# Software Modeling & Analysis

## - Structured Design

Lecturer: JUNBEOM YOO
jbyoo@konkuk.ac.kr

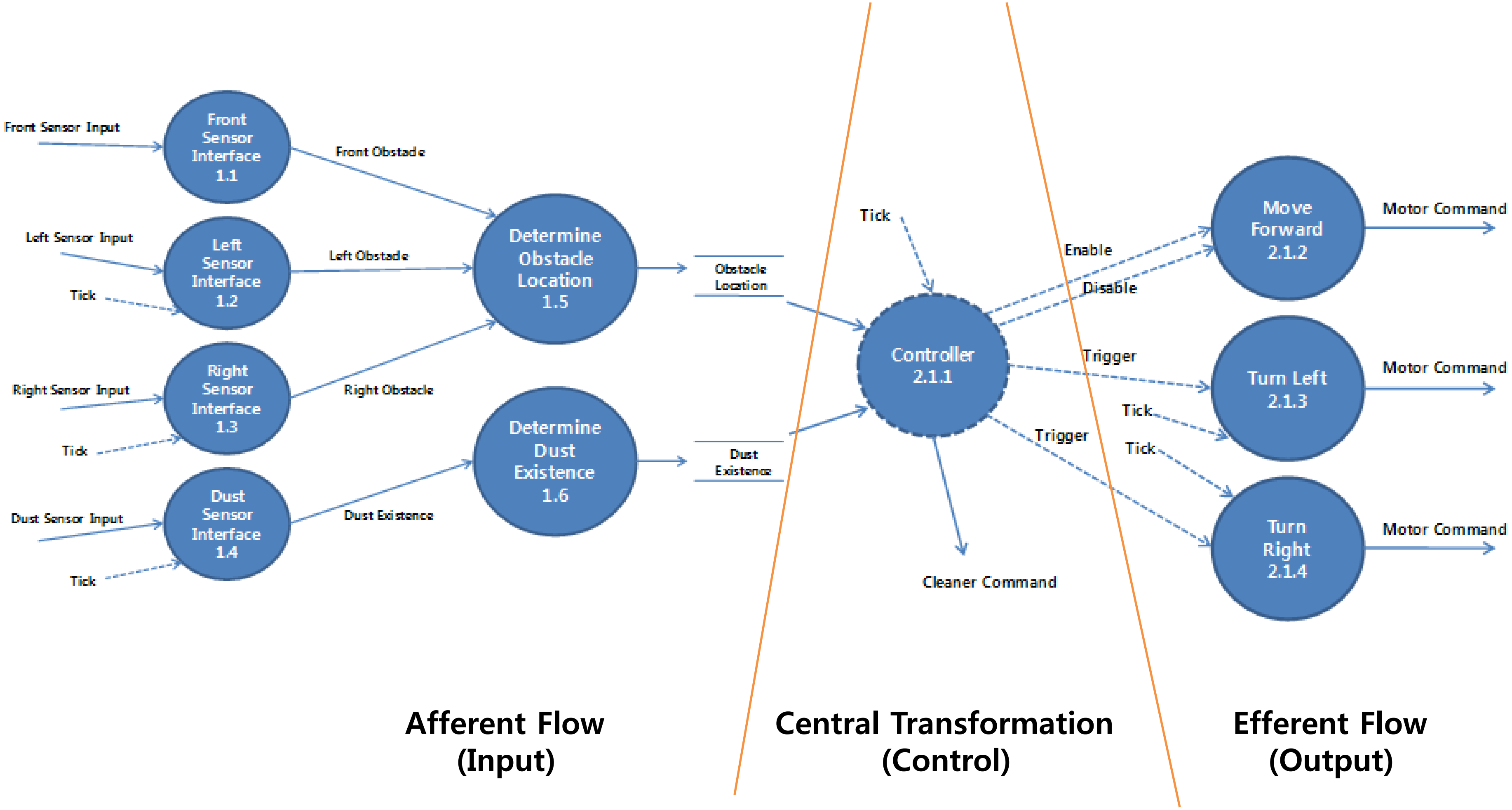
# Structure Charts

- Structured Design (SD)

