Structured Systems Analysis and Design

Sogeti University
February 1, 2011
Dr. Kevin P. Duffy
What is SSAD?

A few definitions to start everybody off on the same page ….

- **System**: A collection of *interrelated* components that function together to achieve some outcome (goal or objective)

- **Structured Systems Analysis and Design**: an organizational process used to develop and maintain computer-based information systems (both business and systems professionals participate in SSAD).
What is SSAD?

Hence, a *Systems Development Methodology* is a standard process followed in an organization to conduct all the steps necessary to analyze, design, implement, and maintain information systems.

- **Why use a “standard” process?**

The most common, or traditional, methodology is the *Systems Development Life Cycle*, or SDLC.
General Systems Theory (or, Systems Theory) became popular as a discipline in the 1940s and 1950s.

Systems theory has proven useful in Systems Analysis and Design as a means of providing definitions and explaining system behavior(s).

Systems theory notes that “All Systems are composed of sub-systems, and all systems are themselves sub-systems to larger systems”

From this foundation, systems theory permits “Decomposition,” or the ability to break a large system down into its component parts.

What is gained by “breaking something into component [“smaller”] parts?

- the ability to better grasp system functionality
- easier to work with smaller components and/or systems
- easier to maintain, update, and replace components
General Systems Theory

Systems Theory provides us with a definition for systems

Systems theory defines nine characteristics of a system:

- System components
- Interrelationships
- Boundary
- Purpose
- Environment
- System interfaces
- Input
- Output
- Constraints

Each of these characteristics provide us with a means of “checking” the system
Analysis and Design Landscape

1950s: focus on efficient automation of existing processes
1960s: advent of 3GL, faster and more reliable computers
1970s: system development becomes more like an engineering discipline
1980s: major breakthrough with 4GL, CASE tools, object-oriented methods
1990s: focus on system integration, GUI applications, client/server platforms, the Internet

The New Century: Web application development, wireless PDAs, component-based applications
Larger Picture Problems

Systems come in late
Systems come in over-budget
Systems tend to be considered as failures (up to 76%) for one reason or another
Technology changes, and systems don’t keep up with technology (hence, the existence of so-called “legacy” systems)
The SDLC is the traditional methodology used to develop, maintain, and replace information systems.

The activities falling throughout analysis, design, and development are categorized into “phases” (the number of phases will vary from one writer to another).
The Phases of the SDLC include:

- Planning
- Analysis
- Design
- Implementation
- Maintenance

The cycle typically flows through numerous iterations.
The SDLC is often referred to as the “waterfall” model
**Planning**: an organization’s total information system needs are identified, analyzed, prioritized, and arranged

**Analysis**: System requirements are studied and structured

**Design**: A description of the recommended solution is converted into logical and then physical system specifications

- **Logical design**: all functional features of the system chosen for development in analysis are described independently of any computer platform

- **Physical design**: the logical specifications of the system from logical design are transformed into the technology-specific details from which all programming and system construction can be accomplished

**Implementation**: the information system is coded, tested, installed and supported in the organization

**Maintenance**: the system is systematically repaired and improved
Structured Systems Analysis and Design – The Analyst’s Role

Sogeti University
February 1, 2011
Dr. Kevin P. Duffy
The systems analyst functions within the organization as a *liaison* between the business professionals and the information systems or computer science professionals.

In this capacity, a number of skills are required of the analyst:

- Working knowledge of information technology
- Computer programming experience and expertise
- General business knowledge
- General problem-solving skills
- Good interpersonal communication skills
- Good interpersonal relations skills
- Flexibility and adaptability
- Character and ethics
By "Problems" that need solving, we mean:

- **Problems**, either real or anticipated, that require corrective action

- **Opportunities** to improve a situation despite the absence of complaints

- **Directives** to change a situation regardless of whether anyone has complained about the current situation
In the role of a liaison, one of the primary roles of the systems analyst is to interact with organizational stakeholders, including the end-users of the system.

Why interact with end-users?

