

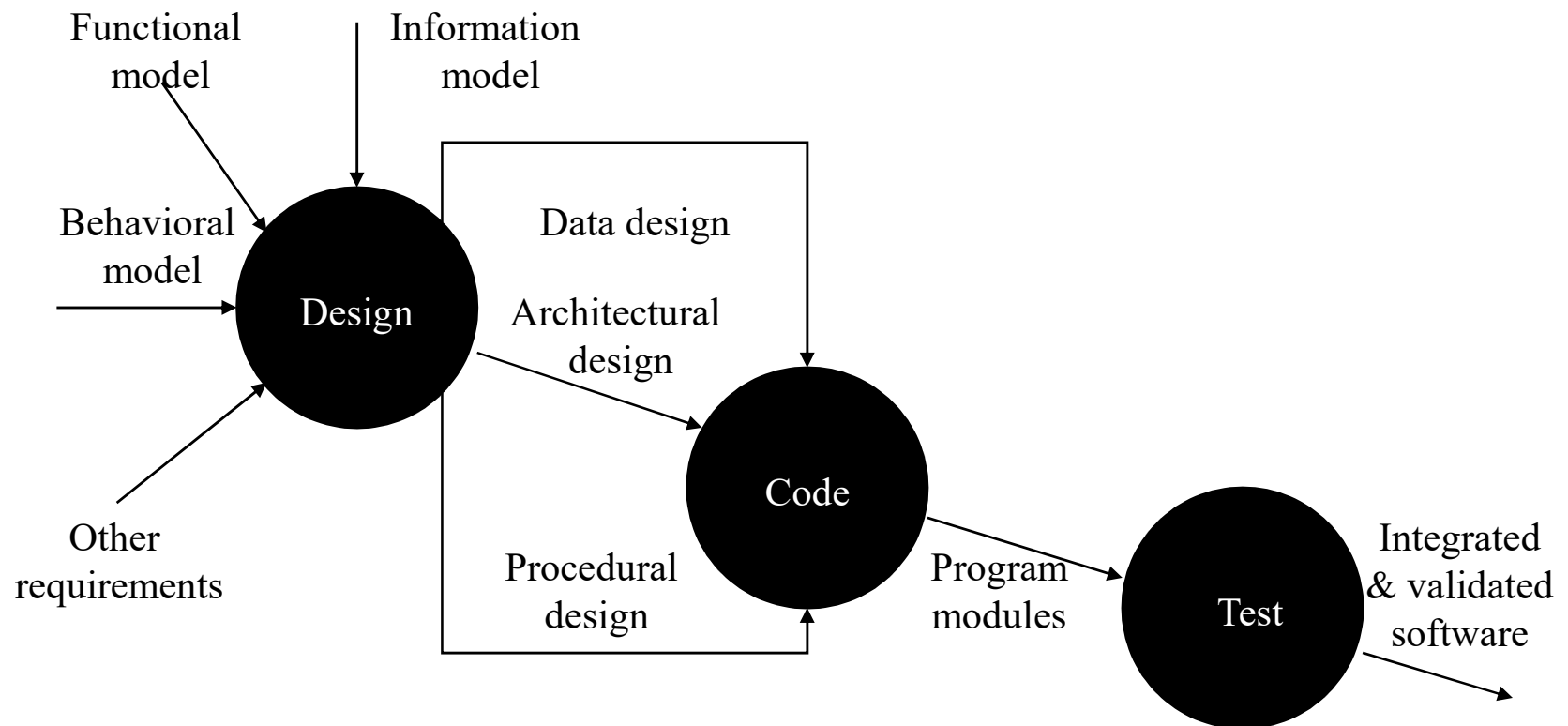
Architectural Design

Introduction

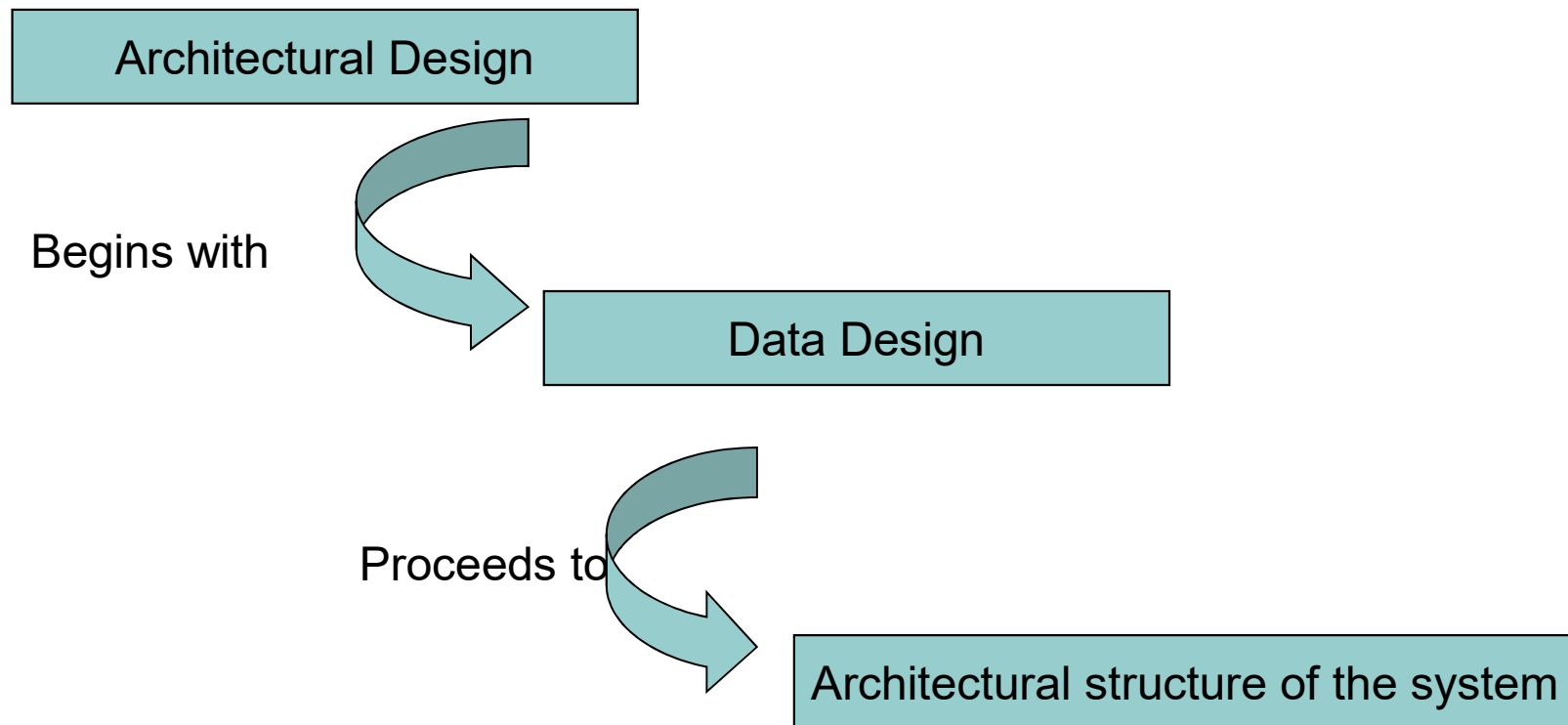
- Design has been described as a multistep process in which representations of data and program structure, interface characteristics, and procedural detail are synthesized from information requirements.
- Design is an activity concerned with making major decisions, often of a structural nature.

-
- Design builds coherent, well planned representations of programs that concentrate on the interrelationships of parts at the higher level and the logical operations involved at the lower levels

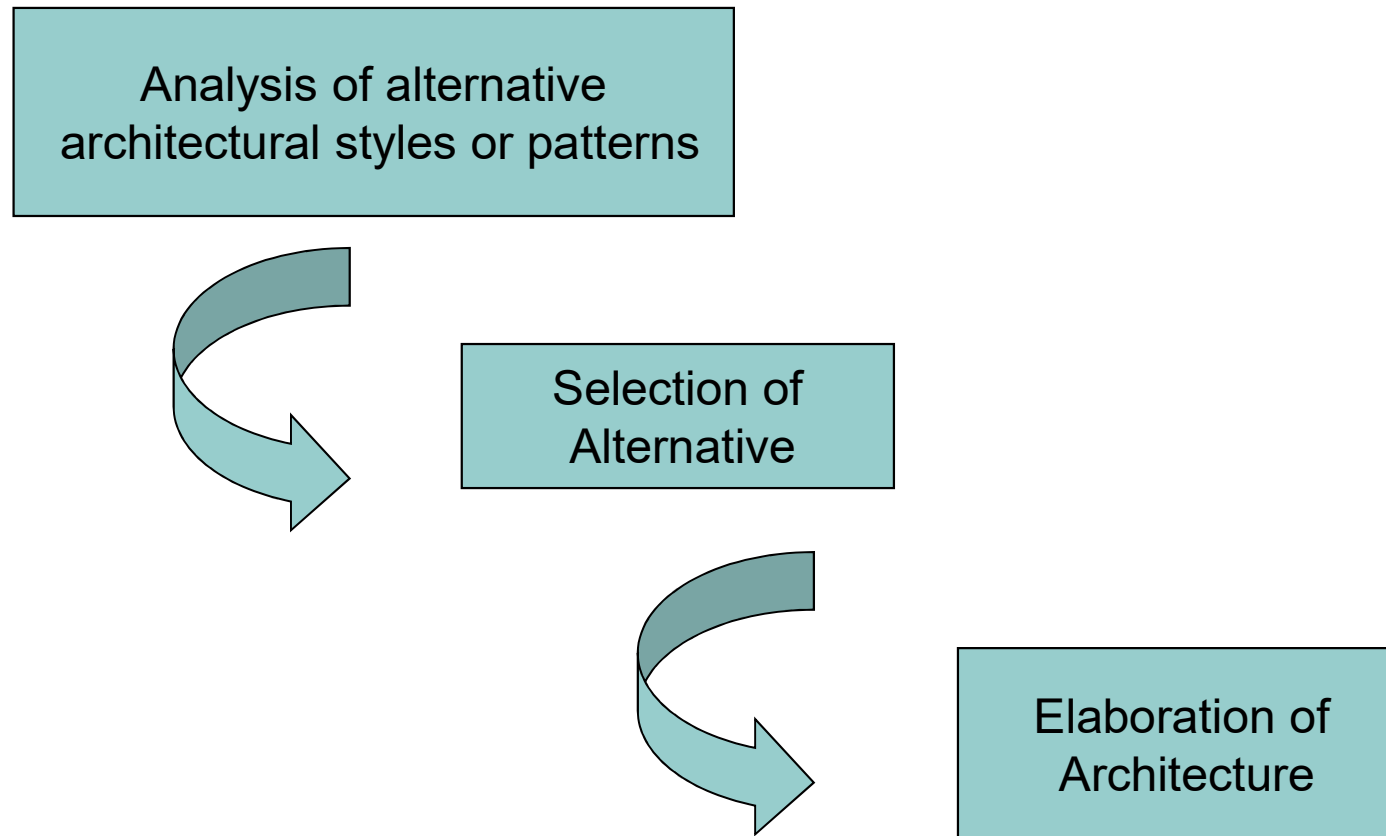
Software Design Model



Steps



Steps (contd.)