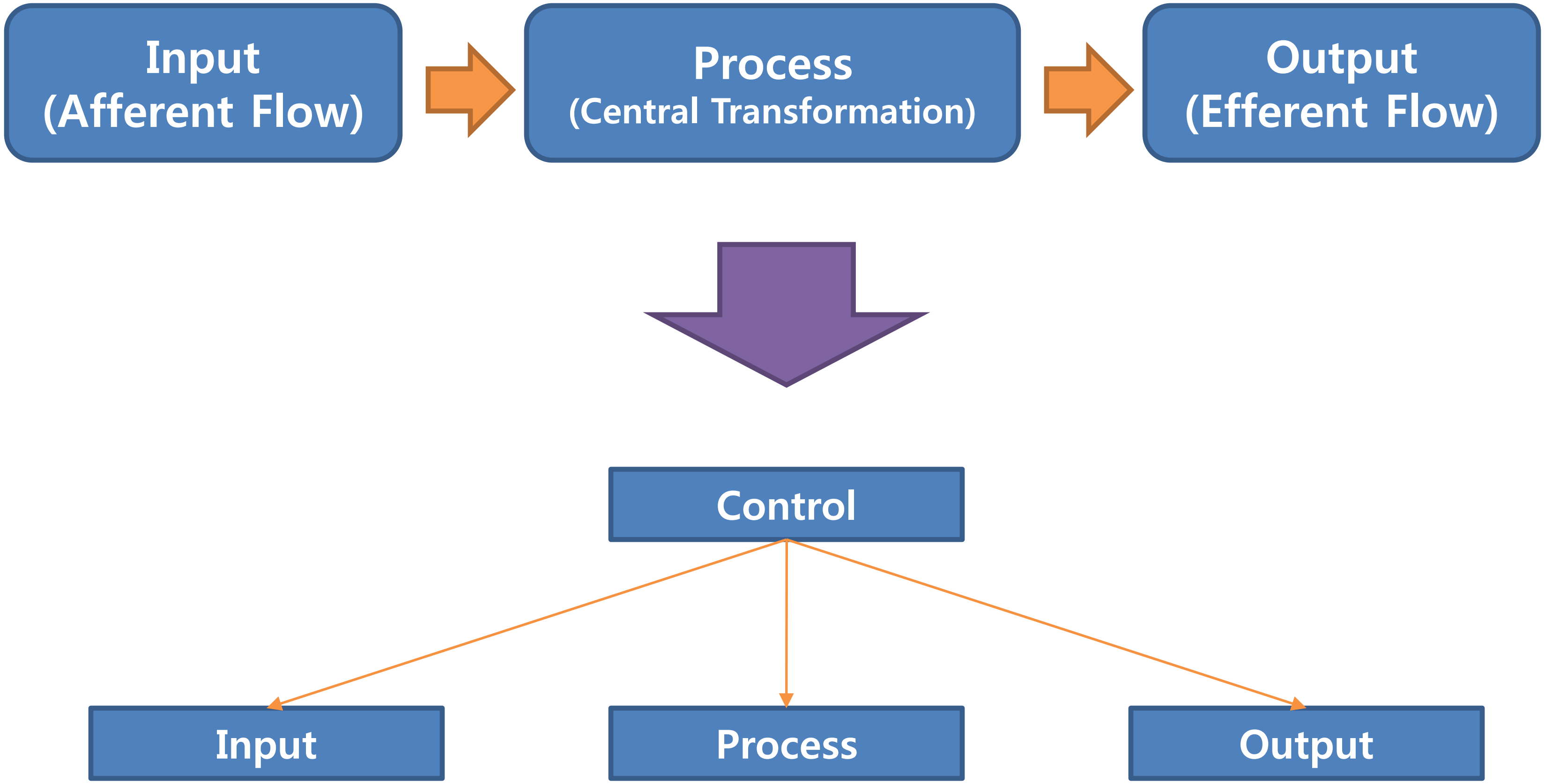
- Functional decomposition (Divide and Conquer)
  - Information hiding
  - Modularity
  - Low coupling
  - High internal cohesion

- Needs a transform analysis.

