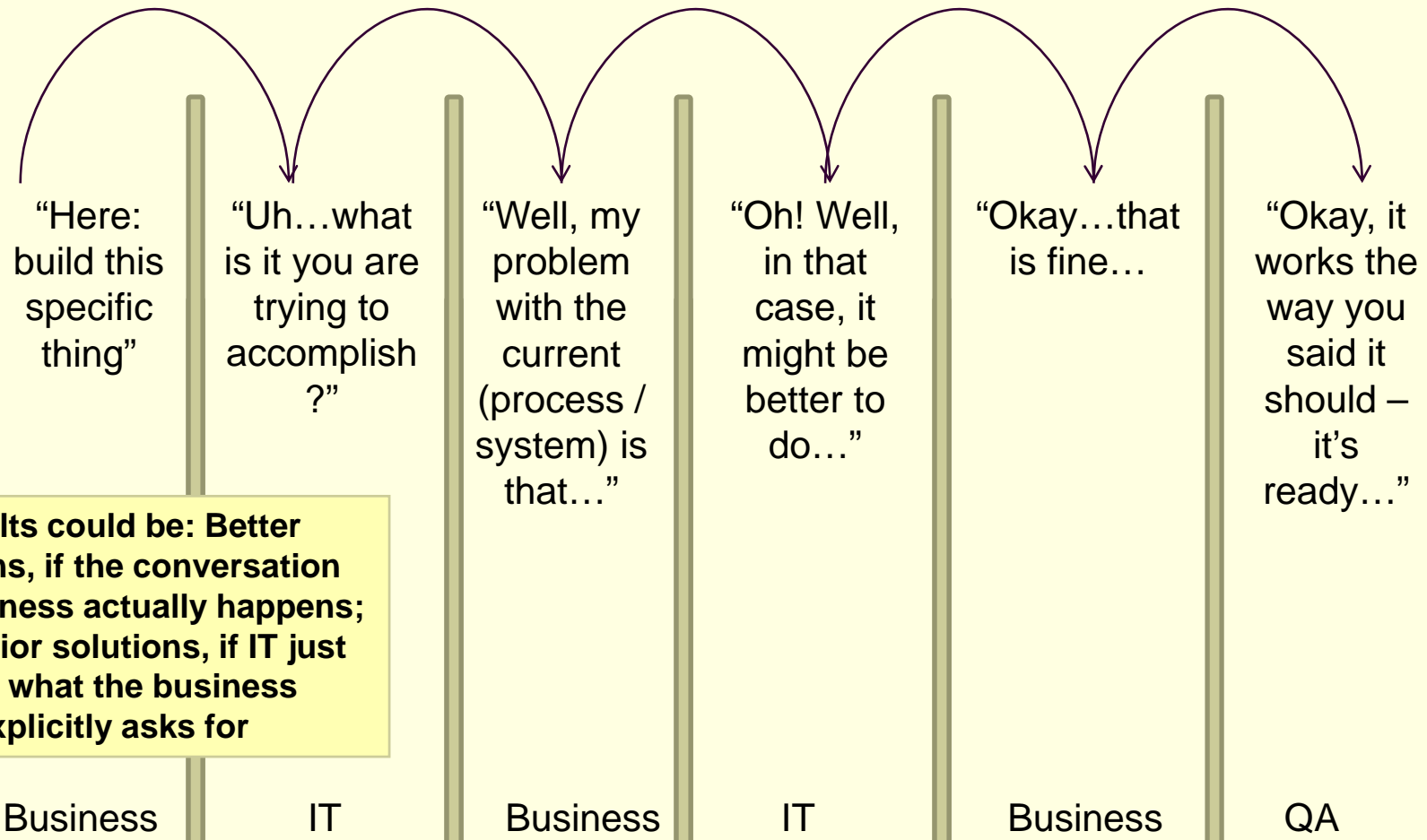
# Documentation involved can be subject to interpretation

---

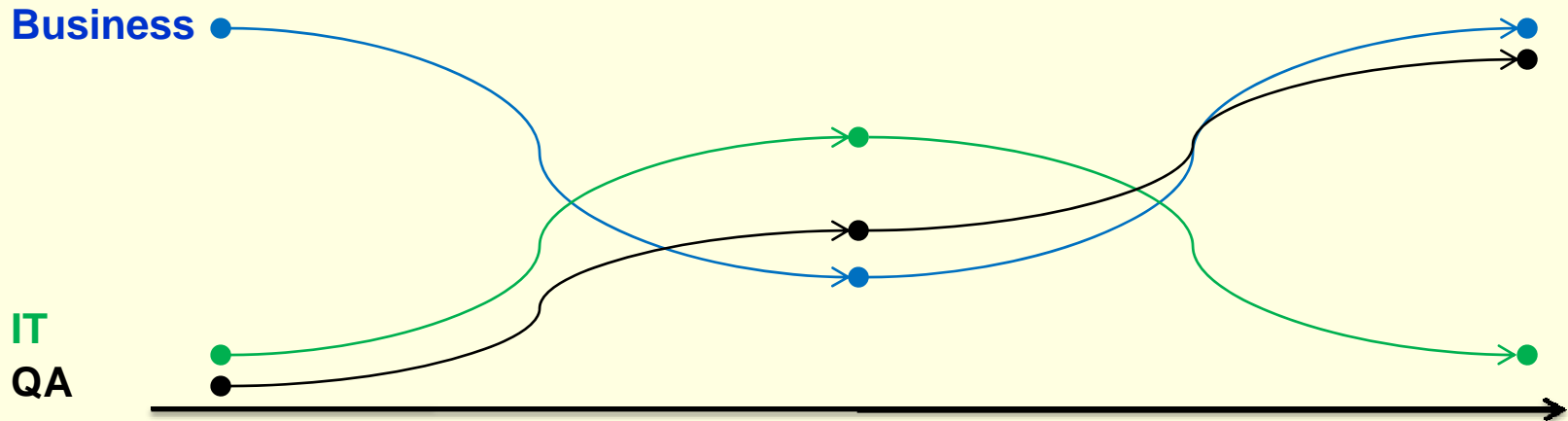
- Business requirements
- Functional specification
- Test plans and test cases
- Document reviewers and approvers...
- ...may be busy...
- ...may not read closely...
- ...may assume that what you mean is what they mean



# Alternative: “Skip right to solutions”



# Traditional time sequencing of roles participation



Requirements

Functional  
specification

(Code)

QA approach

Test cases

Testing

# Problem identification within different approaches

## Example documents

### Traditional Waterfall

High-level Requirements

Detailed Requirements

Design Specifications

(Code)

Test Plans / Approach

Test Cases

Test Results

(Fixes)

(Final Code)

### Iterative, Specification-oriented

High-level Requirements

Detailed Requirements

Design Specifications

Prototypes

(Code)

Test Plans / Approach

Test Cases

Test Results

(Fixes)

(Final Code)

### Iterative, Usage-oriented

High-level Requirements

Scenarios / Use Cases

Storyboards

Design Specifications

Prototypes

(Code)

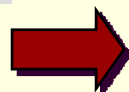
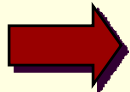
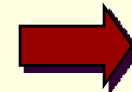
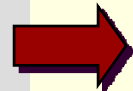
Test Plans / Approach

Usage-focused Test Cases

Test Results

(Fixes)

(Final Code)



*Red arrows show where one might find out that current approach is not usable or implements functionality differently than how the business process flows*

# Pros and cons

	Traditional Waterfall	Iterative, Specification-oriented	Iterative, Usage-oriented
<b>Pro</b>	<ul style="list-style-type: none"> <li>• Everyone knows how</li> <li>• Lifecycles and in-place processes are commonly geared best for waterfall approaches</li> </ul>	<ul style="list-style-type: none"> <li>• Find problems sooner – as soon as functional “chunks” get tested</li> <li>• Less cost to fix issues</li> <li>• More predictable delivery - focus set, fewer unknowns</li> </ul>	<ul style="list-style-type: none"> <li>• Enables better, earlier collaboration with IT and business</li> <li>• Enables issues to be found and addressed earlier</li> <li>• Enables delivery of more usable functionality</li> </ul>
<b>Con</b>	<ul style="list-style-type: none"> <li>• Tasks at the end may be compressed (e.g., testing)</li> <li>• Do not find out that solution is a poor fit or does not meet expectations until the very end</li> <li>• No time to adjust</li> <li>• High cost to fix issues</li> </ul>	<ul style="list-style-type: none"> <li>• More thought in planning</li> <li>• Maybe – additional deliverables</li> <li>• May not be space to compress time if late</li> <li>• Hard to keep things small – scope creep</li> <li>• Waterfall seems more “native”</li> </ul>	<ul style="list-style-type: none"> <li>• Additional work up front to think about usage, not just requirements</li> <li>• Additional deliverables or content for usage</li> </ul>

Less cost (or higher quality) plus increased effectiveness moving right →

# Usability and performance

---

- Solution may not meet needs well if:
  - Requirements worded as functional requirements only
  - Developer has no sense of what “daily life” of the requesting business user is
- Example:
  - User wants two new pieces of functionality added to an existing system: Function X and Function Y
  - Function X supports a process that is run 50 times per day
  - Function Y supports a process that is run a few times per week, whenever “deviances” of some type occur
  - How might the designed code be different if the developer knows the above information versus does not know?