Software Architecture

- What Is Architecture?
 - The software architecture of a program or computing system is the **structure or structures of the system**, which comprise software **components**, the **externally visible properties** of those components, and the **relationships** among them.
 - The architecture is **not the operational software**.

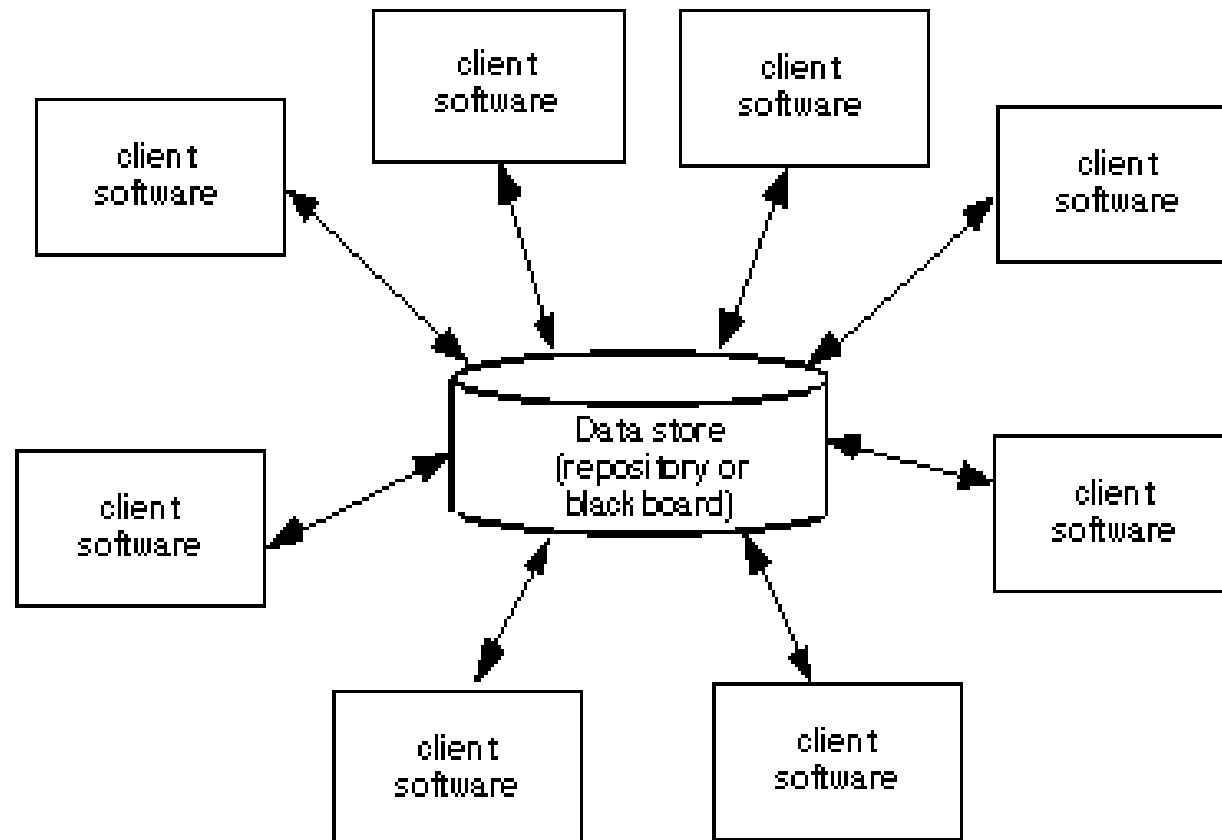
Architectural Styles

Each style describes a system category that encompasses:

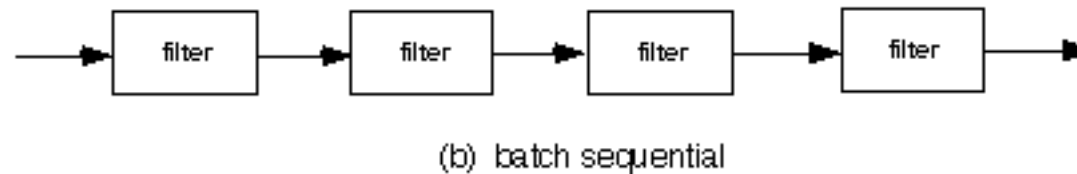
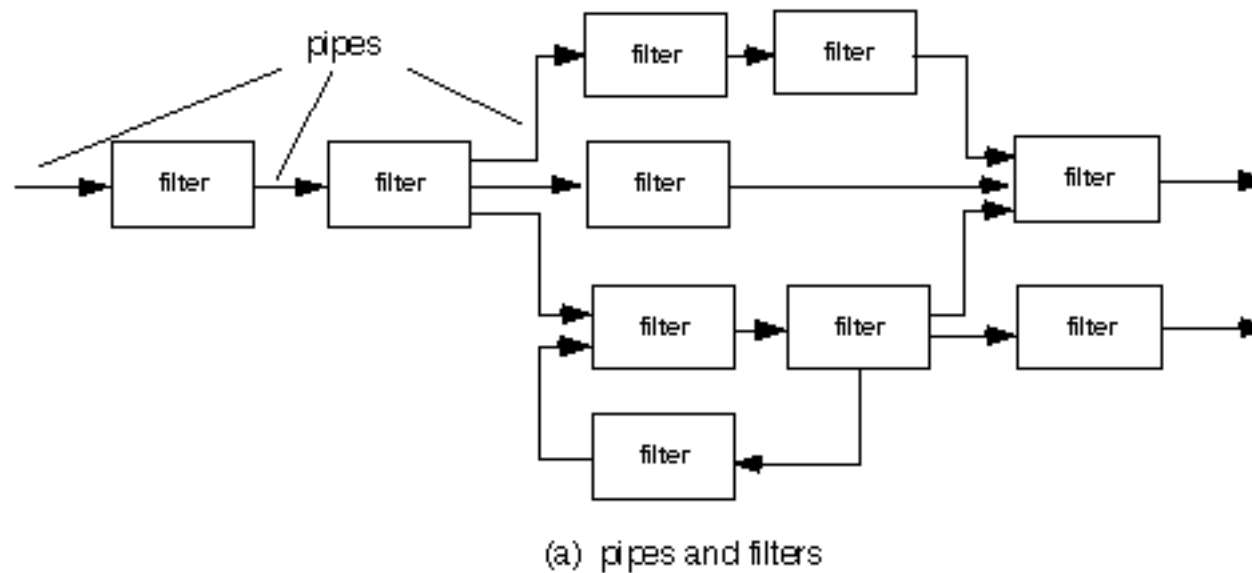
- (1) a set of **components** (e.g., a database, computational modules) that perform a function required by a system,
- (2) a set of **connectors** that enable “communication, coordination and cooperation” among components,
- (3) **constraints** that define how components can be integrated to form the system, and
- (4) **semantic models** that enable a designer to understand the overall properties

- Data-centered architectures
- Data flow architectures
- Call and return architectures
- Object-oriented architectures
- Layered architectures