# Structured Charts – Transform Analysis



# Structured Charts – Transform Analysis



Input (Afferent Flow)

Process (Central Transformation)

Output (Efferent Flow)

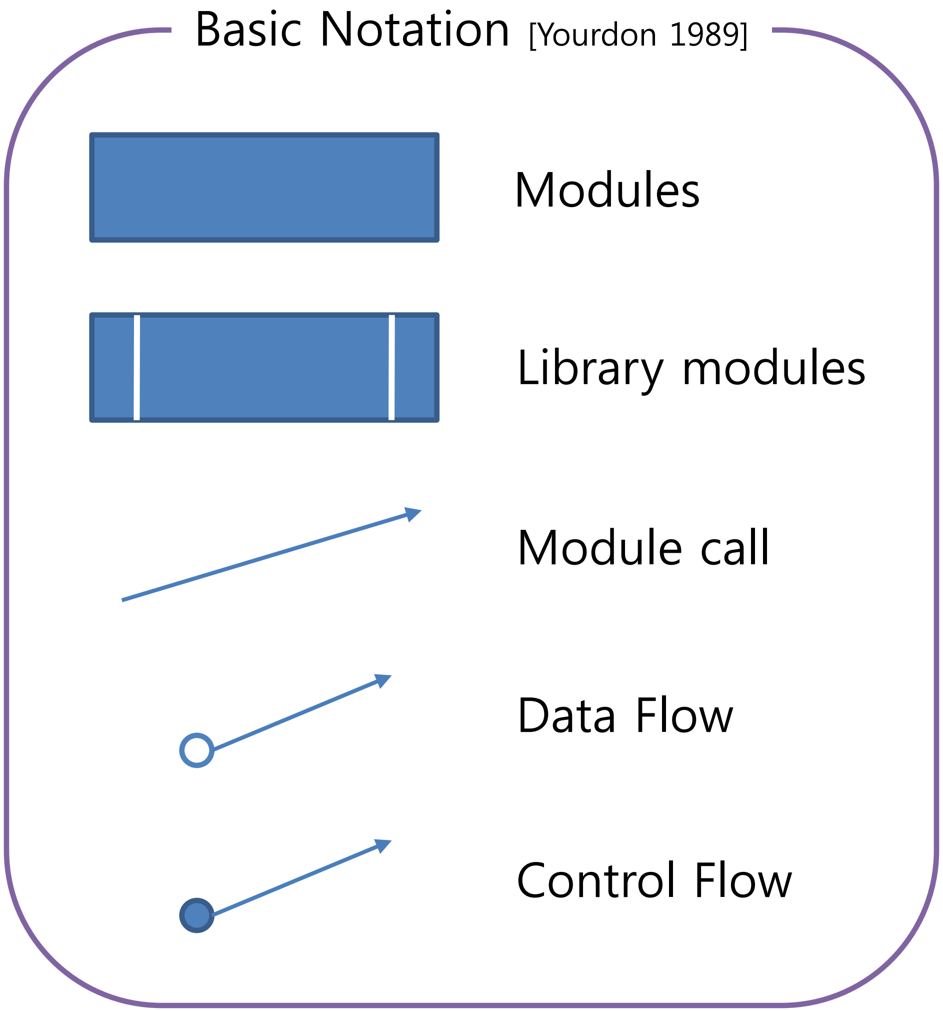
Control

Input

Process

Output

# Structured Charts – Notation



Basic Notation [Yourdon 1989]