To gain *Buy-In* for the system

To calm user fears

To gain the end-users’ understanding of the process

How it operates

Why it operates as it does

What the process accomplishes

Where a process “fits” into a larger process
There are a number of reasons why a system changes

**Business Drivers**

- Globalization of the Economy
- Electronic Commerce and Business
- Security and Privacy
- Collaboration and Partnership
- Knowledge Asset Management
- Continuous Improvement and Total Quality Management
- Business Process Redesign
Technology Drivers

Networks and the Internet

Mobile and Wireless Technologies

Object Technologies

Collaborative Technologies

Enterprise Applications
Systems Analyst Role

Regardless of the driver of change, a systems analyst must
analyze and understand the problem or opportunity
identify solution requirements or expectations
In today’s Information Technology environment, there are numerous sources for software and systems acquisition

- Outsourcing
- Custom built in-house
- Customized off-the-shelf software (COTS)
- Application Service Providers (ASPs)
- Open source software
- Enterprise wide software solutions

The analyst must be able to compare software against a variety of metrics:

- cost
- functionality and response time
- flexibility
- ease-of-use
- vendor support
Project Management and Planning

As you might imagine, an analysis and design project can become a huge undertaking!

Because of the size of projects, and because projects are a team effort (too large, involved, and complicated to be an individual project), project management techniques are invaluable.

Project Management helps to ensure that

- Customer expectations are met
- Budget and Time constraints are satisfied
Project

- A planned undertaking of related activities to reach an objective that has a beginning and an end

Project management

- A controlled process of initiating, planning, executing, and closing down a project

Planning a project

Define clear, discrete activities and the work needed to complete each activity

Tasks

- Define project scope, alternatives, feasibility
- Divide project into tasks
- Estimate resource requirements
- Develop preliminary schedule
- Create preliminary budget

All of these activities or tasks must be re-visited, and perhaps updated, throughout the entire project
Project Management and Planning

Project Scope

The determination of scope will depend on the following factors:

- Which organizational units (business functions and divisions) might be affected by or use the proposed system or system change?
- With which current systems might the proposed system need to interact or be consistent, or which current systems might be changed due to a replacement system?
- Who inside and outside the requesting organization (or the organization as a whole) might care about the proposed system?
- What range of potential system capabilities is to be considered?

Project scope is an iterative process, and will evolve:

- as more information is collected
- as the project proceeds.
Another task is to undertake a feasibility study of a proposed project when a request for service is received.

Feasibility is the measure of how beneficial or practical the development of an information system will be to an organization. There are numerous types of feasibility to be considered:

- **Operational feasibility** is a measure of how well a solution meets the identified system requirements to solve the problems and take advantage of the opportunities envisioned for the system.

- **Cultural (or political) feasibility** is a measure of how people feel about a solution and how well it will be accepted in a given organization climate.

- **Technical feasibility** is a measure of the practicality of a specific technical solution and the availability of technical resources and expertise to implement and maintain it.

- **Schedule feasibility** is a measure of how reasonable the project timetable is.

- **Legal feasibility** is a measure of how well a solution can be implemented within existing legal and contractual obligations.
Project Planning may include scheduling diagrams, such as a Gantt chart (where horizontal bars represent task duration) or network charts (PERT/CPM) where activities and links represent task dependencies.

Gantt Chart:
Project Management and Planning

PERT/CPM chart:
Economic feasibility is a measure of the cost-effectiveness of a project or solution, and involves performing a cost-benefit analysis of an IT investment.

Time Value of Money: undertaking a cost-benefit analysis of an IT investment.

\[ PV_n = Y \cdot \left[ \frac{1}{(1 + i)^n} \right] \]

Where
- \( PV \) = present value
- \( n \) = year (time)
- \( Y \) = payment; some portion of principal
- \( i \) = interest rate
Project Management and Planning

Use a spreadsheet for calculations!
The goal of project management and planning is to begin to
gain an understanding of the project
establish realistic goals that can be met
Structured Systems Analysis and Design, Requirements Elicitation

Sogeti University
February 1, 2011
Dr. Kevin P. Duffy
Requirements Determination

Traditional Methods for Determining Requirements

- Interviewing individuals
- Interviewing groups
- Observing workers
- Studying business documents
One of the primary ways analysts gather information about an information systems project

Interview Guide is a document for developing, planning and conducting an interview

Plan the interview.