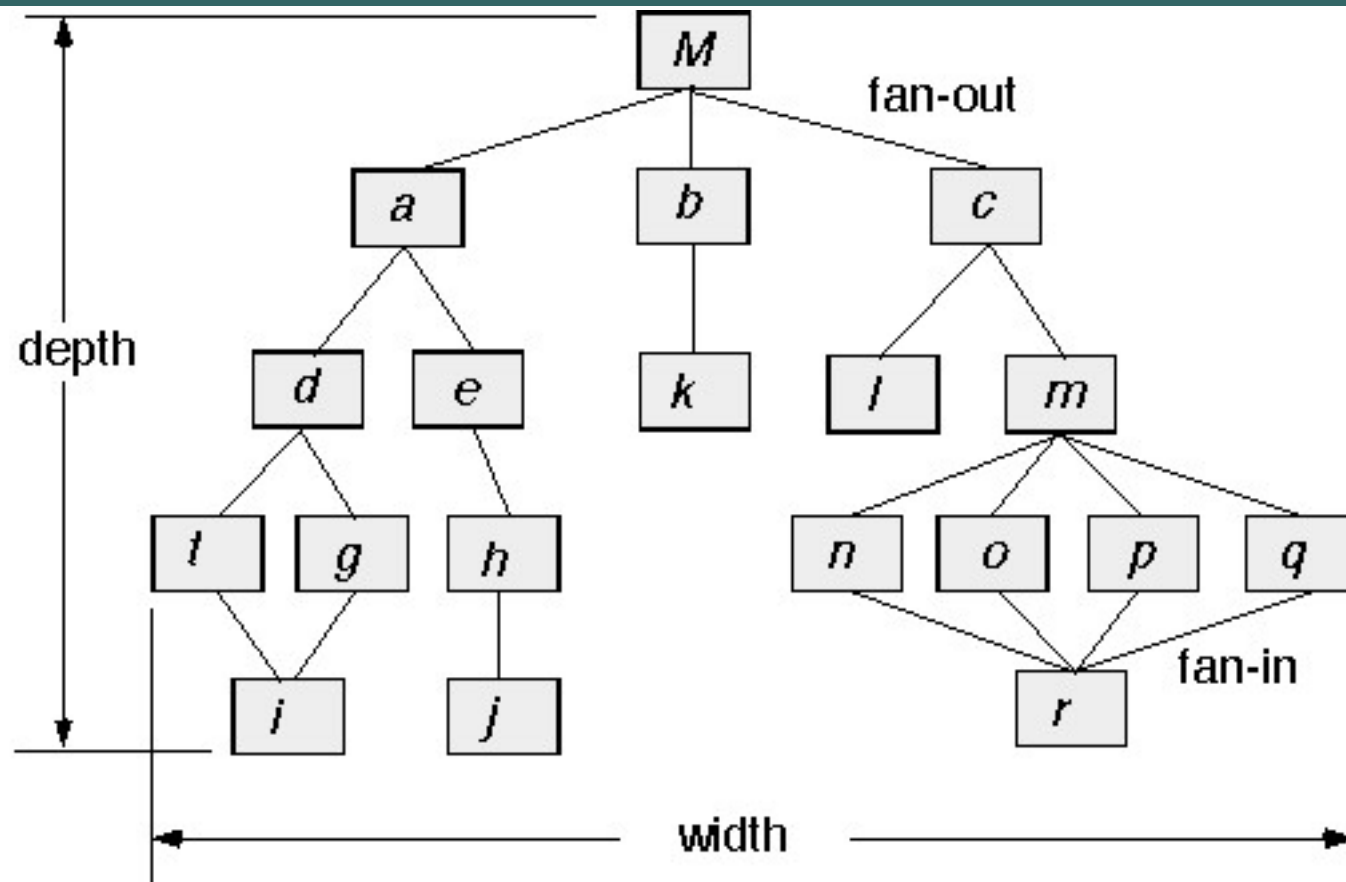
Data-Centered Architecture



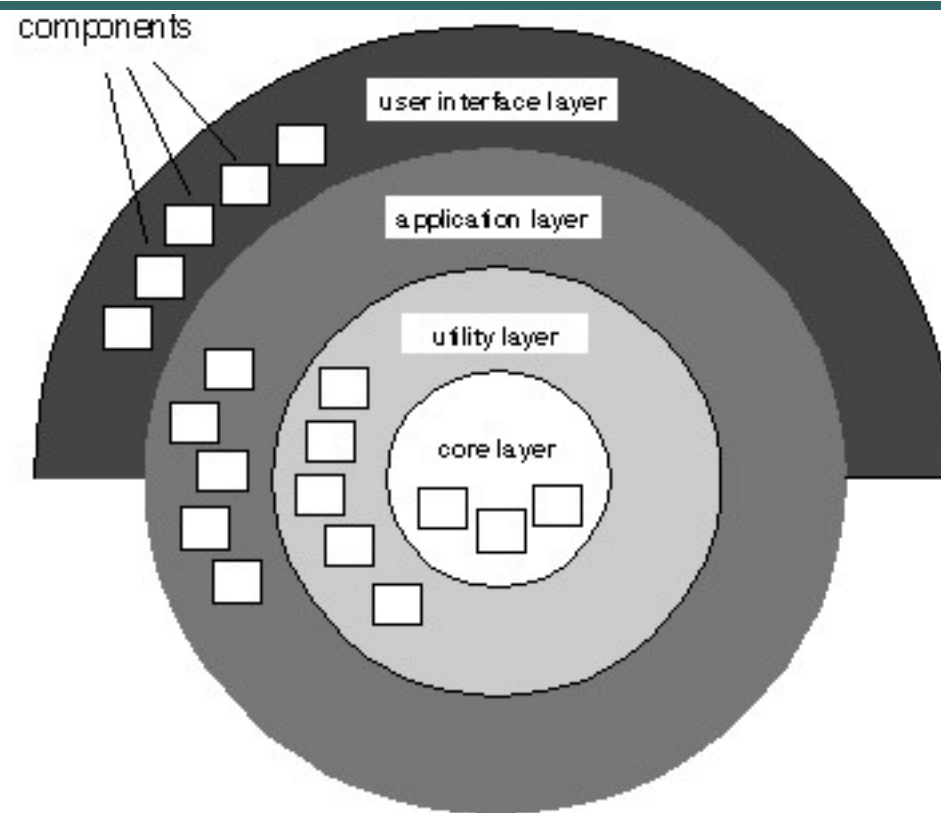
Data Flow Architecture



Call and Return Architecture



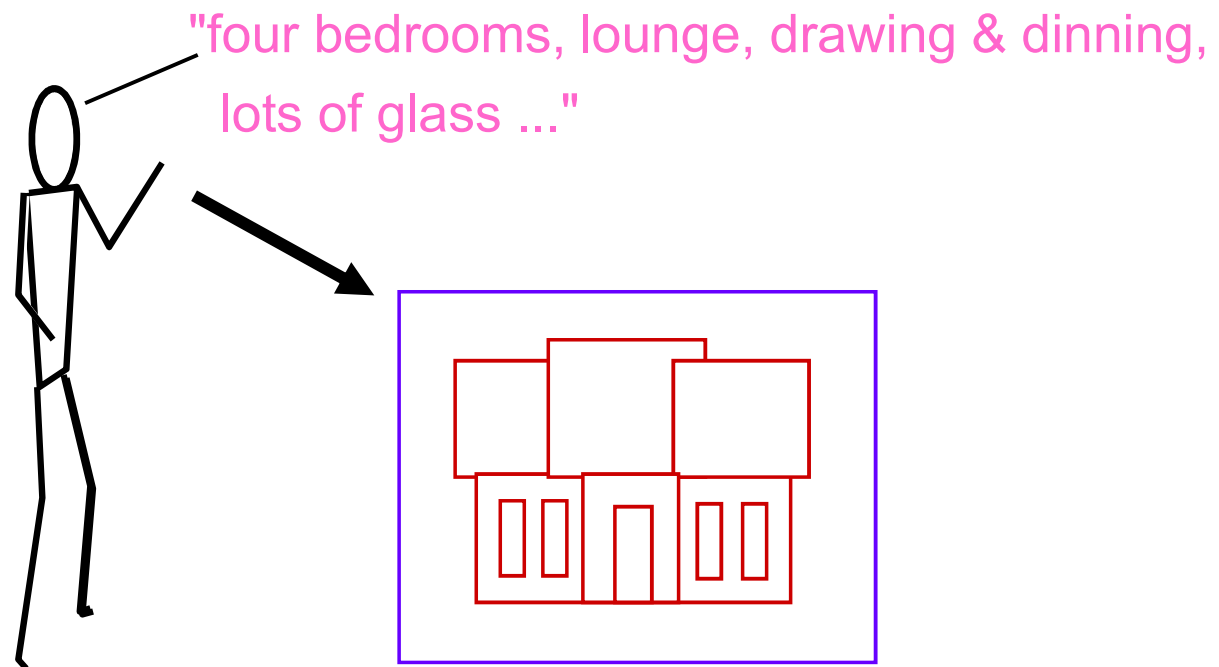
Layered Architecture



Mapping Requirements into a Software Architecture

An Architectural Design Method

customer requirements



architectural design

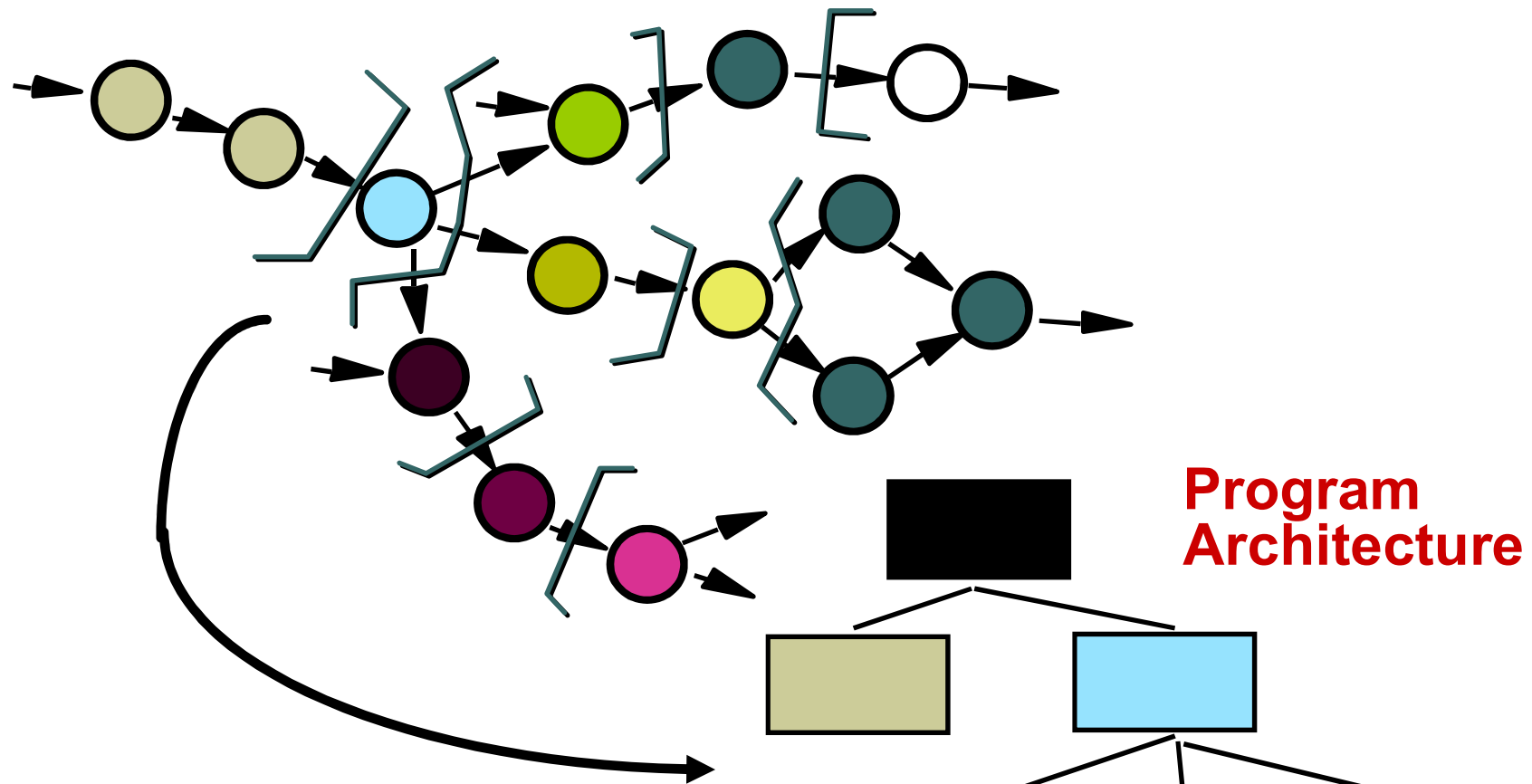
Mapping Requirements Into Software Architecture

Earlier had said certain models can be mapped directing into an architectural design

- methods do not exist for all architectural styles
- Will look at 1 approach for the *call & return* architecture sometimes called *structured design* - origins in *top-down design* [WIR71], structured programming [DAH72],

- **SD is a data-flow oriented design method**
 - **Provides a method to go from a DFD to program structure**

Deriving Program Architecture



Structured Design

- **objective:** develop a modular program structure and represent control relationships between modules
- **approach:**
 - the DFD is mapped into a program architecture
 - the PSPEC and STD are used to indicate the content of each module
- **notation:** structure chart

Structured Design

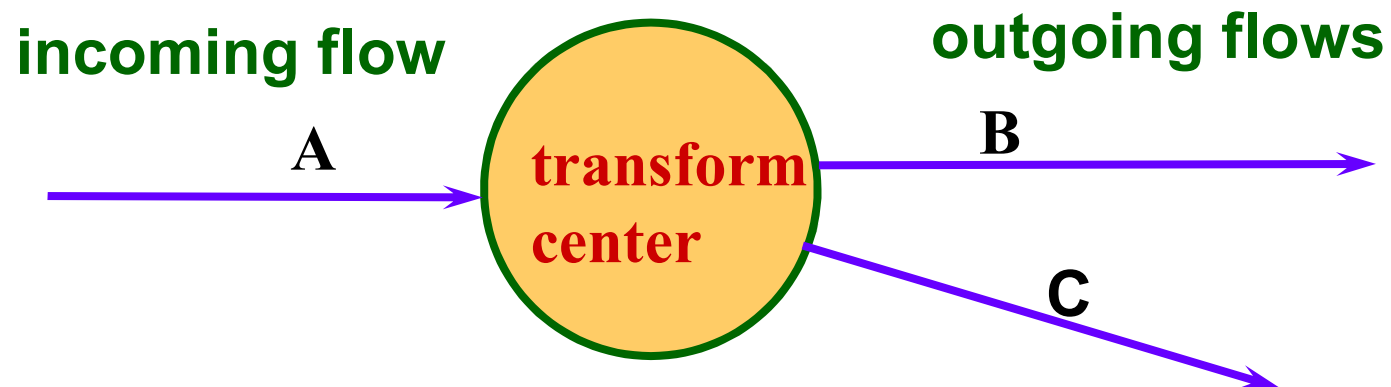
- It provides a convenient transition from a data flow diagram to software Architecture

1. The type of information flow is established
2. Flow boundaries are indicated
3. The DFD is mapped into program structure
4. Control hierarchy is defined
5. Resultant structure is refined using design measures and heuristics
6. The architectural description is refined and elaborated