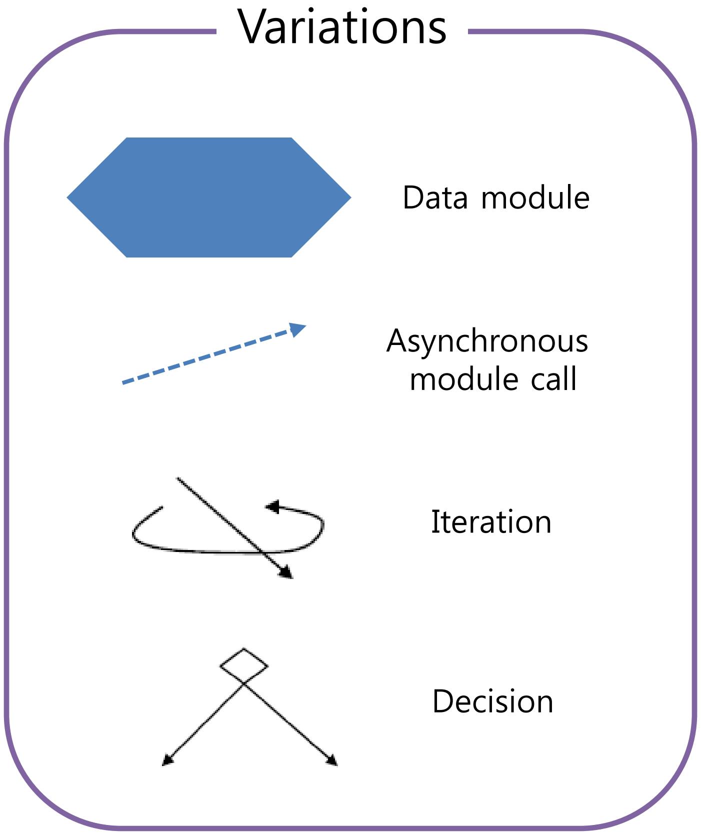
Modules

Library modules

Module call

Data Flow

Control Flow



Variations

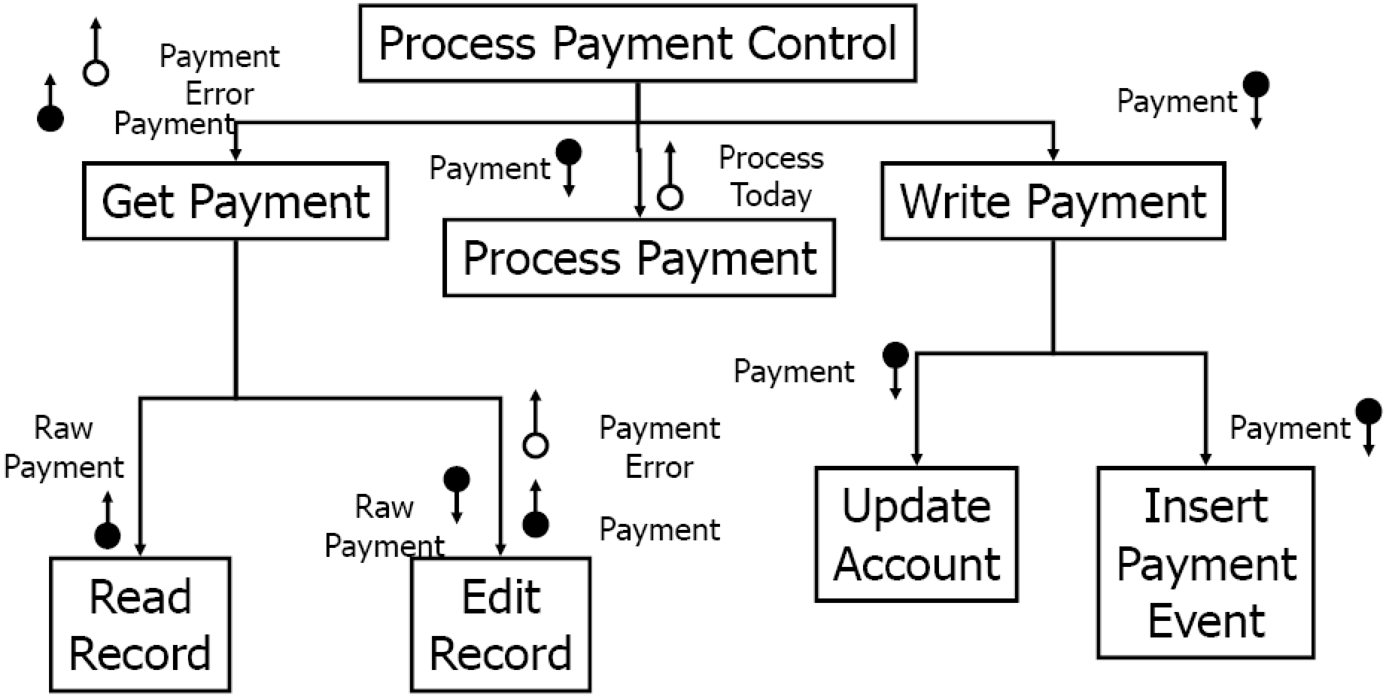
Data module