• Prepare interviewee: appointment, priming questions.
• Prepare agenda, checklist, questions.

Listen carefully and take notes (tape record if permitted).

Review notes within 48 hours.

Be neutral.

Seek diverse views.

Each question in an interview guide can include both verbal and non-verbal information.

– **Open-ended questions**: questions that have no pre-specified answers
– **Closed-ended questions**: questions that ask those responding to choose from among a set of specified responses
Requirements Determination

Direct Observation
- Watching users do their jobs
- Obtaining more firsthand and objective measures of employee interaction with information systems
- Can cause people to change their normal operating behavior
- Time-consuming and limited time to observe

Analyzing Procedures and Other Documents

Document Analysis
- Review of existing business documents
- Can give a historical and “formal” view of system requirements

Types of information to be discovered:
- Problems with existing system, or Opportunity to meet new need
- Organizational direction and Names of key individuals
- Values of organization
- Special information processing circumstances
- Reasons for current system design
- Rules for processing data
Requirements Determination

When analyzing Procedures, be aware:

**Formal Systems:** the official way a system works as described in organizational documentation (i.e. work procedure)

**Informal Systems:** the way a system actually works (i.e. interviews, observations)

**Useful document: Business form**

- Used for all types of business functions
- Explicitly indicate what data flow in and out of a system and data necessary for the system to function
- Gives crucial information about the nature of the organization
**Joint Application Design** (JAD) may be useful:

- Brings together key users, managers, and systems analysts
- Purpose: collect system requirements simultaneously from key people
- Conducted off-site
- But, is *resource-intensive*

**JAD Participants**

- **Session Leader**: facilitates group process
- **Users**: active, speaking participants
- **Managers**: active, speaking participants
- **Sponsor**: high-level champion, limited participation
- **Systems Analysts**: should mostly listen
- **Scribe**: record session activities
- **IS Staff**: should mostly listen
Structured Systems Analysis and Design, Modeling

Sogeti University
February 1, 2011
Dr. Kevin P. Duffy
Modeling, or Diagramming, system requirements

- Assists by structuring requirements
- Graphically represent the processes that capture, manipulate, store, and distribute data between a system and its environment and among system components.
- A process is a systematic series of actions directed toward an end or outcome
- A Data Flow Diagram (DFD) is a process modeling methodology
  - DFDs should be drawn for both the current and the proposed system
  - DFDs are simple to draw (consist of very few symbols)
  - Because of their simplicity, DFDs are explained easily to users
  - DFDs are drawn for successive layers, with each layer delving further into the process and its functionality
- Above all, use diagrams as a communication device
Process modeling using data flow diagrams views a system:

- as a collection of processes
- processes interact with data entities
- processes accept inputs and produce outputs

Data Flow Diagrams are constructed using only a few symbols (hence, they’re easier to understand than typical flowcharts)
The highest level DFD is a “context diagram,” and consists of one and only one process, and absolutely zero data stores. It gives a very high level view of the system.

A context diagram is followed by a level 0 diagram, then a level 1 diagram, and so forth. The lowest level DFD is known as a “primitive” data flow diagram.

Context Diagram for an ATM System
The diagram shows both Input(s) and Output(s)
Only 1 process is shown on a Context Diagram – this process represents the system in its entirety
A Data Store is never shown on a Context Diagram
The inputs to a process are different from the outputs (processes transform inputs into outputs
All Communication must flow through a process
An entity cannot communicate directly with another entity (since an entity is external to the system)
An entity cannot access a data store without going through a process (since a data store is part of the system, and an entity is external to the system). Data cannot move directly from one data store to another; data movement must occur through a process
Data flows in only one direction at a time (data flow arrows are uni-directional)
All entities, processes, data flows, and data stores are labeled
There may be as many entities, processes, data flows, and data stores as needed.
The Level-0 diagram is followed by a Level-1 diagram

- One Level-1 diagram is drawn for each *process* appearing on the Level-0 diagram
- Inputs and outputs from the process should remain balanced with what is depicted on higher level diagrams
- When the process is *fully* understood, diagramming stops (this rule applies to the system as a whole, as well)
- The final set of data flow diagrams is referred to as *primitive* data flow diagrams

All *processes* must have input; if a process is making outputs without inputs, there is a miracle taking place

All *processes* must produce outputs; if a process is taking inputs but not producing outputs, there is a “black hole” (if an object only ever receives inputs without producing outputs, perhaps it should be modeled as a sink (entity) rather than as a process).