Types of Information Flow

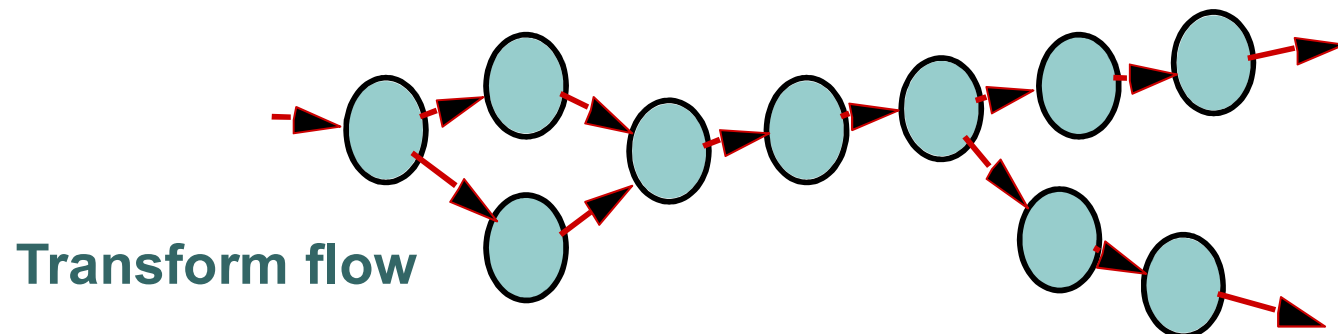
- There are 2 different types of information flow that have different treatments
 - *Transform flow* - Overall data flows in sequential manner and follows one, or only a few, “straight line” paths. (incoming, transform, output)
 - *Transaction Flow* - Info flow has a single transaction node that *triggers* other data flow

Transform Flow



❑ Transform Flow

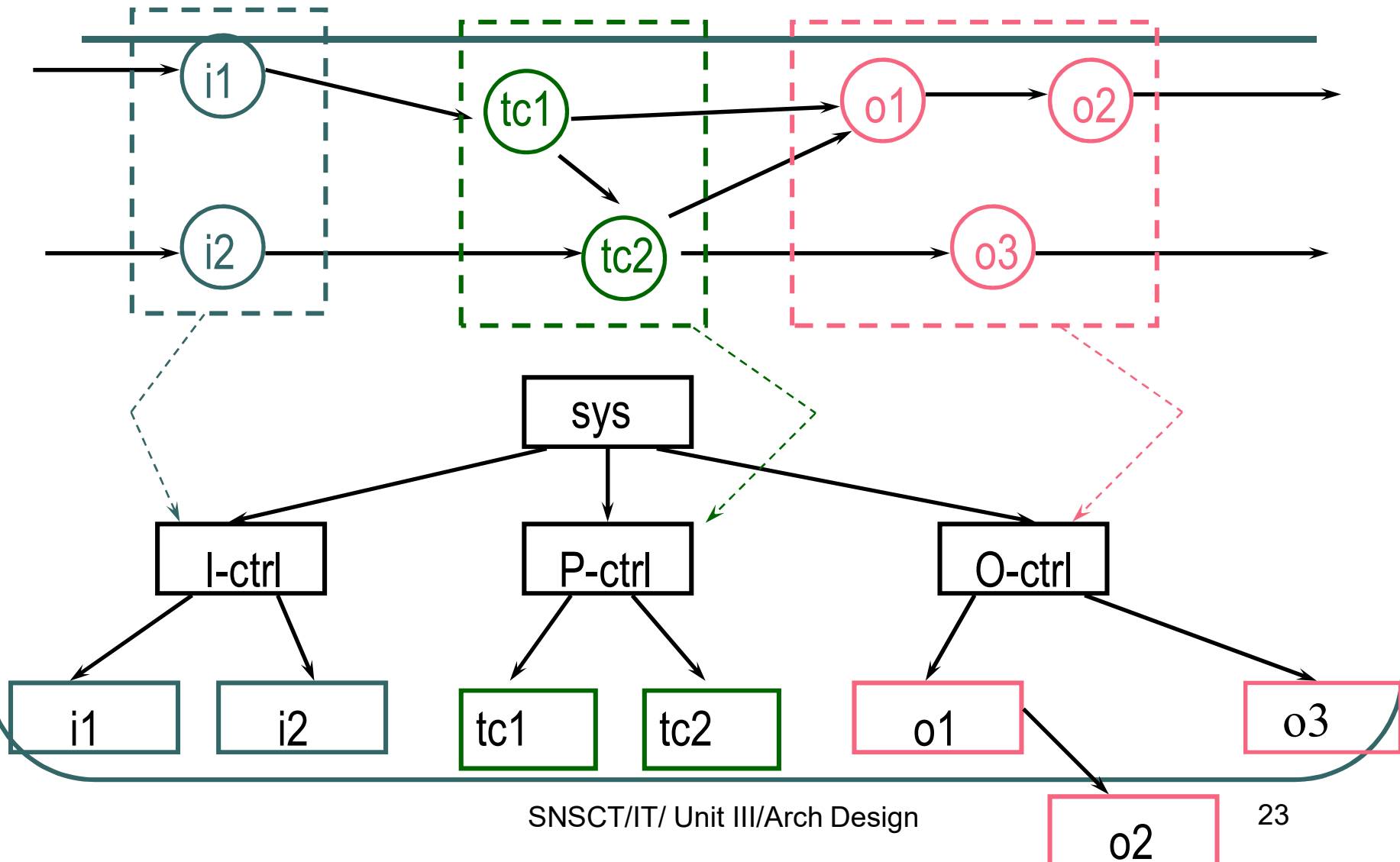
- **Incoming Flow:** The paths that transform the external data into an internal form
- **Transform Center:** The incoming data are passed through a transform center and begin to move along paths that lead it out of the software
- **Outgoing Flow:** The paths that move the data out of the software



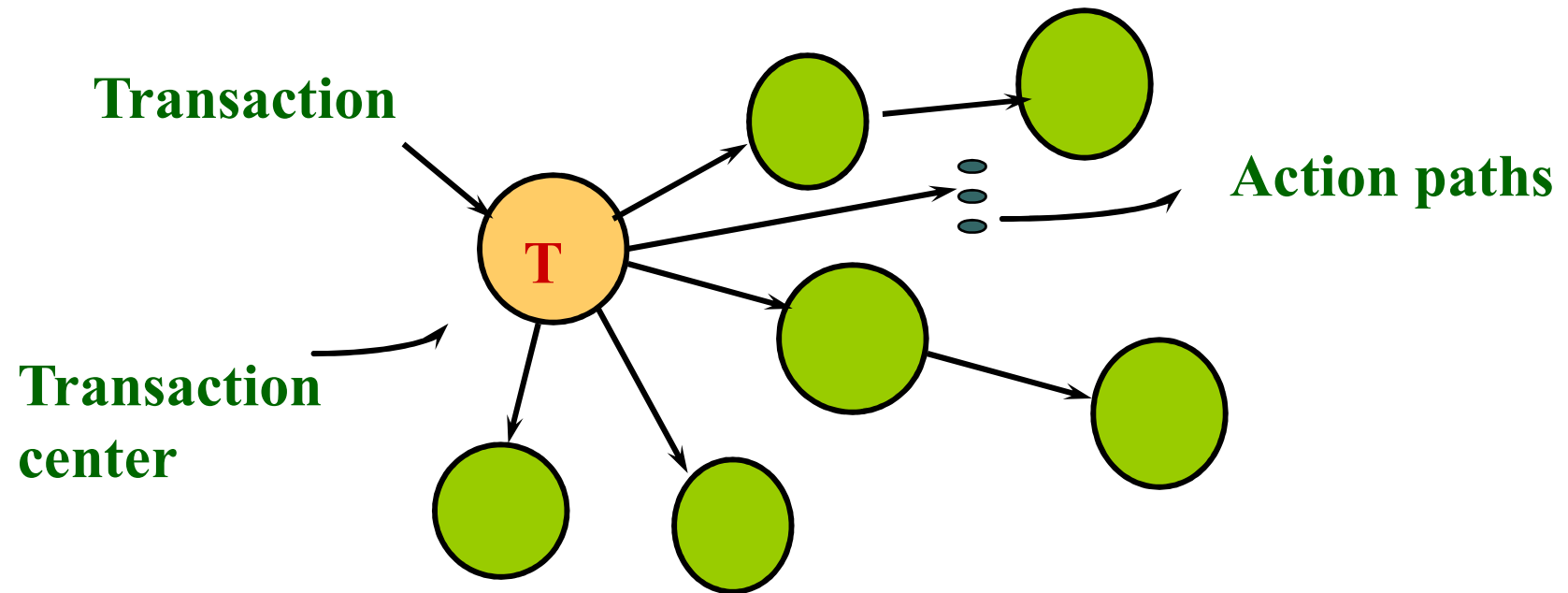
Transform Flow Characteristics

- the system has a **single, coherent objective**
- **transformation center executes** algorithms, data transformation, database manipulation, ...
- input-driven processes filter, check and translate external data flows
- output-driven processes format results for presentation to the environment (user)
- multiple paths to obtain input

Transform Analysis: mapping heuristic

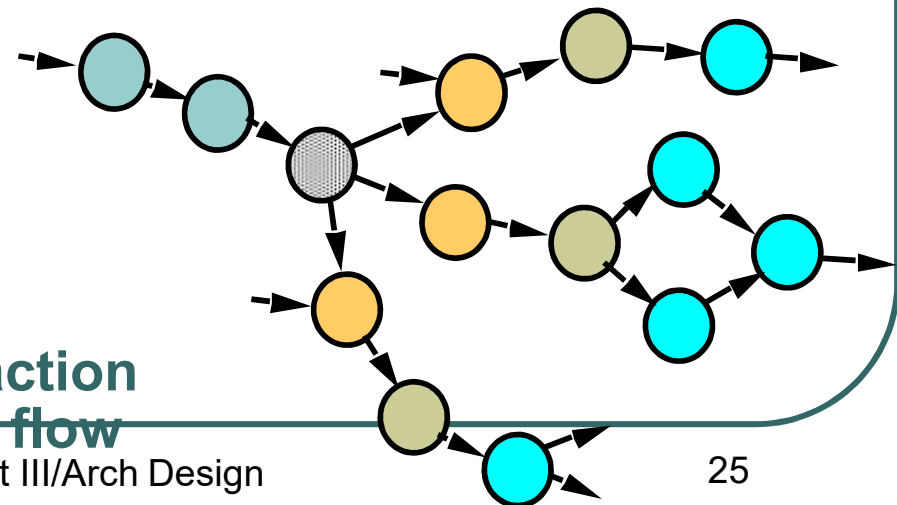


Transaction Flow



Transaction Flow

- Information flow is often characterized by a single data item, called a **transaction** that triggers other data flow along one of many paths
- Action Paths** : The transaction is evaluated and based on its value flow along one of many action paths
- Transaction center** : The hub of info flow from which many action paths originate