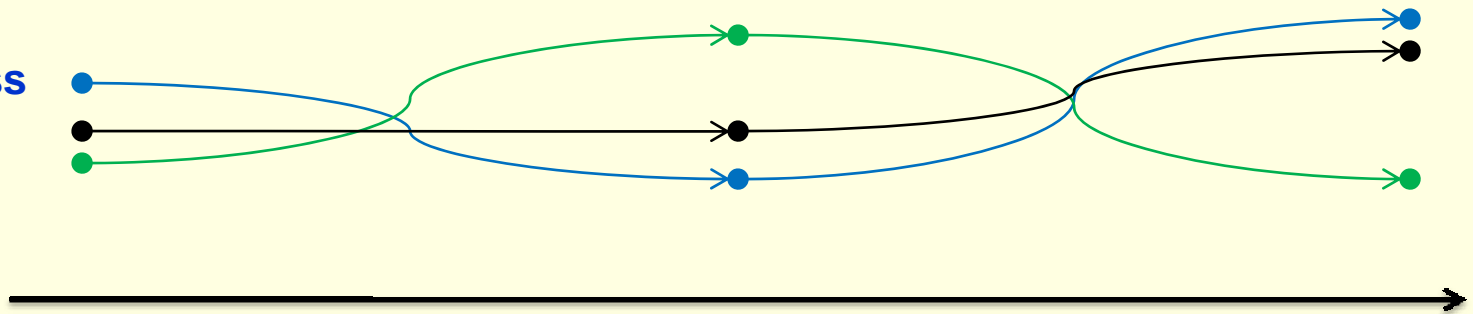
# Usability and performance

---

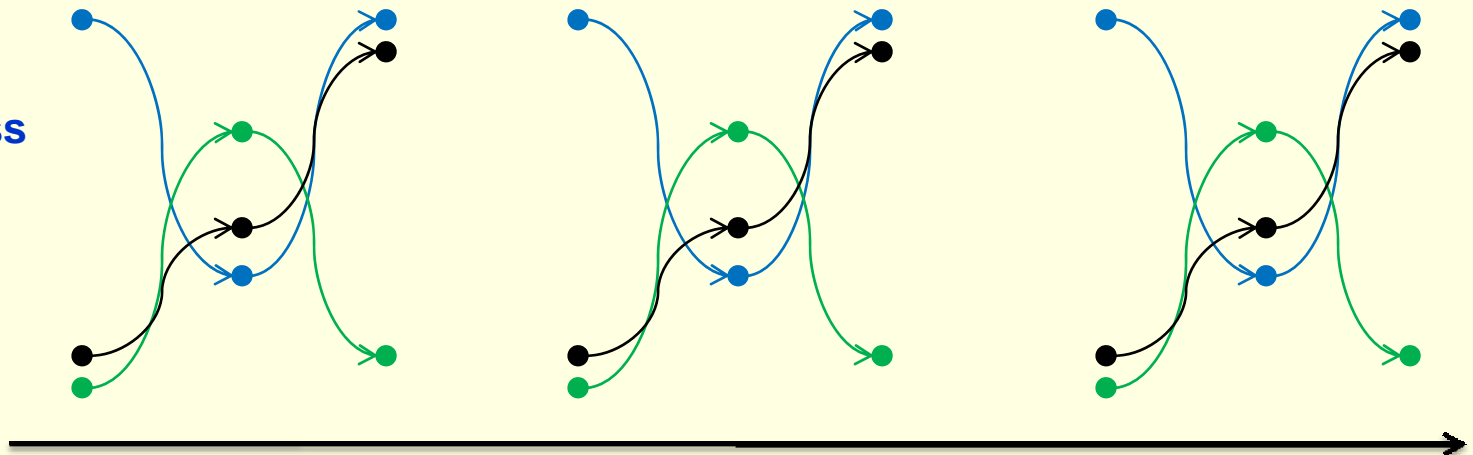
- “Performance” needs to be optimized for X but not Y
- Performance could be technical performance (SQL performance cost, network efficiency, code speed, ...)
- Or, it could also be:
  - Number of steps (clicks, windows, tools, reports) needed to perform the business process step
  - Manual data entry needed even though the information is available elsewhere and could be provided via a lookup
- If these needs are not specified and not understood by the developer, they will likely not be met...
- ...And the user might complain that “IT creates unusable software”

# Iterative/collaborative

Business  
IT  
QA



Business  
IT  
QA



# Requirements Analysis

---

- Requirements analysis is a whole field of study, with a lot of different component parts and theories
- For new systems they guide:
  - Selecting the best-fit system for the needs
  - Configuring and customizing the system to meet needs
  - Verifying that the system meets those needs
- For incremental changes, the last two are relevant
- As they drive how the system will be verified against those needs, they should be written to be *testable*
- Sometimes when needs are not met, it is due to poorly written, un-testable requirements

# Tips before we segue to the next topic (verifying what was built)

---

- “Requirements” are only part of the conversation with business
  - Provide good documentation – but communicate and collaborate early and often
- Build requirements and usage documentation (e.g., scenarios) to drive directly into test cases (write to be “testable”)
- Moving from big-bang waterfall to fully-iterative approach can be a cultural challenge; benefits can be gained by making simple changes
  - Use storyboards and/or prototypes to show users what you will build and how they will use it – and adjust if needed
  - Break planned functionality into meaningful, smaller chunks that are managed independently, with close collaboration
  - May not raise throughput, but likely to improve satisfaction with the outcome and predictability around the plan (time, budget)

# Closing comments

---

- More reliable delivery to needs with increased collaboration:
  - Smaller pieces
  - Checkpoint discussions (IT and business) more often
  - Usage with supporting visuals
  - Iterative: not strictly linear
- Can be a difficult concept to swallow – already pressed for time, cannot add anything else
  - Depends on whether more worried time compliance or best solution for users

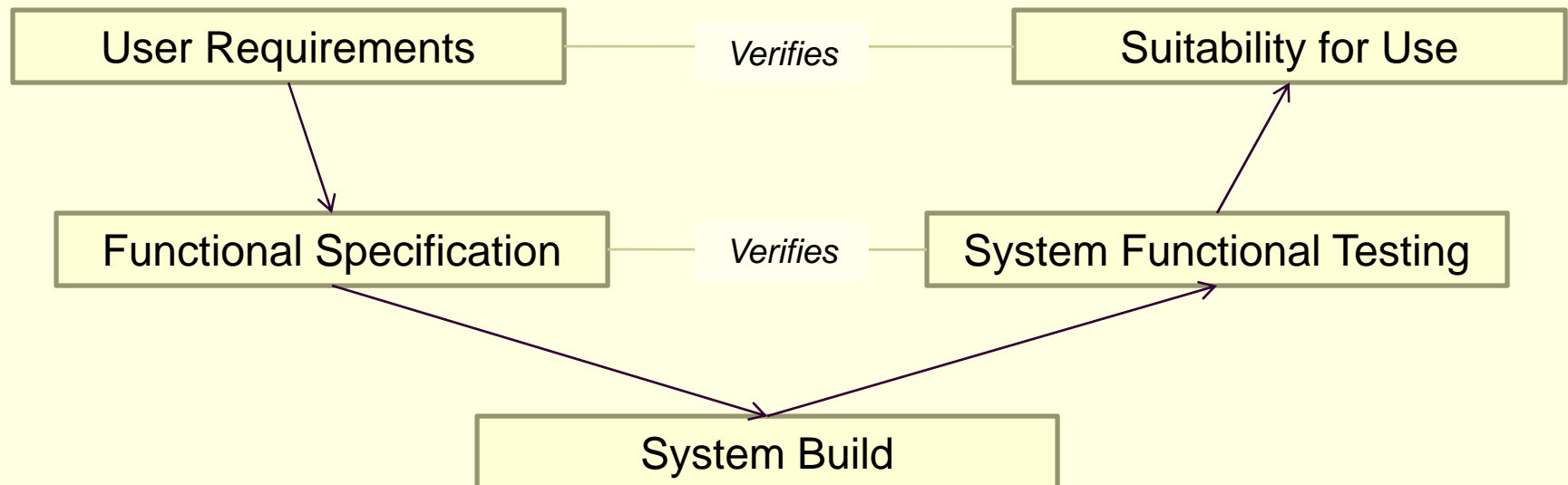
# Business System Implementation

Best practices in Quality Assurance / Testing

# Quality Assurance defined

- Processes for verifying that a system does what it is supposed to do
  - Functional
  - “Fitness for use”

*Simple V model*



# Myriad terminology for test types

Unit testing	“System” testing	UAT	Smoke testing
Integration testing	“Functional” testing	Usability testing	Deployment testing
	“End-to-end” testing	“Flow” testing	
	Interface testing	Performance testing	
	Regression testing	Stress testing	
“White box” testing	“Grey box” testing	“Black box” testing	

Lots of different terms, not always used in consistent ways... Easiest to boil it down to **“what do you need to test?”**



