

소프트웨어 공학개론

-최종 보고

Team 1

201411259 고수창

201511243 김동언

201511263 박종엽

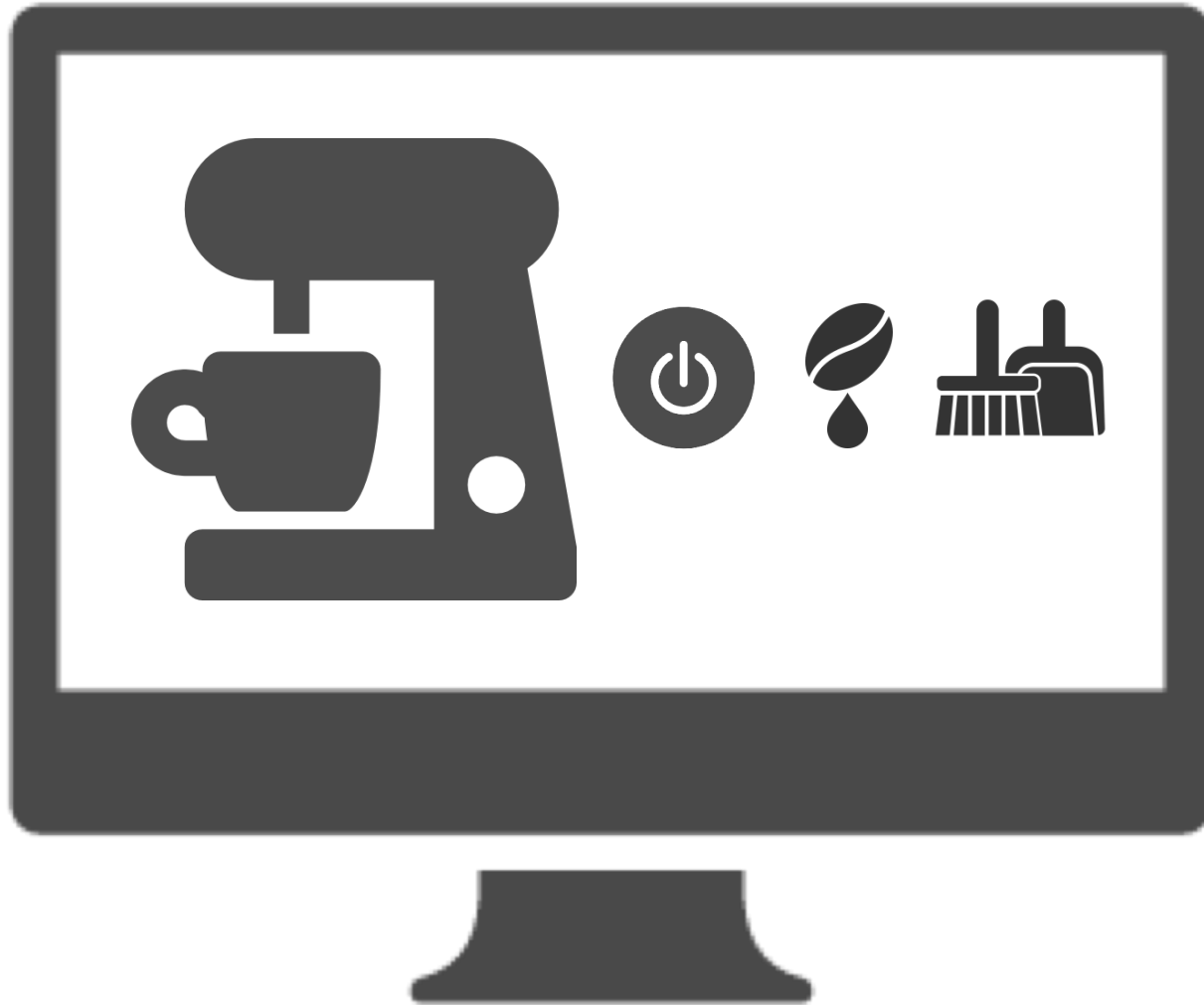
201511280 이선엽

목차

1. 목표
2. 설계와 개발 과정
3. System Requirement Analysis
4. Unit Development
5. Unit Testing
6. Full Development
7. System Testing
8. Cross Testing
9. Summary