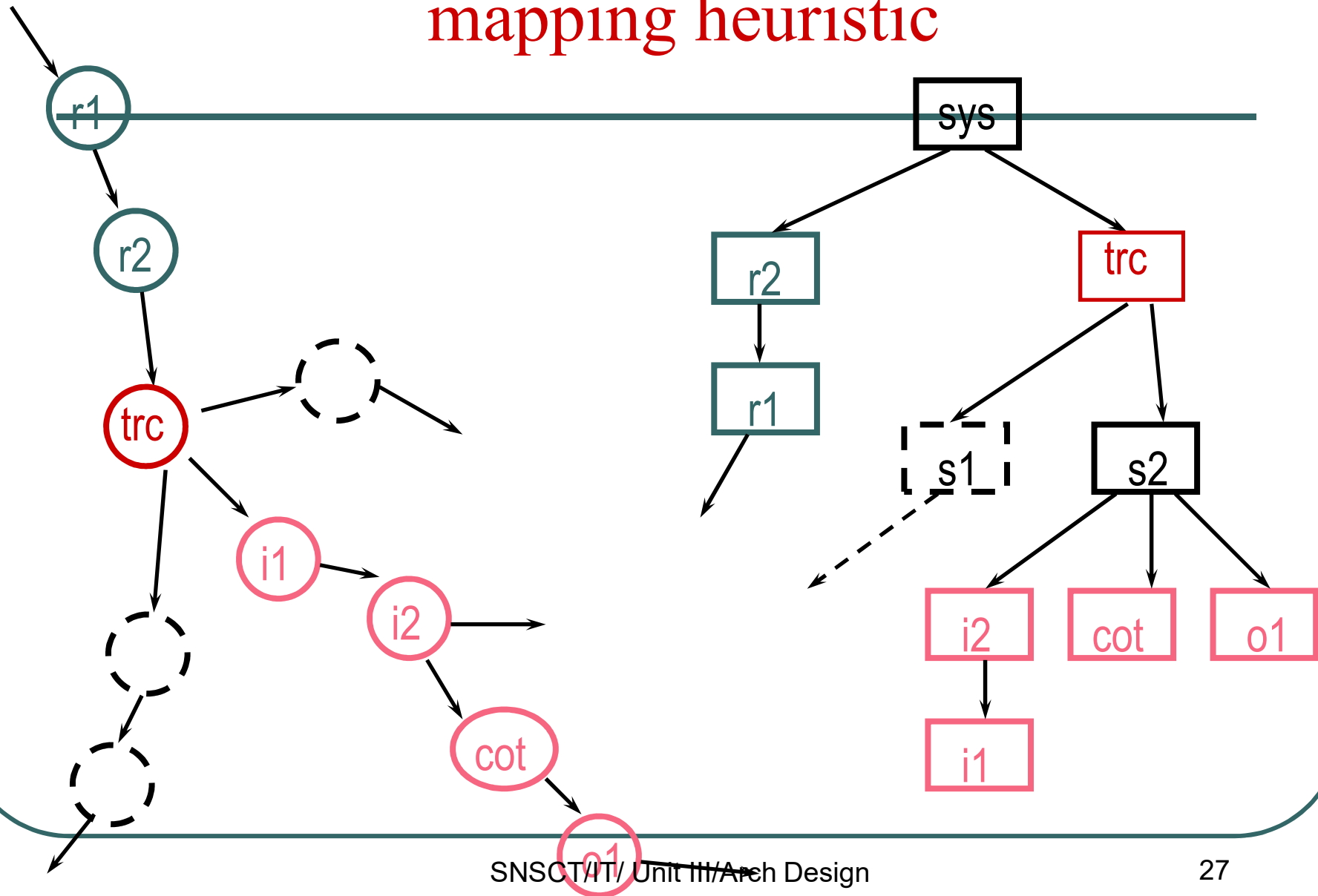


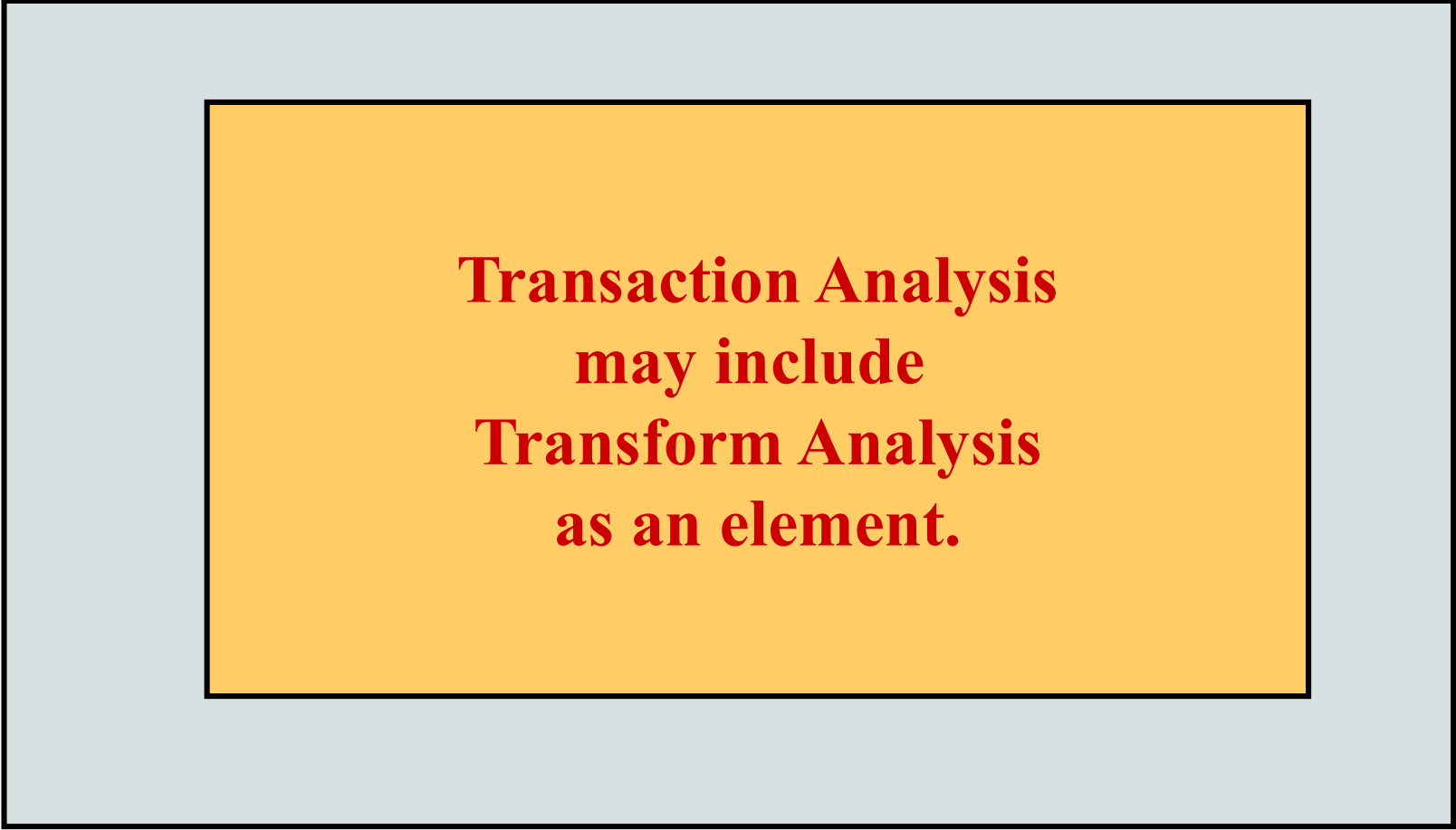
Transaction
flow

Transaction Flow Characteristics

- single line of reception processes
- **transaction**: a single data item that includes all necessary information for execution
- **transaction center** evaluates transaction & initializes correct action-path => distribution
- action-paths implement (clearly) **different types of functionality** => execution
- an action-path could be a complete (sub-)system with transform flow characteristics

