OOAD vs. SASD

200412299 고성학

200412320 문한균

Aspect of Requirements analysis

- By OOAD

  -Analysis by Use Case

  -Two Caregories

  -Functional Requirement

  -Nonfunctional Requirement

- By SASD

  -Analysis by Event (Statement of purpose)

  -Need to more clearly

  in SASD, determines the I/O by 'Statement of purpose'

Aspect of Initial Design

- By OOAD

  -Designed by Use Case

  -Described by the relationship between Actor and Use Case

  -Make a Use Case Diagram

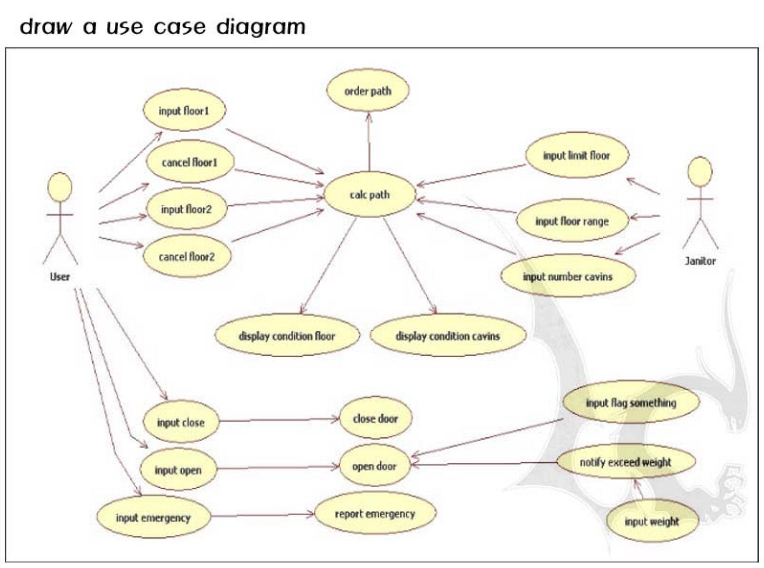
- By SASD

  -Designed by Input/Output

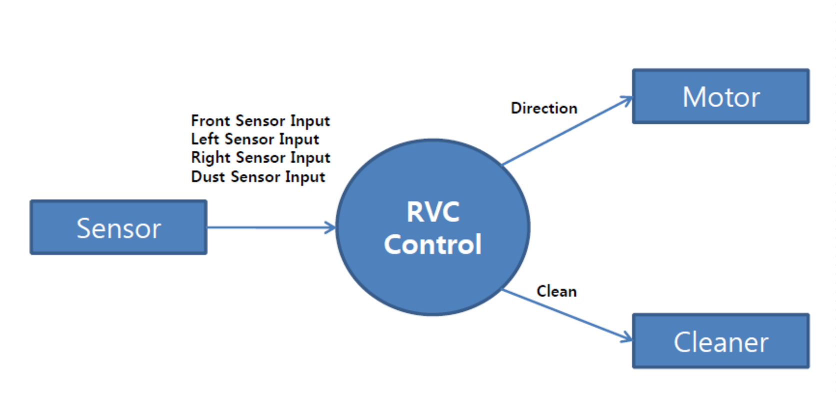
  -First, clarify the input and output

  -make a System Context Diagram

Aspect of Initial Design



VS



Aspect of Design

- By OOAD

  -Establish the relationship between Class

  -Add attribute in Class

  -Make Sequence Diagram

    -Describe the relationship between Actor and Class by Use Case

  -Based on a Sequence Diagram, make Operation contract

Aspect of Design(2)

- By SASD

  -Make Event List(Based on I/O)