# Requirement types and non-functional requirements (the “ities”)

---

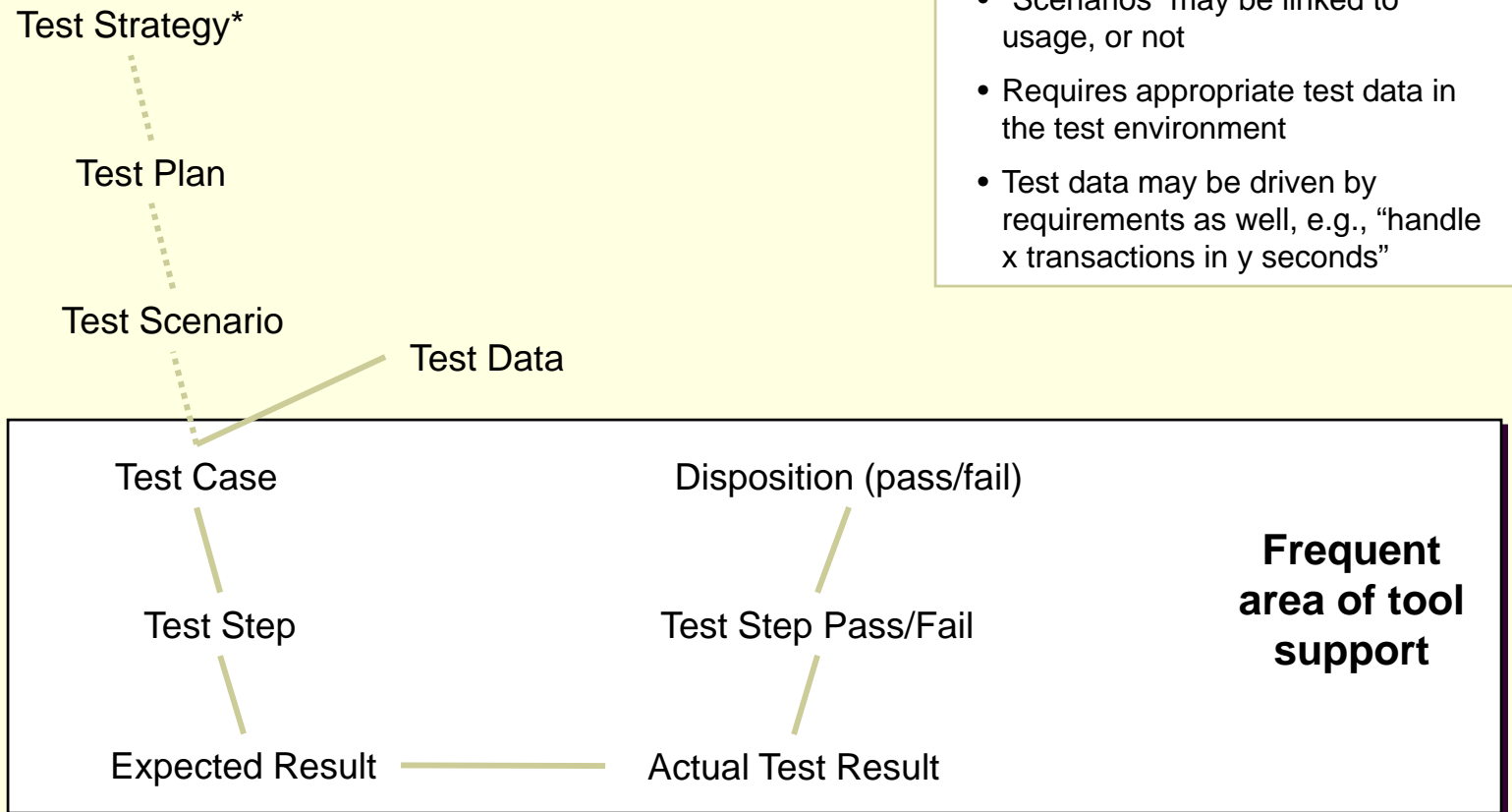
- **Total Requirements Set = Functional Requirements + Non-Functional Requirements:**
  - Testability
  - Scalability
  - Stability / Reliability
  - Maintainability
  - Usability / Suitability
  - Compatibility
  - Performance (to requirements, or under load)
  - Security
- Non-Functional Requirements commonly under-emphasized

# Critical types in summary (although usage of terminology varies)

Type	Description
<b>Unit</b>	Performed by <b>developers</b> to ensure what they have implemented works as expected and is <b>ready for a broader review</b>
<b>System</b> (integration, interface, performance, stability,...)	Brings together all the unit-tested code into a <b>shared environment</b> so that the full <b>range of functionality</b> , integration points, performance, etc. can be tested together. Should include verification of required <b>qualities</b> – not just functional requirements
<b>Regression</b>	Tests <b>unchanged code</b> to ensure that it continues to work. Testing may include “ <b>full regression</b> ” (test everything) or use <b>impact assessment</b> to identify areas of greatest risk of impact from the changes (may be a problem if assessment is not right)
<b>Usage</b> (flow, potentially UAT)	Tests the system <b>end-to-end</b> in <b>the way it will be used</b> . Ideally performed within system testing, and confirmed with User Acceptance Testing
<b>UAT</b>	“User Acceptance Testing” – unspecific term meaning that the users accept the change
<b>Deployment / Smoke Testing</b>	Post-production deployment testing to verify that it “still works” after code migration into production

# Test hierarchy

## Example terminology



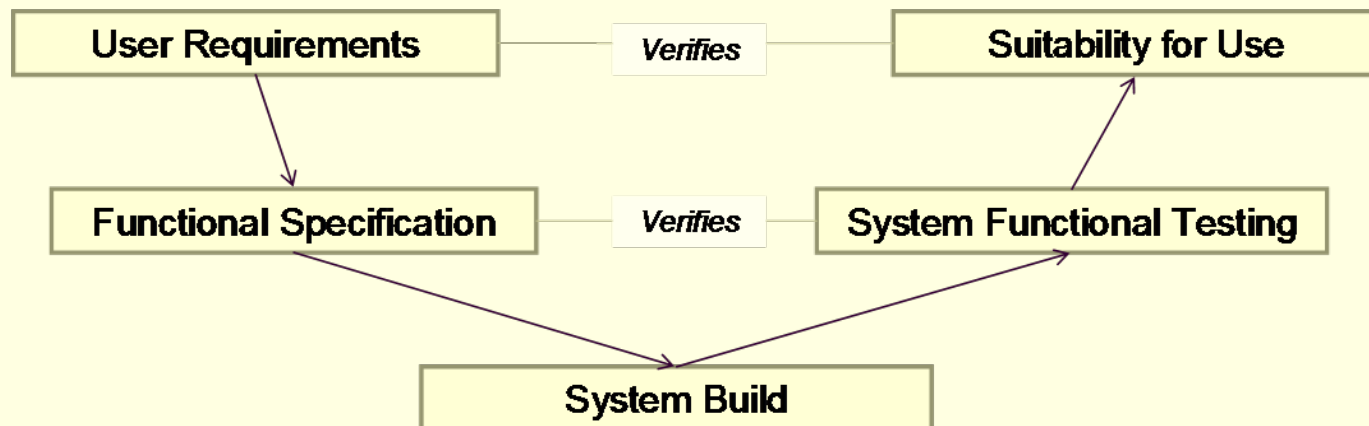
### Notes:

- “Scenarios” may be linked to usage, or not
- Requires appropriate test data in the test environment
- Test data may be driven by requirements as well, e.g., “handle x transactions in y seconds”

*\* Not necessarily a stand-alone deliverable*

# Requirement traceability

- Can be very powerful, but not always done
- Traces testing performed and results to requirements
- Can be used to trace to functional specification and business requirements and/or usage
- More critical in some environments than others