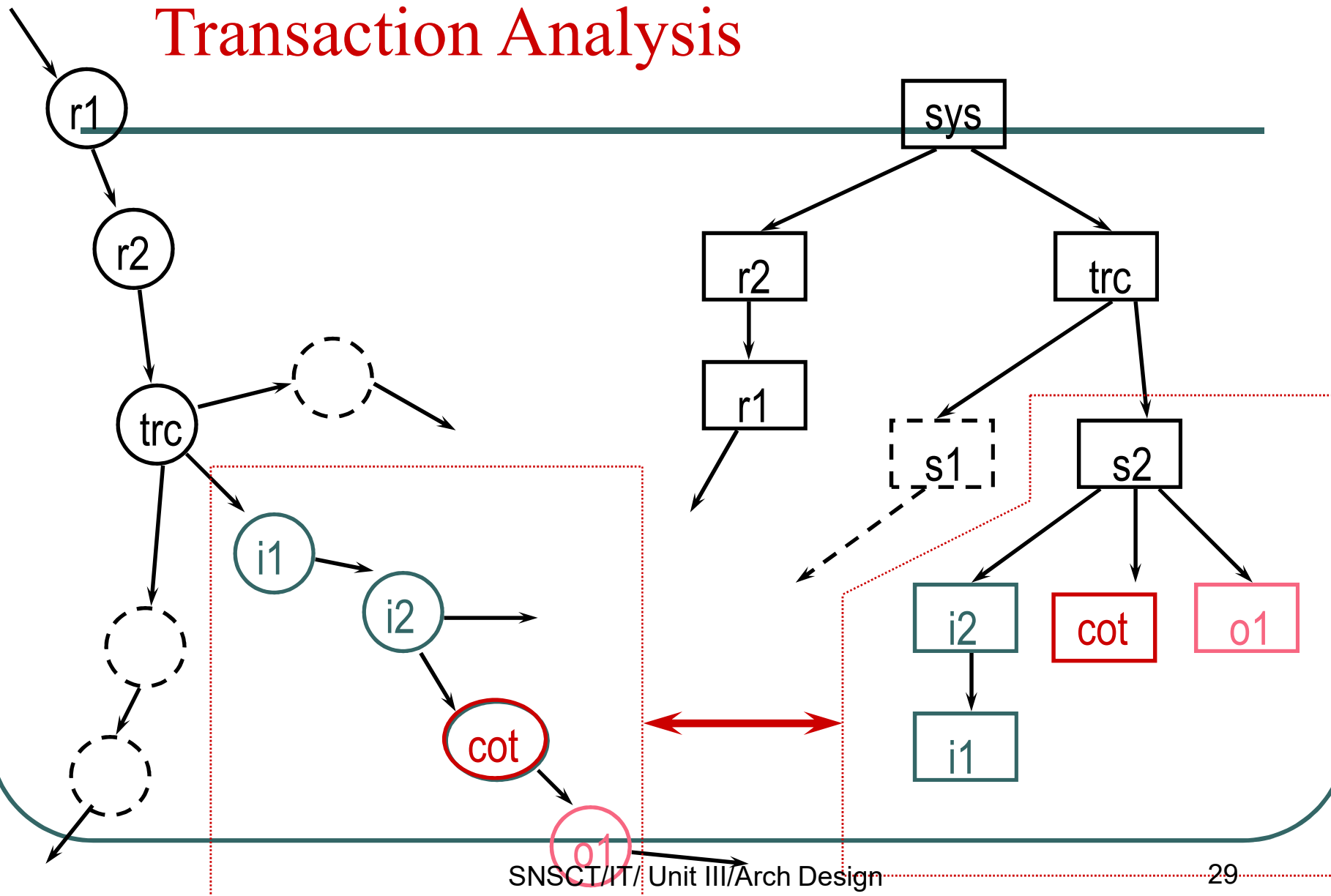
Transaction Analysis: mapping heuristic





**Transaction Analysis
may include
Transform Analysis
as an element.**

Transform Analysis as an element of Transaction Analysis



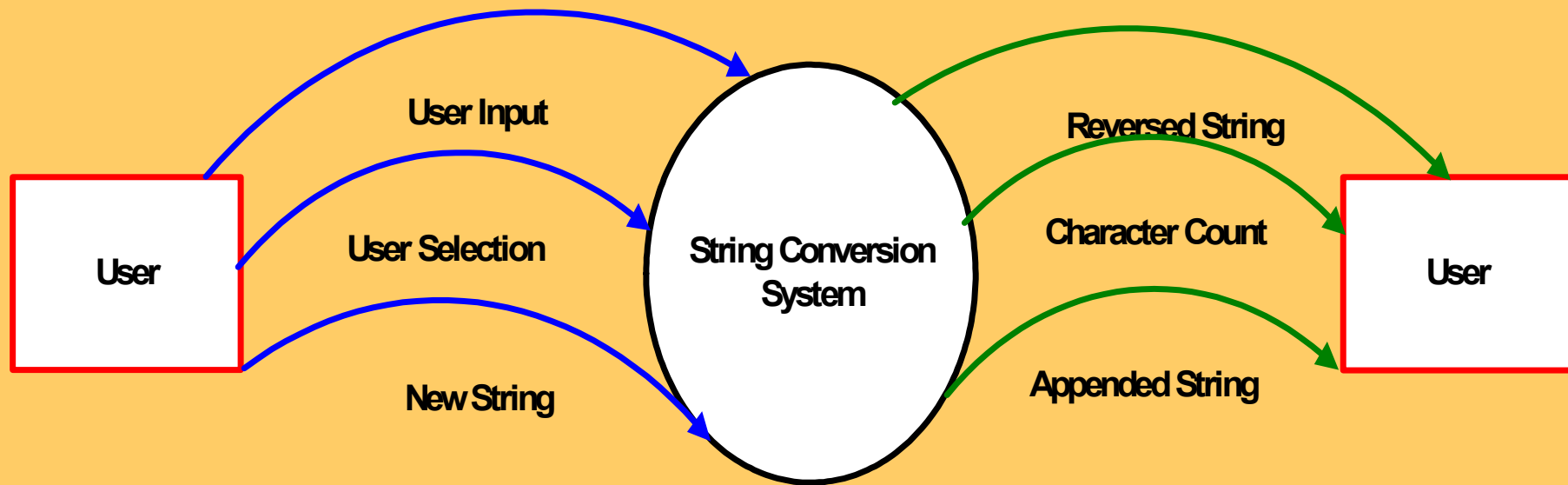
Transform Mapping (cont)

- **Design steps**

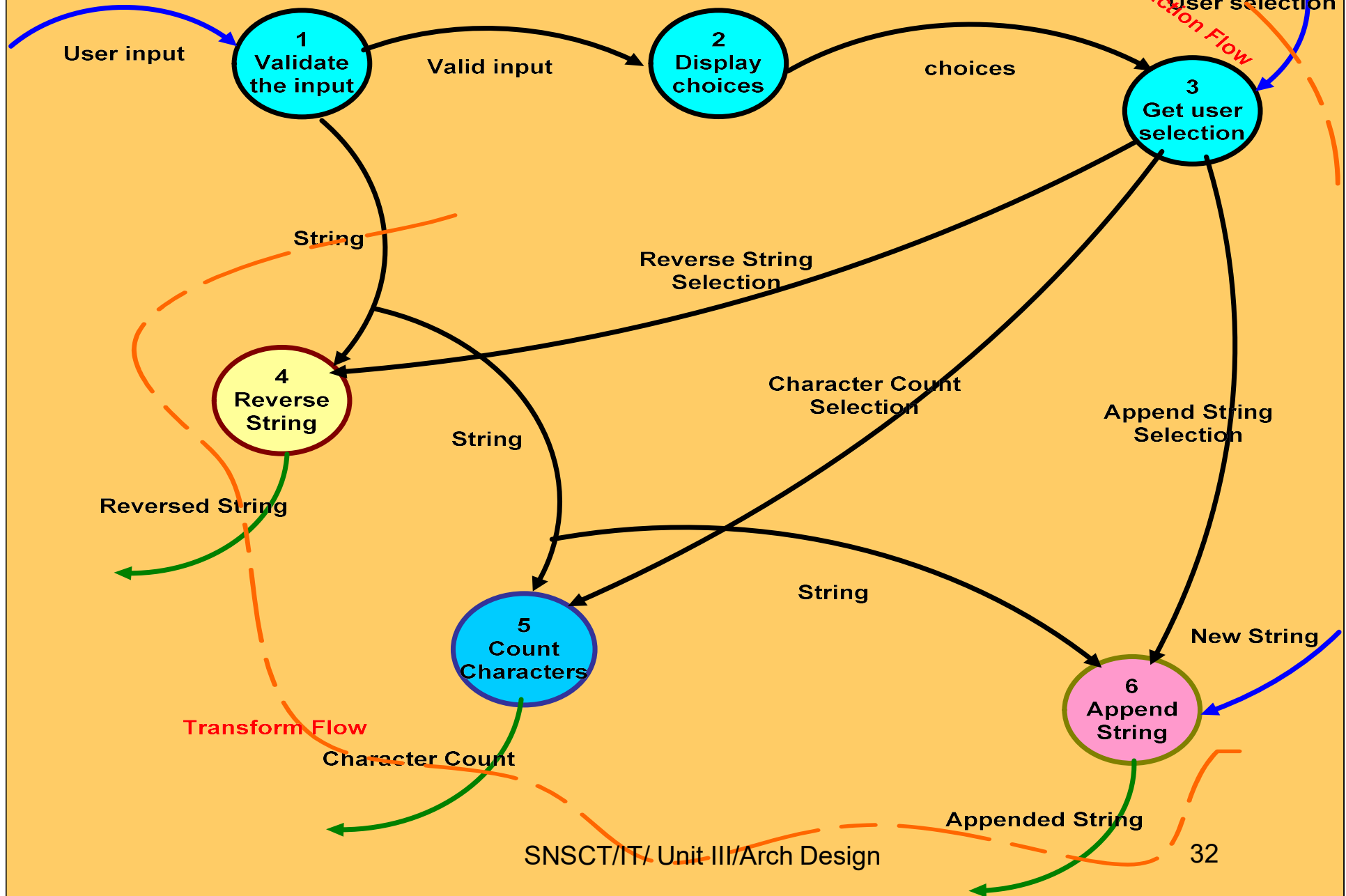
- **Step 1.** Review the fundamental system model.
- **Step 2.** Review and refine data flow diagrams for the software.
- **Step 3.** Determine whether DFD has transform or transaction flow characteristics.

- in general---transform flow
- special case---transaction flow

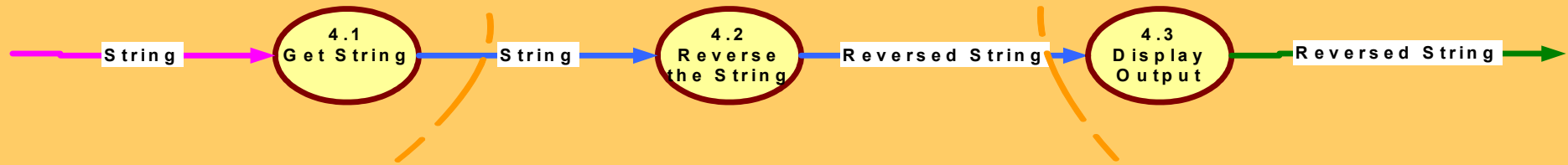
Context Level DFD



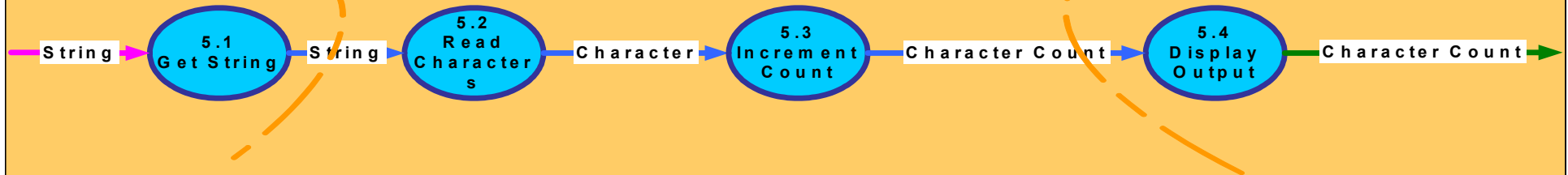
Level 1 DFD



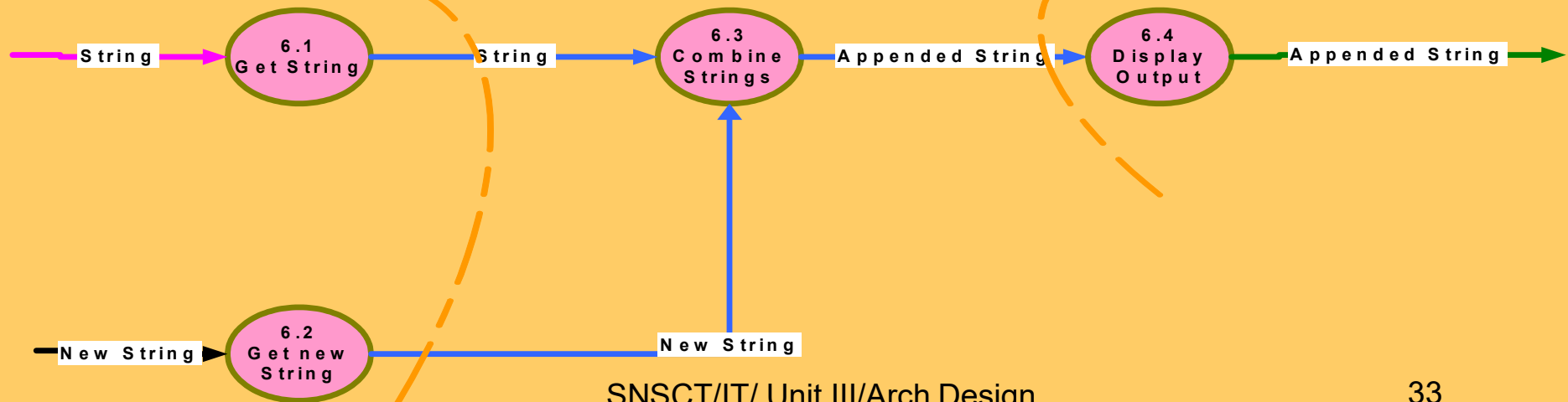
DFD Level-2 For REVERSE STRING - <Process # 4>