Asynchronous module call

Iteration

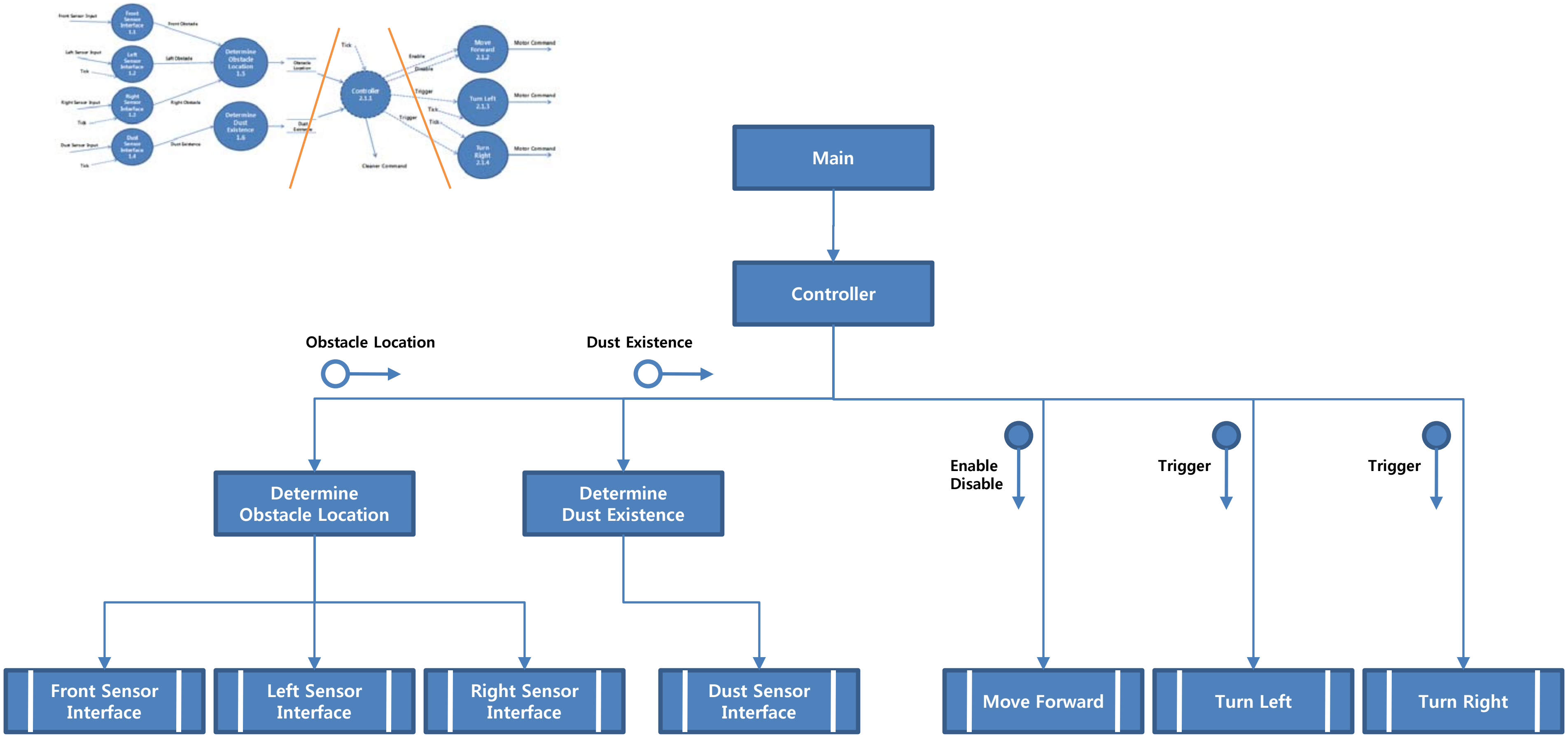
Decision

# Structured Charts - Example



Zhou Qun, Kendra Hamilton, and Ibrahim Jadalowen (2002)