# Closing comments

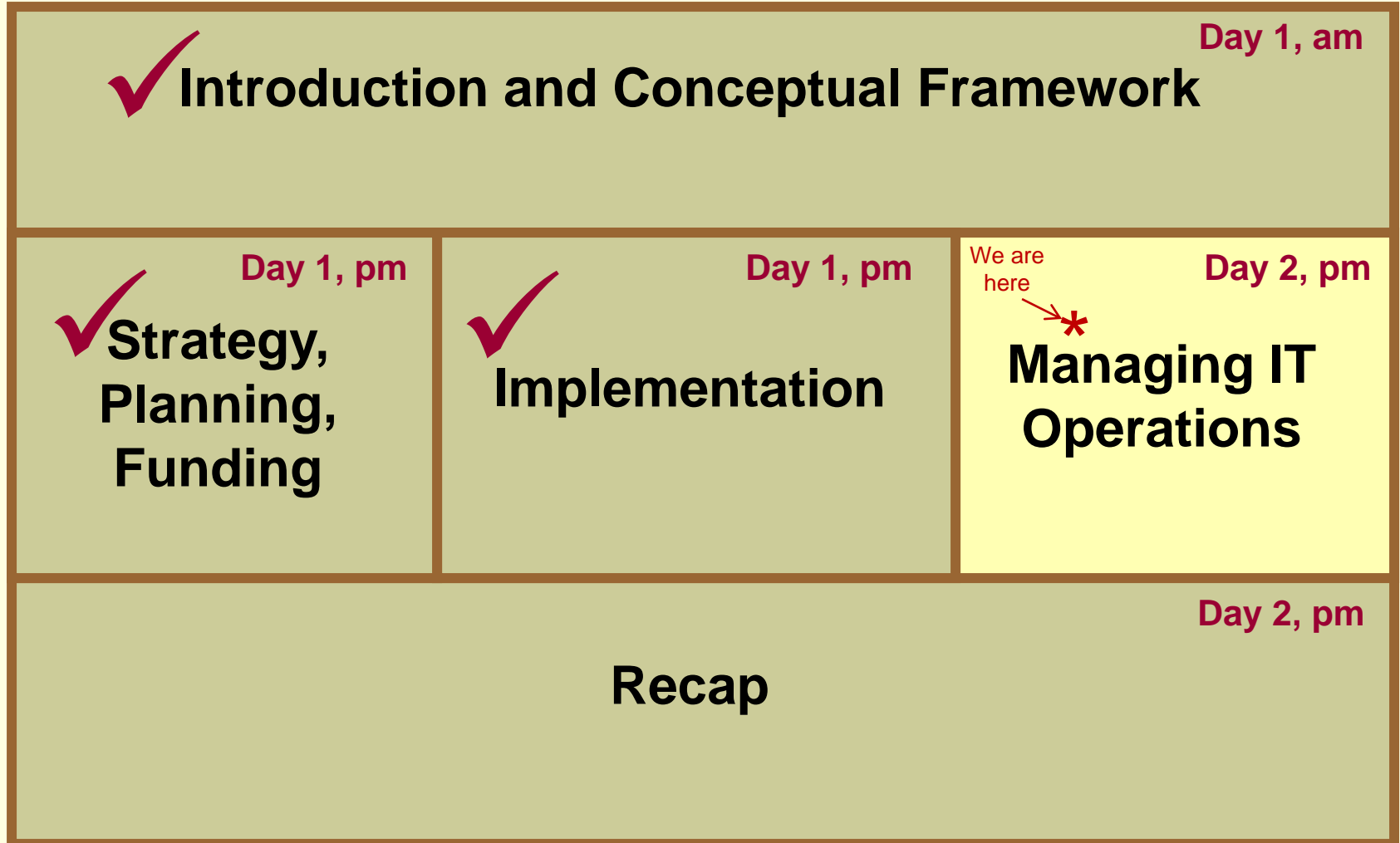
---

- QA may be handled in different ways organizationally
  - Done by IT, done by QA group within IT, done by separate QA organization
  - No one “right” answer; outcomes depend on the quality of the QA practices, not on who does it and where they are
- Good practices:
  - Test the system as it will be used; do not just verify compliance to functional specification text
  - Include “non-functional” required qualities in requirements definition (technical and business driven)
    - Push the business to give you specifics
  - If the requirements and usage discussions are done thoroughly, the test cases should just “fall out” of prior work

# Managing IT Operations

In context

# Session organization



# Recap: IT Management Framework

## Strategy, Planning, Funding

## Implementation

## Managing IT Operations

### Goal

*Ensure delivery to critical business needs and best use of IT resources*

*Ensure effective and efficient project execution and that projects meet intended goals*

*Ensure system resiliency (short- and long-term) and respond quickly to issues*

### Key Questions

- |   |   |   |
|---|---|---|
| <ul style="list-style-type: none"><li>• What are we trying to accomplish?</li><li>• What do we need to do to accomplish those things?</li><li>• What other attractive opportunities exist?</li><li>• What resources (time, money, people, equipment) are required for each?</li><li>• What resources do we have?</li><li>• What are the best uses of our limited resources?</li><li>• How shall we manage our activities?</li></ul> | <ul style="list-style-type: none"><li>• What are the project's goals and target metrics?</li><li>• What are the business requirements?</li><li>• What will our business processes look like?</li><li>• Who and what do we need?</li><li>• How will we manage the project?</li><li>• How will the solution be used? How will we verify the functionality?</li><li>• How will we implement it?</li><li>• Where do we get the data?</li><li>• How will it interact with other systems?</li><li>• How will it be supported?</li></ul> | <ul style="list-style-type: none"><li>• How will we manage change?</li><li>• Are we providing the support that is needed? Are our customers satisfied?</li><li>• Are we still meeting business needs, or have they evolved?</li><li>• Are we meeting SLA targets?</li><li>• What are the key problems we're seeing? What do we need to do to address them?</li><li>• Do we have the capacity to meet longer-term needs? What else is needed?</li><li>• Are we keeping the systems up-to-date?</li></ul> |
|---|---|---|



# Example issues in transition to support

---

- Focus on go-live; poor planning for what happens after
- Adoption issues
  - Inadequate training, inadequate cultivation of expert users
  - Teams disbanded too quickly once the milestones complete
  - People used to doing things the old way; no one following up to identify and address adoption issues after the fact
- Resources for maintenance underestimated, under-funded
  - Systems sometimes created without consideration for support, especially with entrepreneurial, innovative, high-growth culture
- Gaps in definition of support and maintenance for the system
  - Support model, time coverage, response needs
  - Resources and ownership for proactive maintenance (long-term capacity planning, data management, DR planning)
  - Expectations, planning, funding, ownership for system changes

# “IT Operations” defined...

---

IT Operations includes...

The range of activities involved in **managing, maintaining, and supporting systems** in use in a business

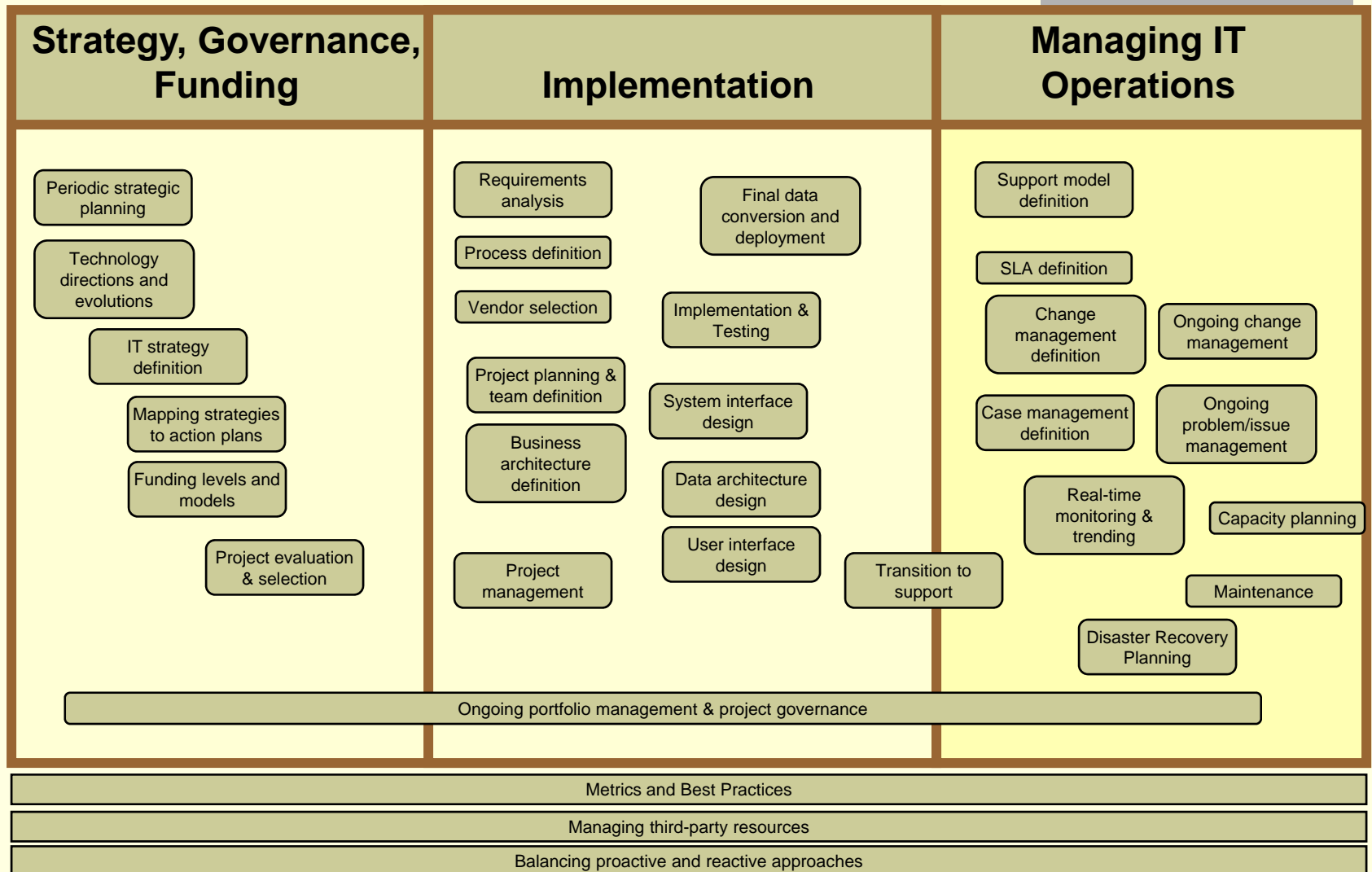
- After a business system is implemented, it transitions into post-deployment support, and becomes part of the set of implemented, managed systems
- Management of all of those systems-in-use is falls under the umbrella of IT Operations
- There are more topics in this area than we are able to cover – we will focus on some key problem areas