DFD Level-2 For Count Characters - <Process # 5>



DFD Level-2 For Append STRING - <Process # 6>

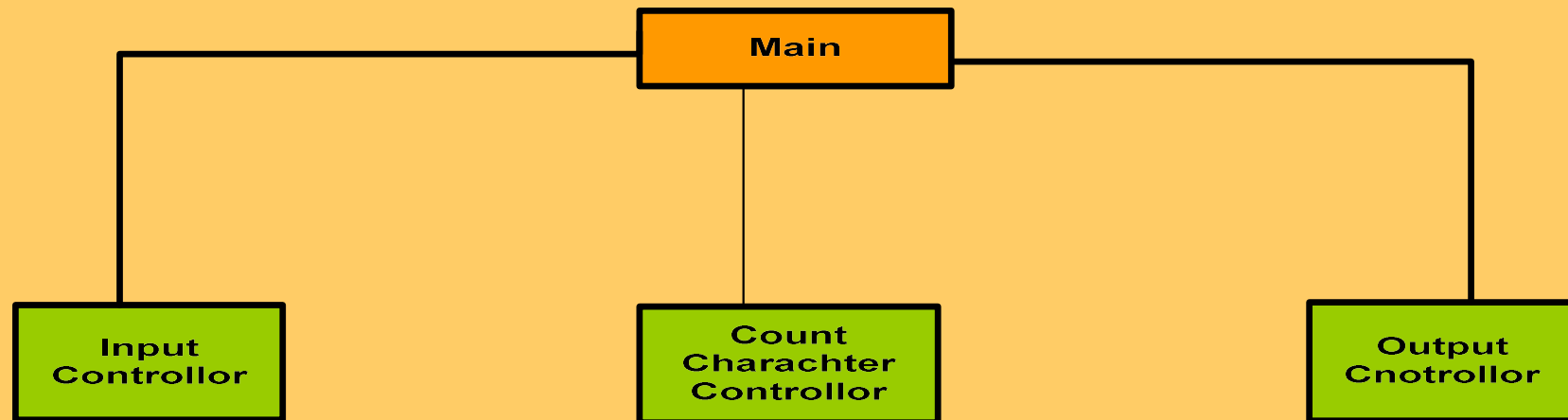


Transform Mapping (cont)

- **Step 4. Isolate the transform center by specifying incoming and outgoing flow boundaries**
 - different designers may select slightly differently
 - transform center can contain more than one bubble.
- **Step 5. Perform “first-level factoring”**
 - program structure represent a top-down distribution control.
 - factoring results in a program structure(top-level, middle-level, low-level)
 - number of modules limited to minimum.



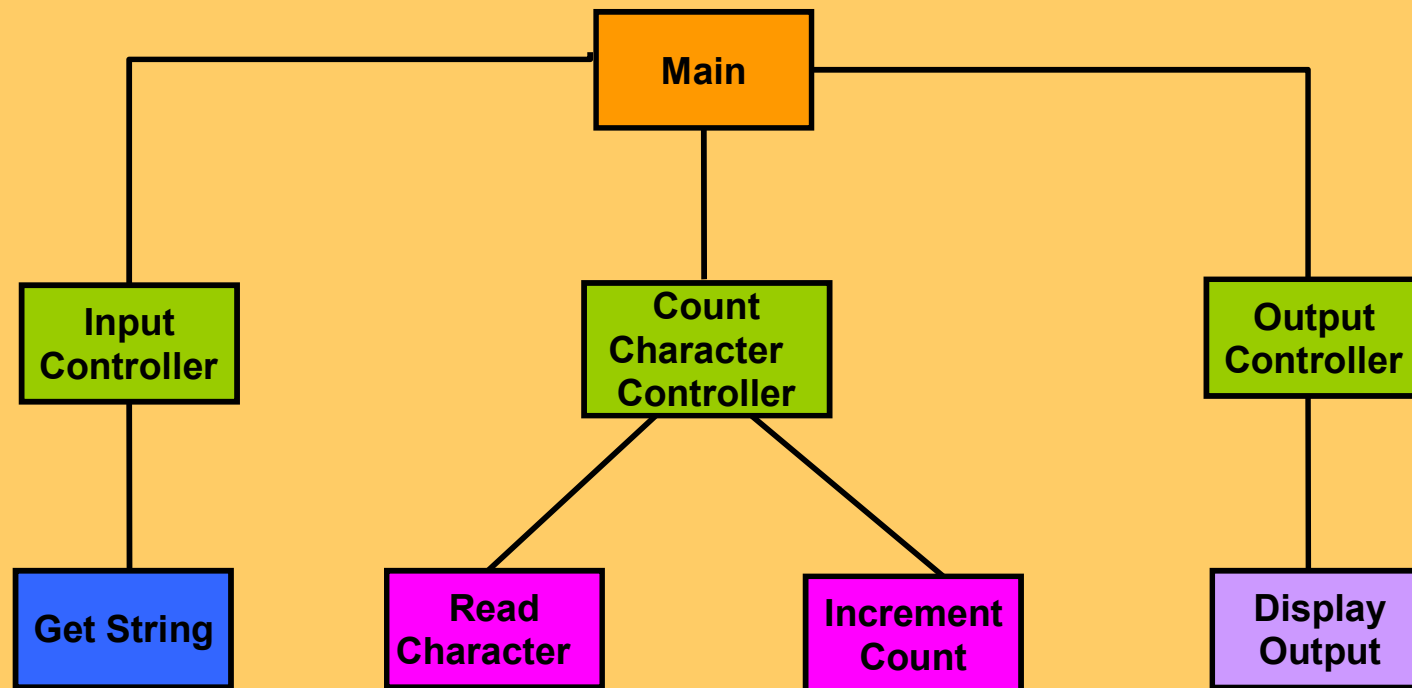
First Level Factoring



Transform Mapping (cont)

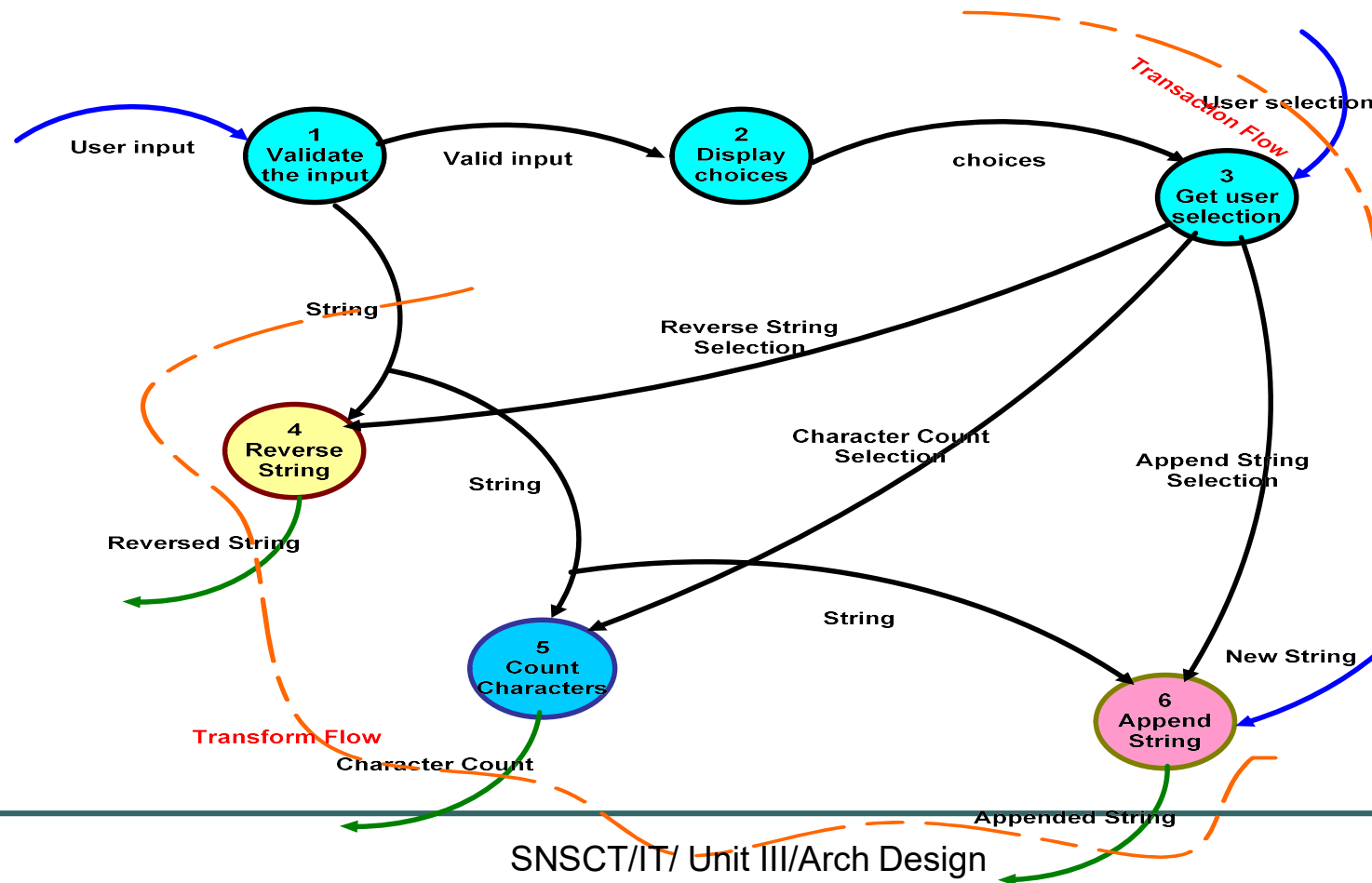
- **Step 6. Perform “second-level factoring”**
 - mapping individual transforms(bubbles) to appropriate modules.
 - factoring accomplished by moving outwards from transform center boundary.
- **Step 7. Refine the first iteration program structure using design heuristics for improved software quality.**

Second Level Factoring



Transaction Mapping

A single data item triggers one or more information flows



Transaction Mapping Design

- **Step 1.** Review the fundamental system model.
- **Step 2.** Review and refine DFD for the software
- **Step 3.** Determine whether the DFD has transform or transaction flow characteristics
- **Step 4.** Identify the transaction center and flow characteristics along each of the action paths
 - isolate incoming path and all action paths
 - each action path evaluated for its flow characteristic.