Processes typically have a verb (action) name, while entities and data stores typically have a noun name

A data flow leading into a data store is an update (i.e., write, modify, delete)

A data flow leading out of a data store is a use (i.e., read, retrieve)
Data may not flow directly from one data store to another data store. All movement of data must occur through a process.

Data may not move from a source into a data store. Data must be sent from the source into a process and from the process into the data store.

Data may not move from a data store into a sink. Data must be sent from the data store into a process and from the process into the sink.

Data may not move directly between a source and a sink. All data must flow through the system. If data does not flow through a process (i.e., through the system), it is external to the system, and therefore does not appear on a data flow diagram.

A data flow may be split on lower level DFDs (a composite data flow may be split into simpler data flows, where part of the data flows into one process, and part into another process).

Note also that the inputs to a process must be sufficient to produce the outputs (think back to “miracles” happening).
There may be four different sets of DFDs drawn in total:

- Current Logical (representation of current system with technology references removed)
- Current Physical (representation of current system, including technology references – data stores actually represent existing data, tables)
- Proposed Logical (representation of proposed system with technology references removed)
- Proposed Physical (representation of proposed system including technology references)

Time is not represented on a DFD

Several iterations of drawings will likely occur (software drawing tool use becomes a must)

Consider stopping drawing when each process captures a single decision, and when each data store represents a single data entity
Data modeling captures the data fed into the system or produced by the system, and provides a structure to house (store) the data.

The most common data model is the relational database model.

In the relational database model, data are grouped into subjects of interest (i.e., customer, product, sales, etc.).

The data model includes a 2-dimensional table, where a table often corresponds to a data subject of interest (i.e., customer, product, etc.).

A table resembles (for example) an Excel spreadsheet, where:

- Each column represents an attribute (or characteristic) of the data. A table column may be called a “field”
- Each row represents an instance of the data. A row may be called a “record”

Data is stored according to its subject, i.e., client data in one table, product information in another table, and order information in another table.

Each row of data contains a field (or fields) which uniquely identify a record, or a single instance of the data. This is known as the “primary key”
Keep in mind that data is a symbolic representation of a real customer, or of a real order. Because it is symbolic, it can’t be “seen” directly. Hence, the primary key is needed to distinguish one record from another record.

Choose a primary key which:

- is a characteristic which all instances of the data are likely to possess
- is a characteristic which will be non-null across all instances of the data
- is a characteristic which will be stable (non-changing) over time

Likely candidates or choices for primary key:

- Customer number
- Product Id or number
- Sales representative
- Employee number

Fields such as employee name are not good candidates to serve as the primary key. More than one person, or employee, may have the same name; more than one person, or employee, will not have the same employee ID.
In this sample entity

The name of the entity is "Customer" (shown in all caps)

The primary key field is Customer_ID

Sales_Rep_Id serves as a “foreign key”

<table>
<thead>
<tr>
<th>CUSTOMER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer_ID</td>
</tr>
<tr>
<td>Customer_Name</td>
</tr>
<tr>
<td>Customer_Addr1</td>
</tr>
<tr>
<td>Customer_Addr2</td>
</tr>
<tr>
<td>Customer_City</td>
</tr>
<tr>
<td>Customer_State</td>
</tr>
<tr>
<td>Customer_Zip</td>
</tr>
<tr>
<td>Customer_Phone</td>
</tr>
<tr>
<td>Customer_POC</td>
</tr>
<tr>
<td>FK Sales_Rep_Id</td>
</tr>
</tbody>
</table>
Why include a foreign key?

- A foreign key is used to link, or “relate” one table to another table.
- By definition, a foreign key is a field which serves as a primary key in another, related, table, but is not the primary key in the current table.