# Structured Charts – RVC Example (Basic)



Main

Controller

Obstacle Location

Dust Existence

Determine Obstacle Location

Determine Dust Existence

Enable Disable

Trigger

Trigger

Front Sensor Interface

Left Sensor Interface

Right Sensor Interface

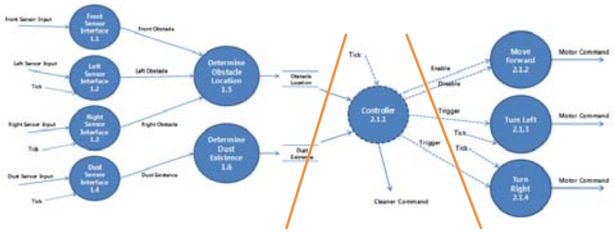
Dust Sensor Interface

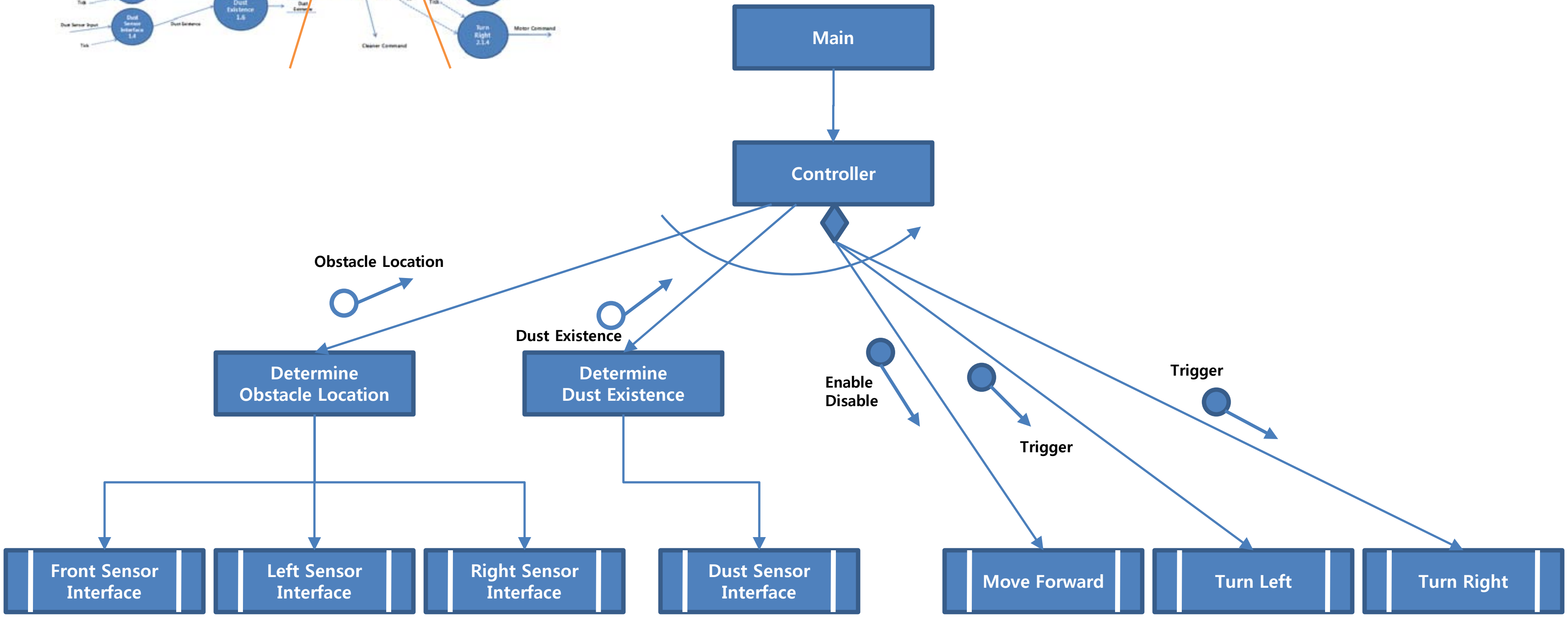
Move Forward

Turn Left

Turn Right

# Structured Charts – RVC Example (Advanced)





Main

Controller

Obstacle Location

Dust Existence

Enable
Disable

Trigger

Trigger

Determine Obstacle Location

Determine Dust Existence

Front Sensor Interface

Left Sensor Interface

Right Sensor Interface

Dust Sensor Interface

Move Forward

Turn Left

Turn Right

# Pros of SASD

- Has distinct milestones, allowing easier project management tracking.
- Very visual – easier for users/programmers to understand
- Makes good use of graphical tools
- Well known in industry
- A mature technique
- Process-oriented way is a natural way of thinking
- Flexible
- Provides a means of requirements validation
- Relatively simple and easy to read

# Pros of SASD

- System Context Diagram
  - Provides a black box overview of the system and the environment

- Event List
  - Provides a guidance for functionality
  - Provides a list of system inputs and outputs
  - A means of requirements summarization
  - Can be used to define test cases (as we will see soon.)

- Data Flow Diagram (DFD)
  - Ability to represent data flows
  - Functional decomposition (divide and conquer)

# Pros of SASD

- Data Dictionary