# Key operations work

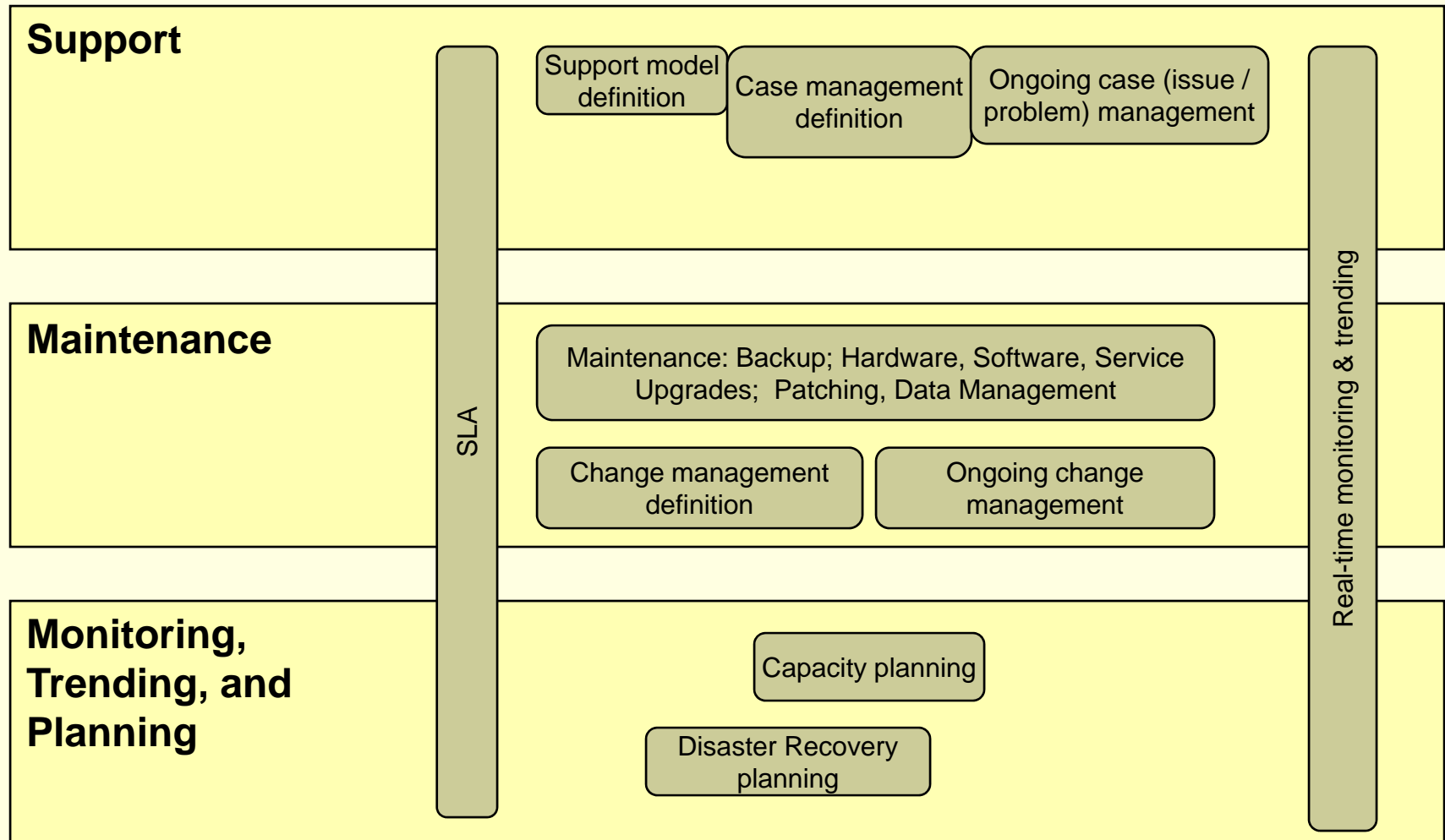
Category	Core Focus
<b>Support</b>	Provide <b>assistance to the business user community</b> , whether in the form of issue capture/resolution, help with system usage, installation of new services, etc.
<b>Maintenance</b>	<b>Keep systems and infrastructure up-to-date</b> which includes backup/restore functions, as well as upgrades, updates, and patches to hardware, software, core services, and infrastructure. May also include management of <b>functional changes</b> to installed systems.
<b>Monitoring, Trending, and Planning</b> (includes <b>Capacity Planning</b> and <b>Disaster Recovery Planning</b> )	<b>Monitor current resource usage</b> (e.g., memory, disk, network), growth rates, capacity constraints, bottlenecks, performance, etc. Identify <b>long-term capacity needs</b> , and define plan to meet them Ensure that key needs for IT enablement can be <b>maintained or recovered as quickly as possible</b> in the event of a natural disaster, terrorism, war, blackout, or other unplanned event
<b>Asset Management</b>	<b>Track and manage hardware and software</b> , potentially including software license management and software charge-backs
<b>Security</b>	Manage security <b>policies</b> , implement and manage <b>tools</b> to maintain security (e.g., firewalls), and <b>monitor security threats</b>

# IT Management Framework:

## Example activities



# Activity alignment by major category...



# Key problem in IT Operations practices

---

Generally speaking, the two most significant thematic issues in IT Operations practices are:

Significant thought given to implementation, but **too little thought given to “what happens after go-live”** (with expectations for post-go live activities not clearly laid out in an SLA)

AND...

...**Over-emphasis on fire-fighting** (responding to issues) **versus proactive planning** (preventing problems)



# Managing IT Operations

Maintenance Practices

# System maintenance activities

---

- Backup / restore (not typically a problem area)
- Patch evaluation, selection, verification, application
- Hardware replacement and upgrade
- Software version maintenance
- Data management



# Proactive maintenance tends to be the larger problem

---

- Keeping software, hardware, OS, patch levels up-to-date to keep systems operational, high-performing, supportable
  - Gartner: 90% of security breaches are due to patch management failures\*
- Replacing or upgrading hardware as needed; anticipating replacement needs and timing
- Managing data: archive/purge/retention
  - Can cause long-term performance issues if not addressed
- Anticipating and addressing future capacity needs

\*Executive's Guide to Information Technology, Baschab & Piot, 2007

# Best Practices: Proactive updates

---

- Defined ownership, accountability, funding for proactive maintenance
  - May get lost in crisis response if just “done in spare time”
- Standards for hardware, software, OS, patch levels
  - Including infrastructural hardware and software (network, storage, interface “plumbing”)
- Means for assessing possible updates and determining which will become new standards
  - Including means to **verify** stability, performance, usefulness in the relevant infrastructure
  - Defined **accountability** for identification and evaluation of possible updates
  - Documented procedures where needed

# Best Practices: Data management

---

- Defined expectations from the business as to:
  - Expected volume (transactions, data)
  - Projected growth over time
  - Data that must be retained, can be archived, can be purged
- Defined data archive/purge/retention plan
  - Agreement with IT and business on what will be purged and archived at what frequency
- Defined ownership, accountability, resourcing for data management
  - Clarity around who will do the work, how it will be organized/funded, how it will be unimpeded by crisis response
- Data management may be included in an SLA

# Best practices: Anticipating and addressing future capacity needs

---

- Topic of the next section...

# Managing IT Operations

Monitoring, trending, planning

# Long-term Capacity Planning

- Capacity planning defined:

“The process for identifying and planning the future resource requirements of the IT infrastructure”

*From CIO Wisdom: Best Practices from Silicon Valley's Leading IT Experts*

- “Capacity” includes:

- Throughput versus total available capacity
- “Average” versus “peak usage” needs
- Technical capacity: Network, storage, memory, CPU, etc.
- Physical capacity: Space, power consumption, etc.
- Functional capacity: E.g., can process “X transactions per second” using current systems and infrastructure

- Examined by system, business area, geography, etc.

# Capacity planning inputs

---

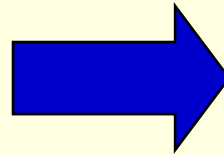
- Some of this can be examined from a purely technical perspective; some may need business input
  - Throughput versus total available capacity
  - “Average” versus “peak usage” needs
  - Technical capacity: Network, storage, memory, CPU, etc.
  - Physical capacity: Space, power consumption, etc.
  - Functional capacity: E.g., can process “X transactions per second” using current systems and infrastructure
- Existing system: Can look at current metrics, trends
- New system: Must project based on business information
  - # users, their locations, # transactions, peak usage, etc.