<table>
<thead>
<tr>
<th>CUSTOMER</th>
<th>SALES_REP</th>
<th>PK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer_ID</td>
<td>Sales_Rep_ID</td>
<td>←</td>
</tr>
<tr>
<td>Customer_Name</td>
<td>Sales_Rep_Name</td>
<td>←</td>
</tr>
<tr>
<td>Customer_Addr1</td>
<td>Sales_Rep_Addr1</td>
<td>←</td>
</tr>
<tr>
<td>Customer_Addr2</td>
<td>Sales_Rep_Addr2</td>
<td>←</td>
</tr>
<tr>
<td>Customer_City</td>
<td>Sales_Rep_City</td>
<td>←</td>
</tr>
<tr>
<td>Customer_State</td>
<td>Sales_Rep_State</td>
<td>←</td>
</tr>
<tr>
<td>Customer_Zip</td>
<td>Sales_Rep_Zip</td>
<td>←</td>
</tr>
<tr>
<td>Customer_Phone</td>
<td>Sales_Rep_Phone</td>
<td>←</td>
</tr>
<tr>
<td>Customer_POC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FK Sales_Rep_Id</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FK: Foreign Key
PK: Primary Key
The relational database model functions on the basis of shared data values among tables (i.e., the field value for Sales_Rep_ID is found in the SALES_REP table as well as in the CUSTOMER table).

The foreign keys serve to map out the relationships among the entities in the database.

A conceptual data model for a relational database is referred to as an Entity-Relationship Diagram (or E-R Diagram).

- The ability to relate data tables to one another allows us to capture intricate relationships among the data while storing data in stable structures.
  - Stability of the structures is enhanced through the *data normalization* process.

When beginning the data modeling task, walk through processes, data flow diagrams, business forms, etc.

- These are invaluable sources in determining what data is important to the business and to its systems.
- Group like data together.
- Ask whether multiple instances can be stored in one record of a table, or are multiple instances an indication that separate tables are needed.
Example:

- An employee may have more than one phone number
  - Office phone
  - Mobile phone
  - Home phone
- These would likely be stored as separate fields in one record

Example:

- A customer may place more than one order
  - Order placed on January 21
  - Order placed on May 27
  - Order placed on August 17
- These would likely be stored as separate records (separate instances of the Order entity)

How can you tell?

- Examine the primary key for the table. Is the data *functionally dependent* upon the primary key?
A **Use Case Model** comes out of the object-oriented perspective. However, the notion of a use case has become part of the business lexicon, and is likely to be a phrase often heard during an analysis and design.

**Use-case modeling:** the process of modeling a system’s functions in terms of business events, who initiated the events, and how the system responds to those events.

Use-case modeling has proved to be a valuable aid in meeting the challenges of determining what a system is required to do from a user and stakeholder perspective. It is now widely recognized as a best practice for the defining, documenting, and understanding of an information system’s functional requirements.

The use case model consists of 2 parts: the use case diagram, and the use case narrative.

The use case diagram graphically depicts the system as a collection of use cases, actors (users), and their relationships.

The use narrative is a textual description of the business event and how the user will interact with the system to accomplish the task.
Example Use Case Diagram
Example Use Case Narrative

Example: Library book

1. Check student status
2. Scan book
3. Assign due date
4. de-security book
5. update book status

1.a. Student status is not found. Notify customer.
1.b. Student status suspended. Refer to student account office.
To construct use cases:

- Identify business actors
- Identify business requirements use cases
- Construct use-case model diagram
- Document business requirements use-case narratives

An “actor” typically represents a role, rather than a particular person

- i.e., Manager, Sales Rep, etc.
- “Time” may be an actor (some use cases are triggered by time of day)
- Another system may be an actor (outputs from one system may be triggering inputs to another system)
Throughout the modeling process, it becomes increasingly important to fully utilize CASE tools

**CASE = Computer Assisted Software Engineering**

**CASE tools serve as a knowledge repository for the system**

- Include completeness and consistency checking
- Upper-CASE tools capture early phases of the SDLC
- Lower-CASE tools capture later phases of the SDLC
- I-CASE (Integrated CASE) tools capture throughout the entire SDLC

**When a system is completely documented within a CASE tool, updates and changes are easy to manage**

**Sophisticated CASE tools can create a system’s data structures, applications, and interfaces from the material recorded**