Transaction Mapping (cont)

- **Step 5. Map the DFD in a program structure amenable to transaction processing**
 - incoming branch
 - bubbles along this path map to modules
 - dispatch branch
 - dispatcher module controls all subordinate action modules
 - each action path mapped to corresponding structure

Transaction Mapping

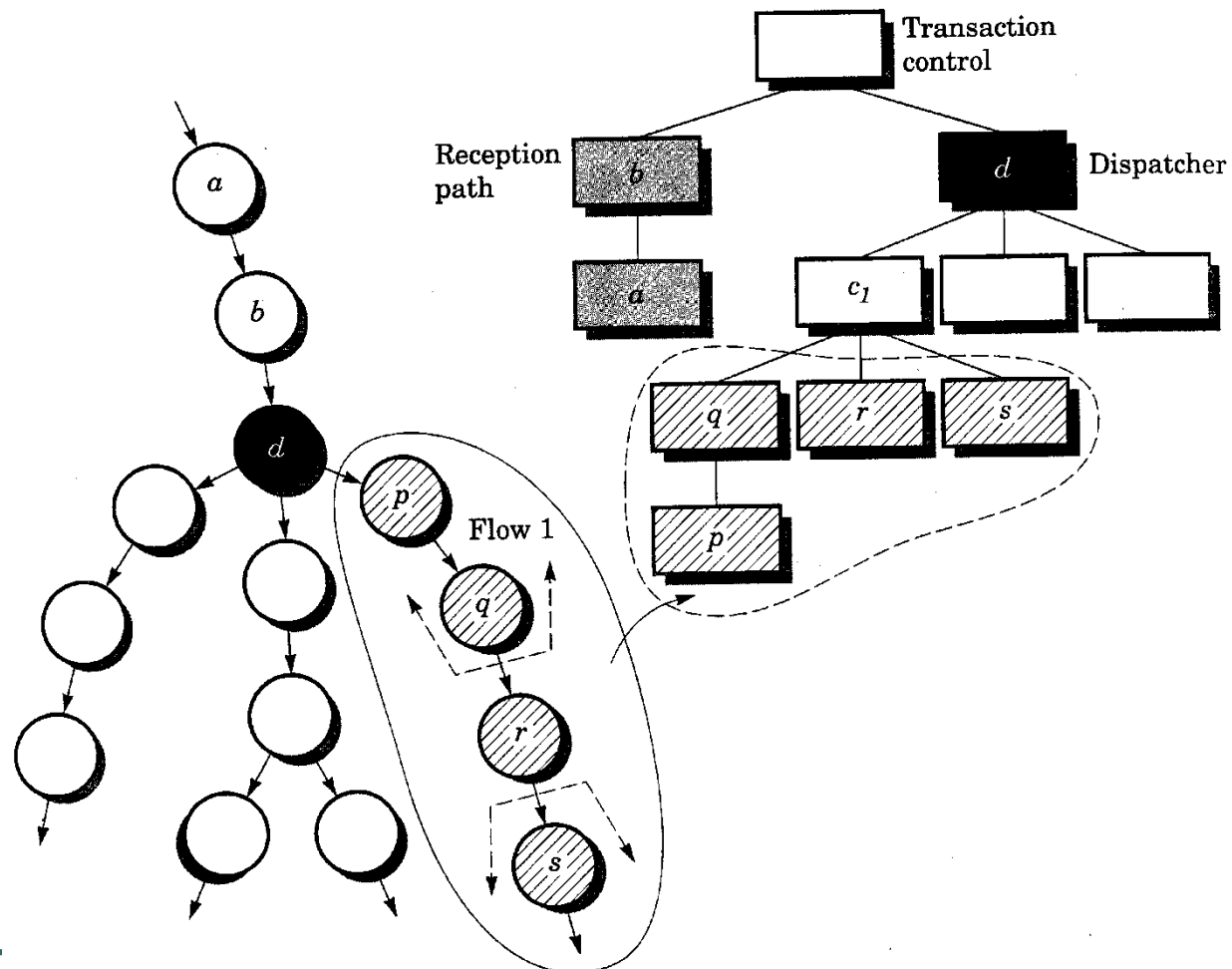
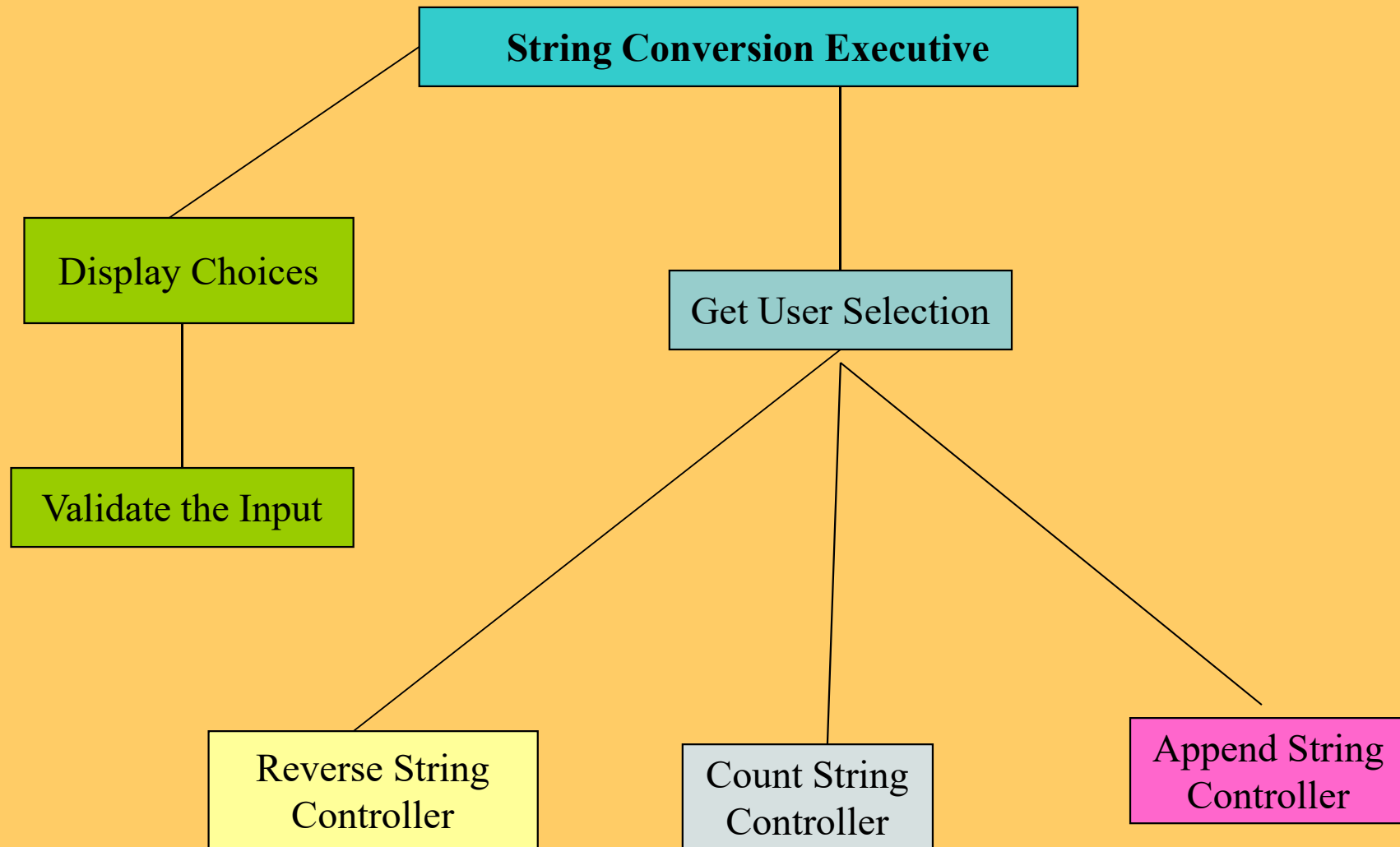
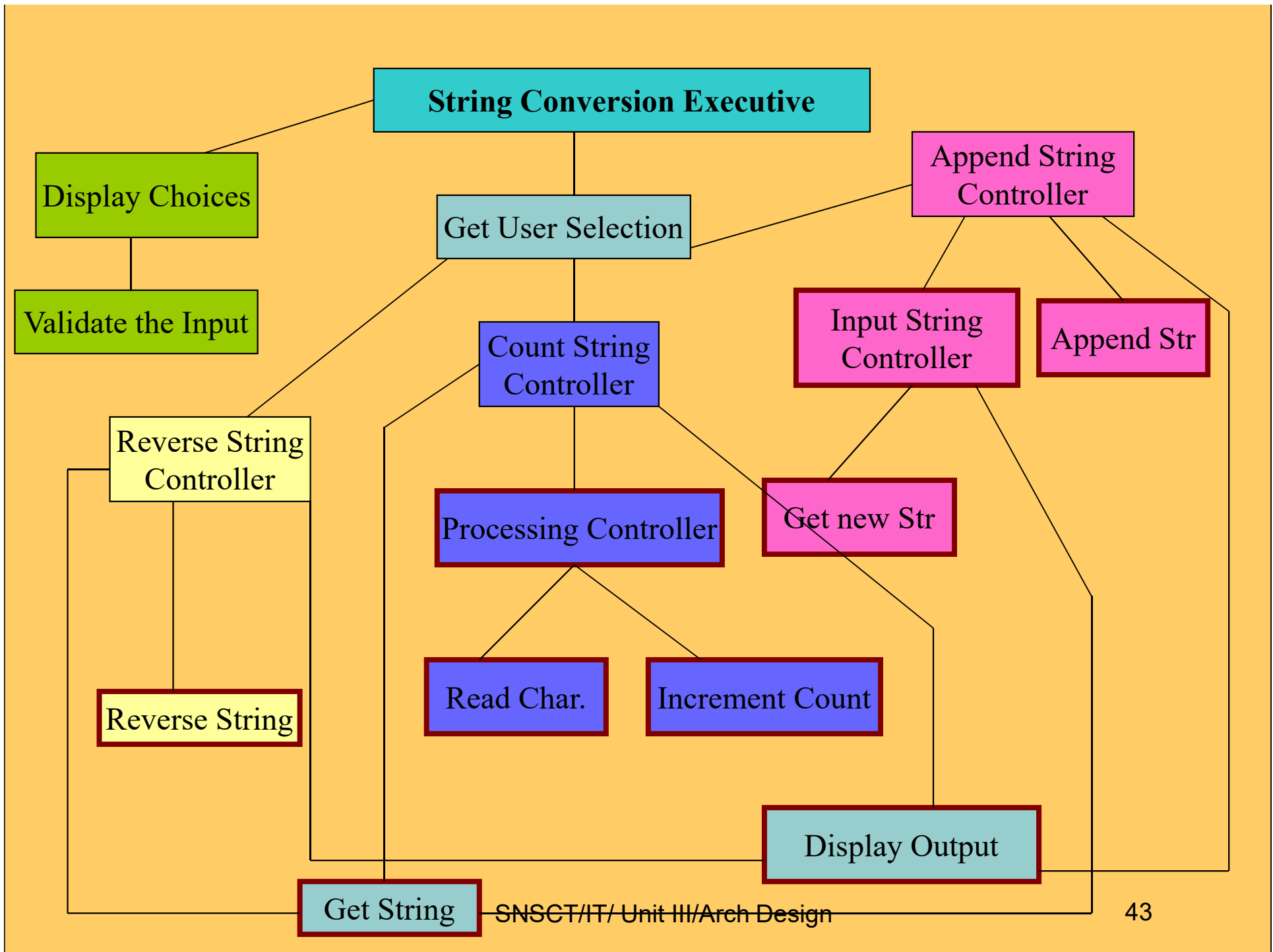


FIGURE 14.12. Transaction mapping SNSCT/IT/ Unit III/Arch Design

First Level Factoring





Transaction Mapping (cont)

- **Step 6.** Factor and refine the transaction structure and the structure of each action path
- **Step 7.** Refine the first iteration program structure using design heuristics for improved software quality

Design Post processing

- A processing narrative must be developed for each module
- An interface description is provided for each module
- Local and global data structures are defined
- All design restrictions/limitations are noted
- A design review is conducted
- “Optimization” is considered (if required and justified)

Summary

- **Data-flow oriented design:**

Natural flow from analysis

- ✓ *Find type of information flow*
- ✓ *Find flow boundaries*
- ✓ *Map DFD to program flow*
- ✓ *Factor control*
- ✓ *Refine structure*

References:

- Software Engineering - A practitioner' s Approach
by Roger S. Pressman (6th Ed)
 - 10.1, 10.3,10.6
- Software Engineering - A practitioner' s Approach
by Roger S. Pressman(5th Ed)
 - Chapter 14
 - 14.1-(14.1.1, 14.1.2), 14.3-(14.3.1), 14.5-(14.5.1, 14.5.2),
14.6-(14.6.1, 14.6.2), 14.7-(14.7.1, 14.7.2)