  - Simplifies data requirements
  - Used at high or low level analysis
- Entity Relationship Diagram (ERD)
  - Commonly used and well understood
  - A graphical tool, so easy to read by analysts
  - Data objects and relationships are portrayed independently from the process
  - Can be used to design database architecture
  - Effective tool to communicate with DBAs
- Process Specification
  - Expresses the process specifications in a form that can be verified

# Pros of SASD

- State Transition Diagrams
  - Models real-time behavior of the processes in the DFD

- Structure Charts
  - Modularity improves the system maintainability
  - Provides a means for transition from analysis to design
  - Provides a synchronous hierarchy of modules

# Cons of SASD

- Ignores non-functional requirements.
- Minimal management involvement
- Non-iterative – waterfall approach
- Not enough use-analysts interaction
- Does not provide a communication process with users.
- Hard to decide when to stop decomposing.
- Does not address stakeholders' needs.
- Does not work well with Object-Oriented programming languages.

# Cons of SASD

- System Context Diagram
  - Does not provide a specific means to determine the scope of the system.
- Event List
  - Does not define all functionalities.
  - Does not define specific mechanism for event interactions.
- Data Flow Diagram (DFD)
  - Weak display of input/output details
  - Confused for users to understand.
  - Does not represent time.
  - No implied sequencing
  - Assigns data stores in the early analysis phase without much deliberation.