# Capacity planning considerations

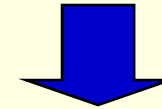
## Inputs

- Current usage patterns (average, peak)
- Current utilization of resources
- Current performance measures
- Current available capacity and bandwidth
- Support case load related to utilization or performance
- Trends in usage, utilization, performance (rates of change in current values)
- Future business needs and plans; current un-met needs
- Other planned changes to dependent infrastructure

## Immediate Actions

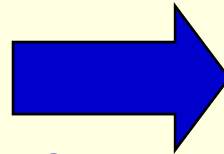


- Archive and/or purge non-essential data
- Add additional stop-gap resources (hardware, storage, network) as needed
- Define longer-term plans



## Example

	Current	Projected
Network throughput (Average / Peak)		
Storage capacity used		
Memory consumption (Average / Peak)		
Data center floor space		
Power consumption (Average / Peak)		



## Capacity Plans

Current information

Forward-looking projections



# Challenges in capacity planning

---

- Accountability for the activity – who’s job is it?
  - Process, periodicity, action plans
- Availability and usage of monitoring and trending information... maybe...
- Funding to act upon identified actions
  - Management can be “reactive” too... if it’s not a fire today, it might not get funding
- Specification of functional capacity needs
  - Requires conversation and collaboration with the business
- Ability to translate functional needs to physical and technical specifications... maybe...
  - Possible implications for performance testing environments

# Disaster Recovery Planning

---

- Sometimes referred to as business continuity planning
- Not a small topic – and technical compared to some of the class topics...
- ...But critical to the “business-driven” view: When critical systems are down – the business is down!
- Recent “lessons”: September 11, 2001, Hurricane Katrina

# DRP: Things to think about

---

- Which systems are critical?
- Redundancy of systems, data centers, and work sites
  - Backing up systems and data is not enough
- Identification of key personnel, communication mechanisms, leadership in event of crises
  - Leadership in a crisis is a typical failure point – whether you are talking about DR or any type of crisis
  - Senior management involvement from IT and business should be part of the plan definition and execution
  - Accountability from each organization should be defined
- Execution of the plan may fail if it was never tested!

DR Plan should be based on **business requirements** – and should be communicated to and agreed upon with the business



# Managing IT Operations



Change Management

# Questions in Change Management

---

How do changes get **funded**? **Who decides** which ones will get approved? On what basis? How does it integrate into **portfolio management**?

How are changes **delivered**? Releases? Ad-hoc? How are change **prioritized** for delivery dates? Who **coordinates**? Who **schedules** the changes?

How do changes get **validated**? Where? By **whom**? What are the **criteria** for accepting them into Production? Who owns which environment? Who makes **decision to promote** or not?

Who has **access** to make changes? Of which types? How is access controlled? How are executed changes **documented**?

What processes are used to manage data and code affected by **Sarbanes-Oxley**?

# Change Management Activities

---

## ■ Approval Activities:

- Identifying and documenting candidate changes
- Prioritizing changes
- Allocating resources for change implementation
- Evaluating risks and impacts, including impacts to business users and to interfaced systems
- Formally approving (or rejecting) proposed changes
- Scheduling changes (ideally to release windows)

## ■ Implementation Activities:

- Defining validation and roll-back approaches
- Implementing and validating a change
- Communicating upcoming changes to user community
- Validating success or failure of production deployment

# Best Practices and Problem Areas in Change Management

## Best Practices

- Process for and ownership of prioritization and approval
- Release orientation, with formal validation processes
- Documentation of planned changes, planned testing, test outcomes
- Audit of implemented changes (approver, when, what, implementer)
- Control over access and approvals for test and production environments
- Resource planning aligned with prioritization – ability to meet critical needs

## Problem Areas

- Lack of control over changes to test and production environments
- Inadequacy of test environments – not representational of production
- Unclear ownership and accountability (“support” versus “project” activity)
- No means to prioritize changes; not linked well to portfolio planning, resource allocation
- Inappropriate balance of control and agility – too little control, too much overhead – or both!

Resources are always limited in some way...so requests should be prioritized and resourced / scheduled accordingly...



# Managing IT Operations



Overall support structure



# “Typical” support structures

Support is usually talked about in terms of “tiers”, for example:

## Tier 1 Support

- Trouble ticket created
- Basic user questions, common issues, or straightforward problems may be resolved at this level

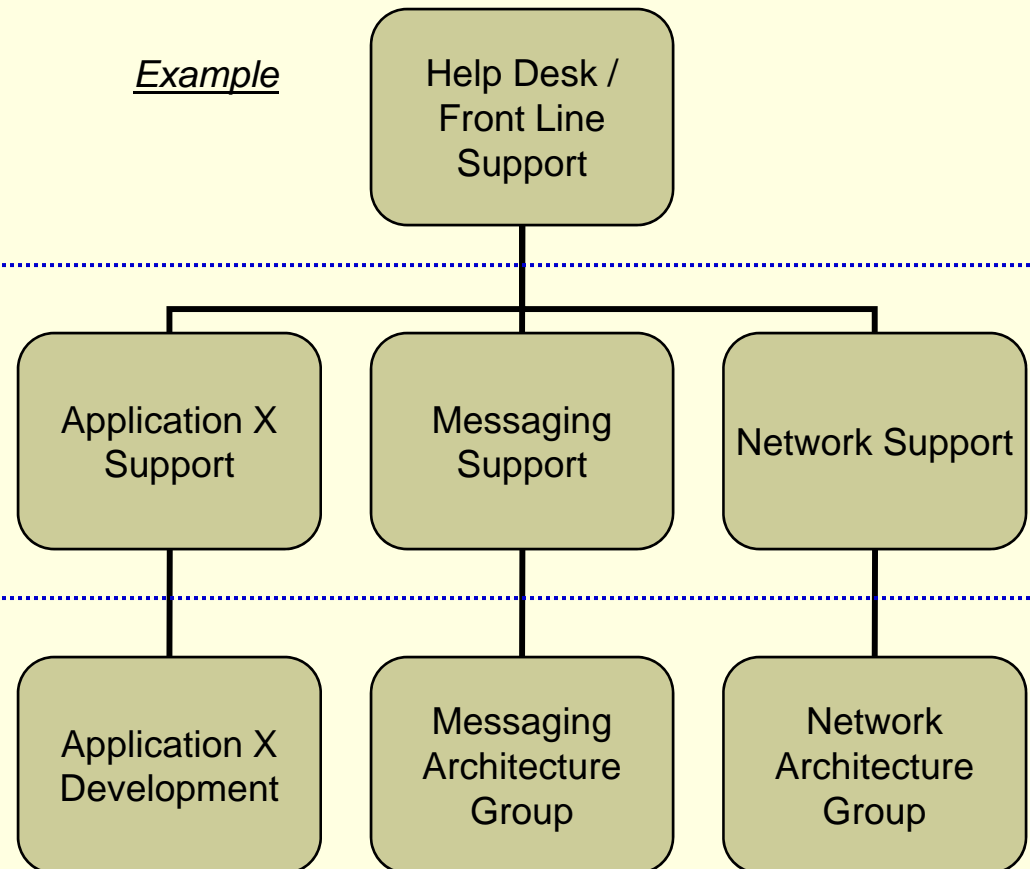
## Tier 2 Support

- Request / issue handed off to group responsible for that problem category
- Typically addressed by an SME within that category

## Tier 3 Support

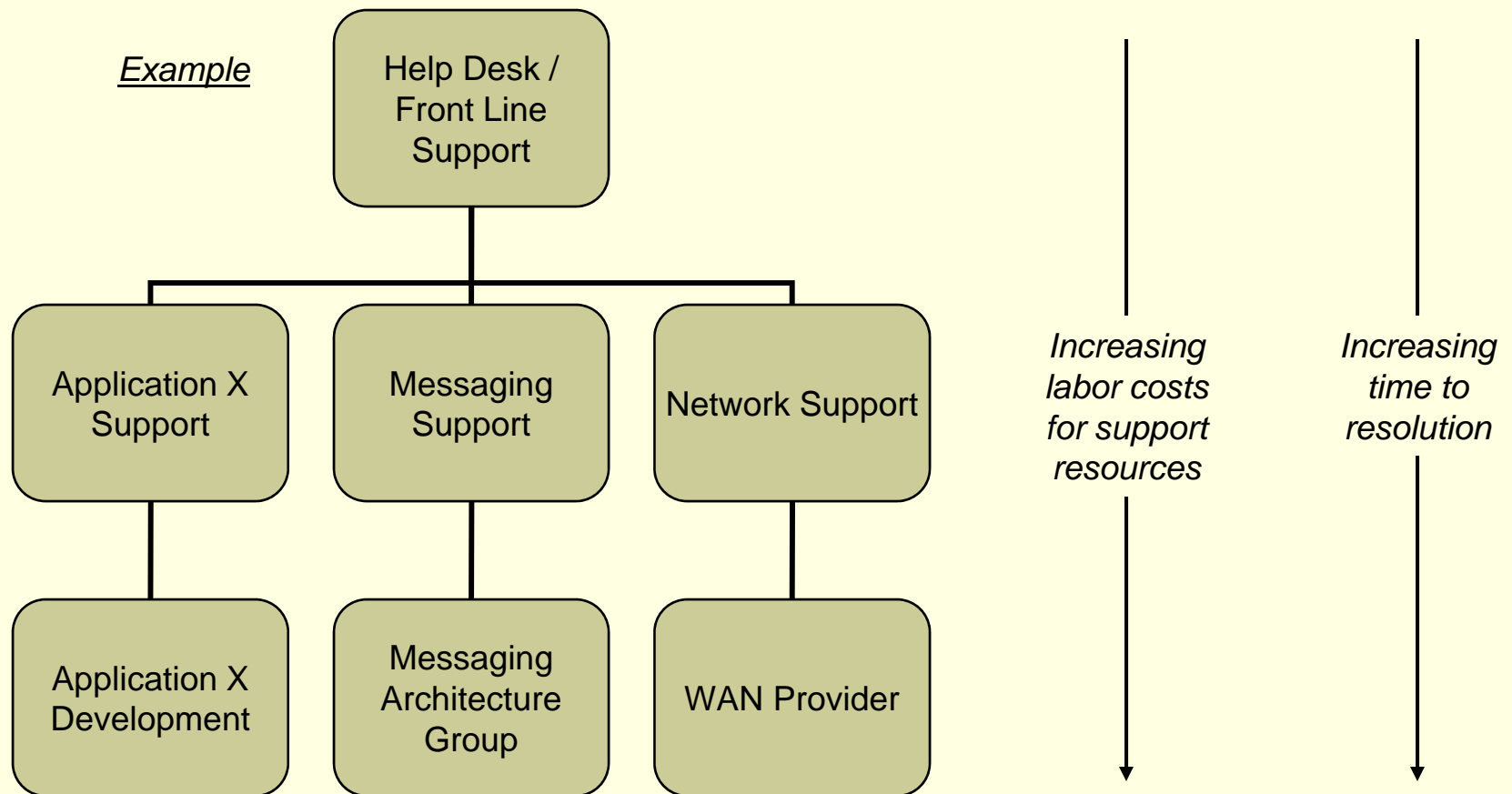
- Unresolved issues handed off to “experts” in the appropriate area
- Experienced, more expensive support (or development) resources