  -Make Data Flow Diagram

  -Improvement based on I/O

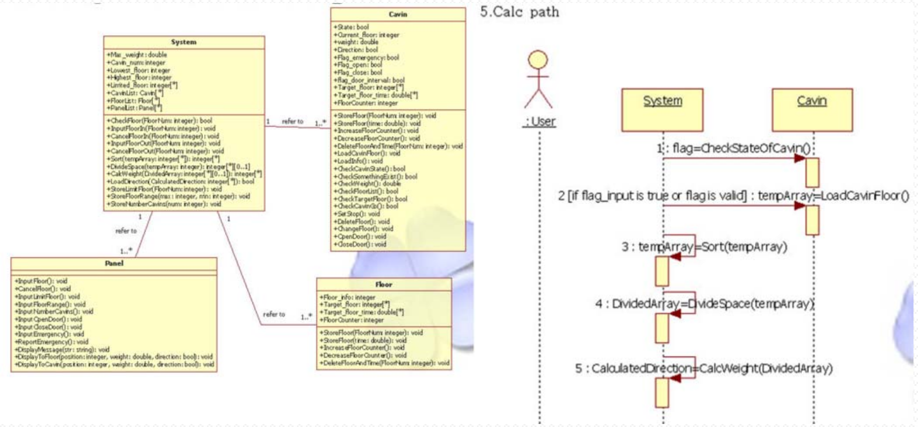
  -Input, Process, Output

  -Make Data Dictionary(Based on Event List)

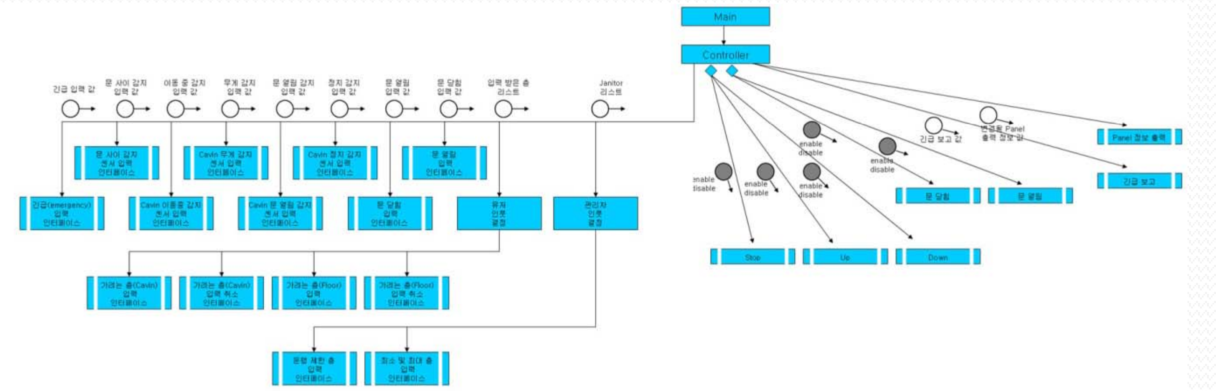
  -Make State Transition Diagram(for Control Process)

  -Make Structured Chart

Aspect of Design(3)