# Cons of SASD

- Data Dictionary
  - No functional details
  - Formal language is confusing to users.
- Entity Relationship Diagram (ERD)
  - May be confused for users due to its formal notation.
  - Become complex in large systems.
- Structure Chart
  - Does not work well for asynchronous processes such as networks.
  - Could be too large to be effectively understood with larger programs.

# Cons of SASD

- Process Specification
  - They may be too technical for users to understand.
  - Difficult to stay away from the current “How to implement.”
- State Transition Diagram
  - Explains what action causes a state change, but not when or how often.

# When to use SASD?

- Well-known problem domains
- Contract projects where SRS should be specified in details
- Real-time systems
- Transaction processing systems
- Not appropriate when time to market is short.

- In recent years,
  SASD is widely used in developing real-time embedded systems.

# SASD vs. OOAD

- Similarities
  - The both have started off from programming techniques.
  - The both use graphical design and tools to analyze and model requirements.
  - The both provide a systematic step-by-step process for developers.
  - The both focus on the documentation of requirements.

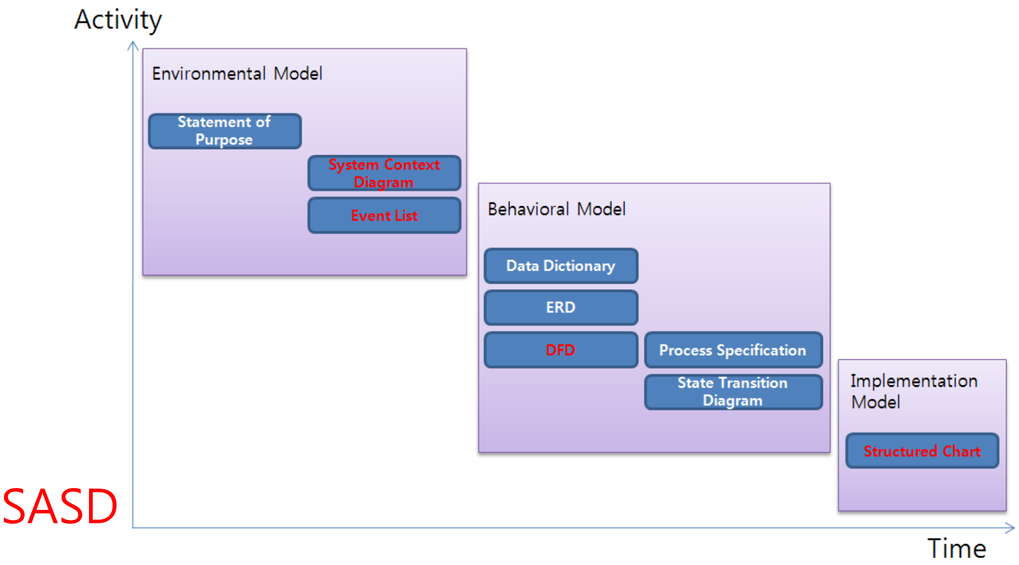
- Differences
  - SASD is process-oriented.
  - OOAD is data(object)-oriented.
  - OOAD encapsulates as much of the system's data and processes into objects,
  - While SASD separates them as possible as it can.

# Class Questions

- What is your opinion on ?
  - Does it reduce maintainability costs?
  - Is it useful?
  - Is it efficient?
  - Is it appropriate for E-commerce(business) development?
- What is SASD's target domain?

# Summary

- SASD is a process-driven software analysis technique.
- SASD has a long history in the industry and it is very mature.
- It provides a good documentation for requirements.
- In recent years, it is widely used for developing real-time embedded system's software.



# Final Presentation (OOAD vs. SASD)

- English presentation
- Compare OOAD with SASD using your elevator controller team project.
  - Pros and Cons of SASD and OOAD for developing elevator controllers respectively
  - Your opinion and suggestion!!!