Example



# Goals of system support

Goal: Address issues as quickly and as cost-effectively as possible



# Some means to manage support costs

---

- Self-service means to provide users with information
  - E.g., web site content, published “FAQs”
- Automated means for common requests
  - E.g., new login accounts, password change, etc.
- Use of knowledge bases to help support personnel identify common issues and solve them quickly
  - May also allow problems to be solved by a lower support tier
- Alignment of support levels with criticality of support
  - Promise rapid response only where needed; allow slower response where practical
  - May use combination of type of issue / request along with priority of system and/or user group
  - Support funding / charge-backs based on level needed

# Service Level Management

---

“Typical” items included:

**Agreement** between **IT** and **business** that **defines expectations** for system performance, stability, and support response, such as:

- **Response time** to reported issues or requests
- **Time-to-resolution** for reported issues or requests
- **Availability** / uptime
- **Performance** (e.g., time to complete a particular business action)
- **Support coverage** times (e.g., 24x7, 8x5, etc.)

Target values for each may vary by priority of the system or of particular functionality within a system, criticality of the issue, user group, geographic location, etc.

Key need is that an agreement between IT and the business exists and is documented – if the support needs are not known, they may not be met!

# Service Level Management

- Service level is typically documented in a “Service Level Agreement”, or SLA
- Other factors in an SLA could include:
  - What happens if an SLA target is not met
  - Maintenance / upgrade responsibilities and expectations
  - Roles/responsibilities between IT and business groups
- May be defined based on priority of a system or business function – and/or priority of request/issue

- Each **level of service** has an associated **cost**
- If higher level of support is needed ➡ requires staffing, tools, etc. as enablers
- “Premium” support should result in **greater cost** (chargeback, allocation) to the **business** requesting that increased level of support
- **Ties business need** for support to the level of **support promised**

# Service Level target examples

## Example

Business Criticality*	Support Coverage	Target Availability	Time-to-Response	Time-to-Resolution
<b>Business Critical</b>	24x7	99.999%	Urgent issue: 10min High: 1 hour Medium: 4 hours Low: 1 day	Urgent: 1 hour High: 1 day Medium: 2 weeks Low: not defined
<b>High Priority</b>	24x7	99.5%	Urgent issue: 1 hour High: 2 hours Medium: 4 hours Low: 1 day	Urgent: 4 hours High: 1 day Medium: 2 weeks Low: not defined
<b>Medium Priority</b>	8x5	99%	Urgent issue: 10min High: 1 hour Medium: 4 hours Low: 1 day	Urgent: 4 hours High: 2 days Medium: not defined Low: not defined
<b>Low Priority</b>	8x5	95%	Urgent issue: 10min High: 1 hour Medium: 4 hours Low: 1 day	Urgent: 1 hour High: 1 day Medium: not defined Low: not defined

\* Criticality of the system as a whole, of specific functionality within a system, or of a particular user group

# Key points in SLA definition

---

- By nature, support tends to be reactive, not proactive
- If you want proactive actions to be taken – consider documenting expectations in the SLAs:
  - Responsibility and frequency of patching, upgrades, etc.
  - Performance monitoring, trending, capacity planning
  - Presence of a business-approved disaster recovery plan for critical systems – or a plan to complete one by a target date
  - Periodic examination of case data for recurrent problem, themes, and trends – and action plans when needed
- Identify what is supposed to happen when SLA targets or responsibilities are not met
  - If there is no incentive to comply – or no measurement of compliance – outcomes may not be in compliance with SLA

# SLA definition

---

- We are going to return to the concept of a documented agreement between business and IT at the very end of the class...



# Managing IT Operations

Case management

# Case Management versus Problem Management

---

- ITIL uses concept of “Problems” versus “Incidents”
- For example:
  - User calls helpdesk to report inability to log in to System X
  - HelpDesk assigns case 1234
  - Support staff member resolves the immediate issue by (for example) killing hung sessions
  - “Incident” is over, user is happy
  - However...IT fails to notice that this issue happens on average 15 times per month
  - AND the functionality and users affected are business critical
  - In ITIL terminology, this is a “Problem”
- Recurrent issues sometimes form a good fraction of the cases in the overall caseload

# Examples of unaddressed problems

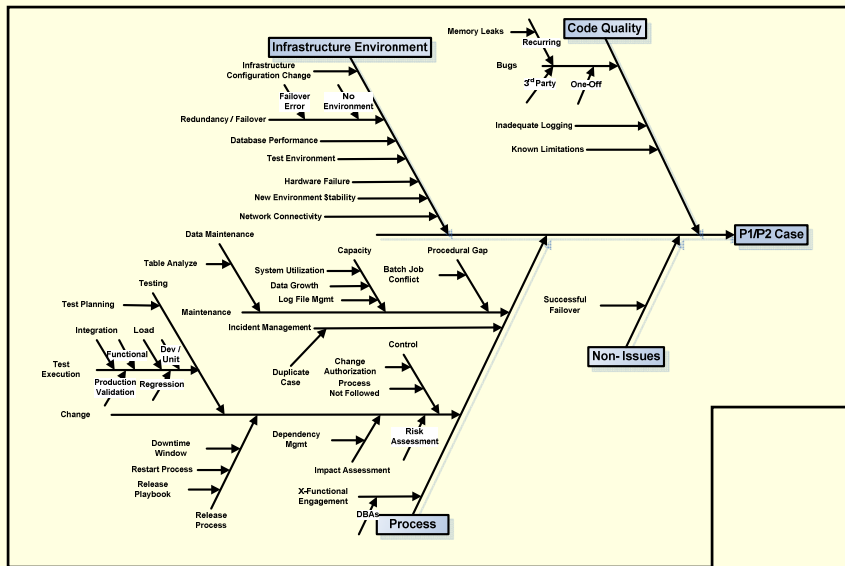
- Recurring bugs in specific areas of code
- Increasing performance issues (indicative of lack of trending & capacity planning)
  - Could be due to bandwidth/capacity constraints in applications, infrastructure, hardware, network, storage, etc.
- Sometimes, process and/or training issues:
  - E.g., poor coding, quality, or change documentation, or change control practices
- Potentially, emerging hardware issues

**Common “root cause” area: Poorly implemented change**

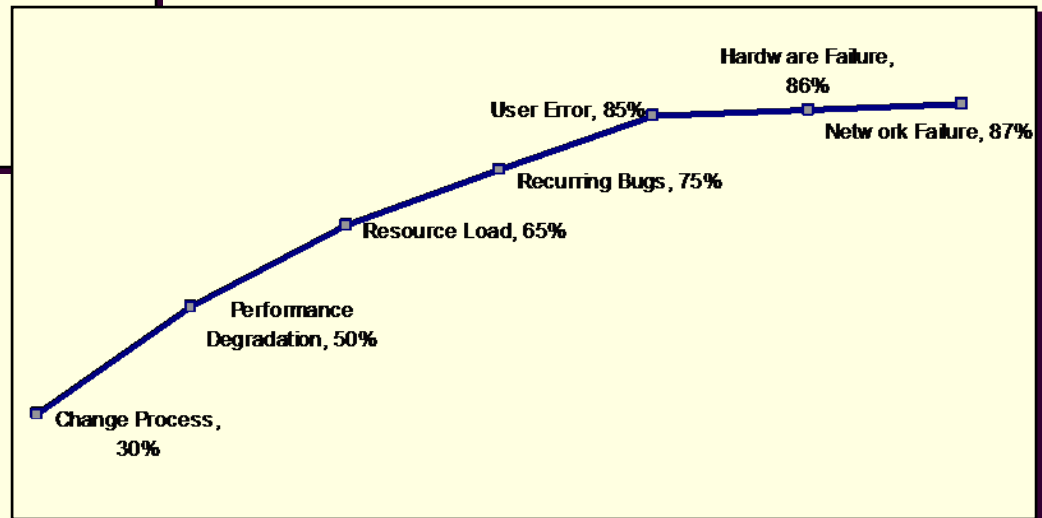
If no one is **accountable for identifying recurrent problems**, themes, and trends – then probably **no one will take responsibility**...Over time, the business may become increasingly unhappy with service provided by IT...until someone gets fired