System
+Max_weight : double
+Cavin_num: integer
+Lowest_floor: integer
+Highest_floor: integer
+Limited_floor: integer[*]
+CavinList: Cavin[*]
+FloorList: Floor[*]
+PanelList : Panel[*]
+CheckFloor(FloorNum: integer): bool
+InputFloorIn(FloorNum: integer) : void
+CancelFloorIn(FloorNum: integer) : void
+InputFloorOut(FloorNum: integer): void
+CancelFloorOut(FloorNum: integer): void
+Sort(tempArray: integer[*]): integer[*]
+DivideSpace(tempArray: integer): integer[*][0..1]
+CalcWeight(DividedArray: integer[*][0..1]): integer[*]
+LoadDirection(CalculatedDirection: integer[*]): bool
+StoreLimitFloor(FloorNum: integer): void
+StoreFloorRange(max : integer, min: integer): void
+StoreNumberCavins(num: integer): void
Cavin
+State: bool
+Current_floor: integer
+weight: double
+Direction: bool
+Flag_emergency: bool
+Flag_open: bool
+Flag_close: bool
+flag_door_interval: bool
+Target_floor: integer[*]
+Target_floor_time: double[*]
+FloorCounter: integer
+StoreFloor(FloorNum: integer): void
+StoreFloor(time: double): void
+IncreaseFloorCounter(): void
+DecreaseFloorCounter(): void
+DeleteFloorAndTime(FloorNum: integer): void
+LoadCavinFloor(): void
+LoadInfo() : void
+CheckCavinState() : bool
+CheckSomethingExist(): bool
+CheckWeight(): double
+CheckFloorList(): bool
+CheckTargetFloor(): bool
+CheckCavinGo(): bool
+SetStop(): void
+DeleteFloor(): void
+ChangeFloor(): void
+OpenDoor() : void
+CloseDoor() : void
1
refer to
1..*
Panel
+InputFloor(): void
+CancelFloor(): void
+InputLimitFloor(): void
+InputFloorRange(): void
+InputNumberCavins(): void
+InputOpenDoor() : void
+InputCloseDoor() : void
+InputEmergency(): void
+ReportEmergency() : void
+DisplayMessage(str : string): void
+DisplayToFloor(position: integer, weight: double, direction: bool) : void
+DisplayToCavin(position: integer, weight: double, direction: bool): void
Floor
+Floor_info: integer
+Target_floor: integer[*]
+Target_floor_time: double[*]
+FloorCounter : integer
+StoreFloor(FloorNum: integer): void
+StoreFloor(time: double): void
+IncreaseFloorCounter(): void
+DecreaseFloorCounter(): void
+DeleteFloorAndTime(FloorNum: integer): void
5.Calc path
: User
System
Cavin
1 : flag=CheckStateOfCavin()
2 [if flag_input is true or flag is valid] : tempArray=LoadCavinFloor()
3 : tempArray=Sort(tempArray)
4 : DividedArray=DivideSpace(tempArray)
5 : CalculatedDirection=CalcWeight(DividedArray)

VS


Main
Controller
긴급 입력 값
문 사이 감지 입력 값
이동 중 감지 입력 값
무게 감지 입력 값
문 열림 감지 입력 값
정지 감지 입력 값
문 열림 입력 값
문 닫힘 입력 값
입력 받은 층 리스트
Janitor 리스트
문 사이 감지 센서 입력 인터페이스
Cavin 무게 감지 센서 입력 인터페이스
Cavin 정지 감지 센서 입력 인터페이스
문 열림 입력 인터페이스
긴급(emergency) 입력 인터페이스
Cavin 이동중 감지 센서 입력 인터페이스
Cavin 문 열림 감지 센서 입력 인터페이스
문 닫힘 입력 인터페이스
유저 인풋 결정
관리자 인풋 결정
가려는 층(Cavin) 입력 인터페이스
가려는 층(Cavin) 입력 취소 인터페이스
가려는 층(Floor) 입력 인터페이스
가려는 층(Floor) 입력 취소 인터페이스
운행 제한 층 입력 인터페이스
최소 및 최대 층 입력 인터페이스
enable disable
enable disable
enable disable
enable disable
enable disable
긴급 보고 값
변경된 Panel 출력 정보 값
Panel 정보 출력
긴급 보고
문 닫힘
문 열림
Stop
Up
Down

Private Opinion

- OOAD

-More flexible than SASD

ex)if program's function is expanded, or program's input is changed, OOAD is can more flexibly response.

- Function segmentation can be easily by Use Case.

- Confirmed by performing repeat.

So, if the requirements is uncertain, it is good to use.

- but result is more complicated than SASD

Private Opinion(2)

- SASD

  - Difficult to modify

    ex)If program's function is expanded, or program's input is changed, SASD is need to redesign.

  - It need to simplify for improvement of DFD.

  - SASD is progress by waterfall model.

    So, It need to definite requirement for design.

  - Result is more simple than OOAD

  - It is so difficult to define a State Machine

Private Opinion(3)

- OOAD is easier than SASD for modify. So, It is suitable for frequently changing program like a Application.
- OOAD is suitable for imprecise requirements.

- SASD is harder than OOAD for modify. So, It is suitabl e for unchangeable program like a system program or e mbedded program.
- SASD suitable for definite requirements.

Epilogue

We are friendly with C++.
So we have feel a familiar with OOAD.
SASD is not familiar, we feel a relatively difficult.
But now we think, SASD have the advantage of make a simple program.
Considering these points, we can design a program more practicaly.