# Example root cause analysis

## Fishbone diagram



## Pareto Analysis of Root Cause Areas



# Summarized issues in support

---

- Lack of SLA definition or any formal definition of support needs, expectations, commitments
- No process or ownership to measure IT's performance relative to business expectations
- Reactive focus – solve the current issue, then the next issue
- No process, accountability, or analytical skill-set to identify recurring “problems” from case histories
- Sometimes, support turnover & knowledge retention issues

# Suggested reading for more information...

---

- CIO Wisdom, Best Practices from Silicon Valley's Leading IT Experts, Dean Lane, 2004, ISBN 0-13-141115-2
  - Chapter 12 (written by Joe Feliu)
- The Executive's Guide to Information Technology, Baschab and Piot, 2007, ISBN 978-0-470-09521-8
  - Seems to use ITIL-like terminology, but does not refer to ITIL
  - Chapter 8, IT Operations and Infrastructure (from p.241)
  - Chapter 9, Problem Management
    - Has some nice information on help desk call routing, metrics, severity definitions, call volume patterns, etc.
  - Chapter 10, Application Management (p. 321-325)
    - Information on setting up SLAs, roles, other key needs in supporting newly-deployed applications
- IT Infrastructure Library (ITIL) reference material

# Balancing Proactive versus Reactive

	Reactive	Proactive
<b>Support</b>	<ul style="list-style-type: none"> <li>• Provide helpdesk / front-line support; capture issues noted through helpdesk functions (e.g., calls, email, web)</li> <li>• Provide additional support tiers as applicable</li> <li>• Respond to and resolve issues noted through helpdesk functions (e.g., calls, email, web) and real-time monitoring tools</li> <li>• Assist users</li> </ul>	<ul style="list-style-type: none"> <li>• Identify recurring problems or themes in common issues; identify problem resolutions and implement</li> <li>• Define support structure, staffing, expertise needed to cover all areas within scope of support (infrastructure, business applications, etc.)</li> <li>• Measure resolution metrics (e.g., time-to-resolution) and satisfaction metrics; identify and complete actions needed</li> </ul>
<b>Maintenance</b>	<ul style="list-style-type: none"> <li>• Contribute to issue identification and resolution for back-end services and components</li> <li>• Return components to operational status as issues identified</li> <li>• Apply patches and fixes as needed to address operational issues</li> <li>• Restore lost data when needed</li> </ul>	<ul style="list-style-type: none"> <li>• Perform Backup for all key systems and data</li> <li>• Apply patches, fixes, upgrades as needed</li> <li>• Perform proactive hardware maintenance</li> </ul>
<b>Monitoring, Trending, and Planning</b>	<ul style="list-style-type: none"> <li>• Monitor key components to identify load, performance, availability issues*</li> <li>• Ensure issues noted through monitoring are captured, tracked, and resolved</li> </ul>	<ul style="list-style-type: none"> <li>• Periodically review load versus capacity trends*, identify future capacity needs, and define plan to meet needs</li> <li>• Track and manage software and hardware assets</li> <li>• Monitor and manage system security</li> </ul>

# Managing IT Operations

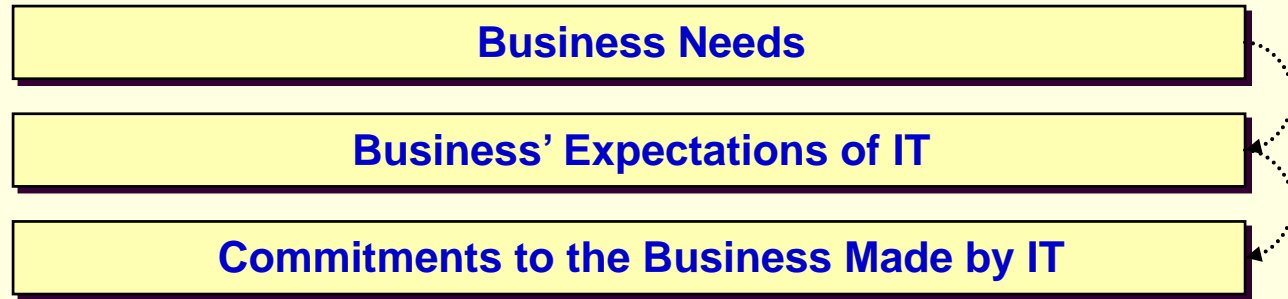
Funding, budgeting, governance, and  
organizational implications



# Getting started

---

- To get to the right answers in this space, start with:



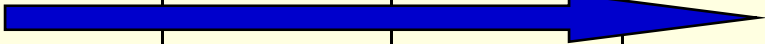
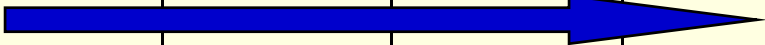




- Don't know what those are?
  - How do you know staffing required to provide support?
  - How do you know staffing requirements for maintenance activities?
  - How will you provide the performance and stability the business needs – if you don't know what they need?
  - How will you continue to provide functionality they need how they need it – if you don't know what they need?

# Business commitments to accountability, funding, approvals

## Service Level Agreement (SLA)

- System priority
- Critical functions
- Target availability
- Target performance
- Support coverage
- Support response
- Proactive maintenance commitments
- Means for change management
- Expected change volume
- Redundancy needs and means
- Costs and staffing

	Accountability	Staffing Needed	Budget Needed	Approval Processes
Basic Incident Response				
Problem Management				
Basic System Maintenance				
Change Management				
Performance & Capacity Planning				
Disaster Response and BC Planning				

# Session wrap-up

Recapping major topics

# General IT failure categories

Dimension	Notes
<b>Project execution</b>	Often the key focus area for “IT failures”, and subject of whole certification programs – and a key failure area regardless
<b>Alignment</b>	Alignment of stated priorities with real allocation of time, money, people – often mismatched
<b>Communication</b>	Especially across “silos” – too little contextual information for full effectiveness
<b>Accountability</b>	If no one owns the problem, no one will solve it. Metrics, visibility, incentives may be required to address gaps
<b>“Customer” experience and perception</b>	The result of gaps between expectations and outcomes

# Some notes for the business folks...

---

- IT folks do not always understand your job or your processes in depth
- If you didn't tell them, they don't know...
- If you didn't say it was required, it likely will not be delivered...
- If you did specify that it was required, it will not be tested, either...
- If you fund IT work for projects and you don't provide adequate funding for quality solutions, you won't get them – you'll get shortcuts on solutions or quality
- If proactive maintenance isn't funded, you likely won't get that either

# ...And some notes for the IT folks

---

- If you have no insight into business context, ask...
- If you want to know how it will be used, go find out...
- If you think there is insufficient time or money for the “right” solution, identify critical items and options
- If you want to deliver software that performs – ask what “performs” means...
- If you want to know if they will be satisfied with a solution, show them and find out – before testing!
- If you want to know what they expect after it goes live, how often, for how long – discuss it
- And, if you can...find some time to shadow your business counterparts and watch them do their job

# Your input

## Comment

- High overhead to make small changes; even small changes required to go through “releases”
- IT is underfunded
- Release process does not meet business goals – only benefits IT
- Business expects IT to own the Business Requirements Document
- Business also expects IT to own the testing; UAT not viewed as critical from Business perspective
- Business does not seem to want involvement with definition of test approach, scenarios, etc.
- IT’s version of status (red/yellow/green) does not seem to match Business’ view: IT thinks they can “recover” and always shows “green” until it is not
- Business wants to add requirements to the very last minute and puts pressure on IT to perform when the requirements have changed
- IT can come across negatively when questioning the business justification of a request. Typically seems to turn out to mean that they have a hard time supporting the request, not that they are questioning the need for the request
- “Us versus Them” mentality is deeply rooted in the organization. Need to get to a point where everyone is working to the same goals
- Need shared metrics that tie together Business and IT

# Closing comments

---

- “Business-oriented” versus “Business-driven”
  - Orientation around maximizing business value <> “do what the Business tells you”
  - Particular things get short-changed when you do this
- Criticality of “criticality” – IT needs priority information
  - Of projects, requirements, systems, incidents, data,...
  - There are always bottleneck resources, even for well-funded organizations
- IT delivery
  - Need to figure out how to meet needs of the Business without over-burdening all things with process overhead
  - Conversation & agreement with the business: Cost of quality versus risk (and resulting cost) of non-quality