1. 목표

Make coffee machine software



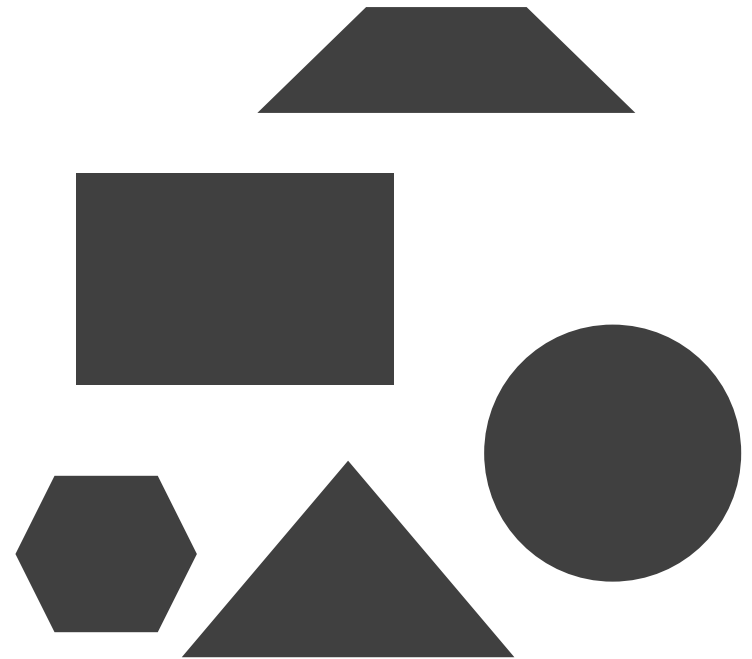
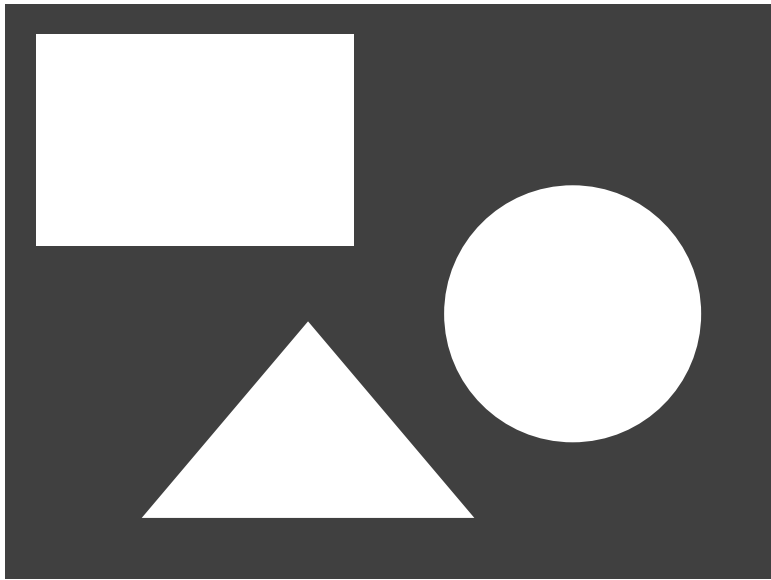
1. 목표

Get command by key stroke



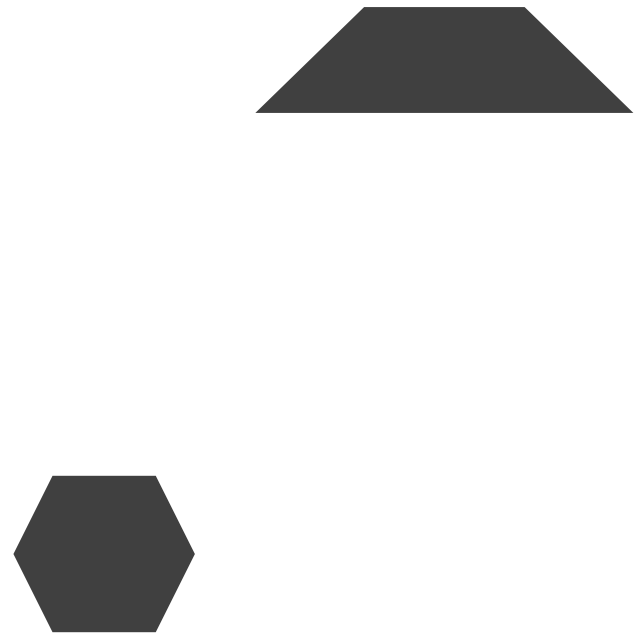
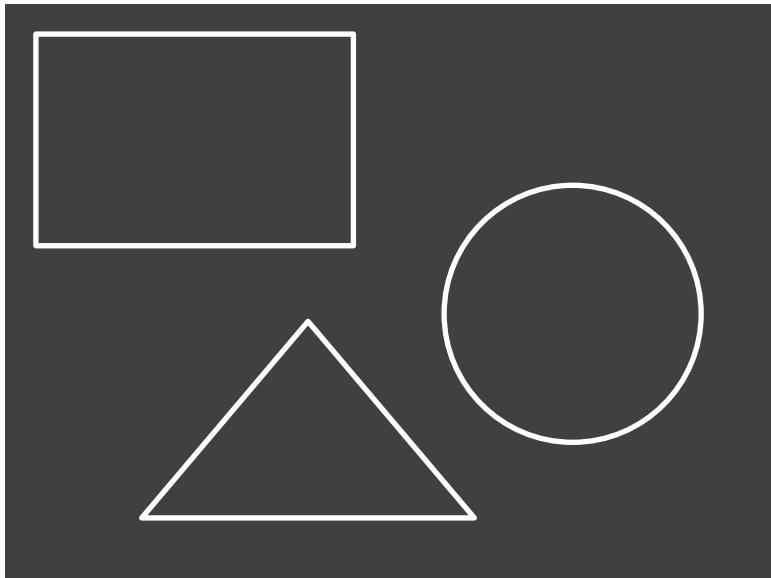
2. 설계와 개발 과정

Make sure requirement



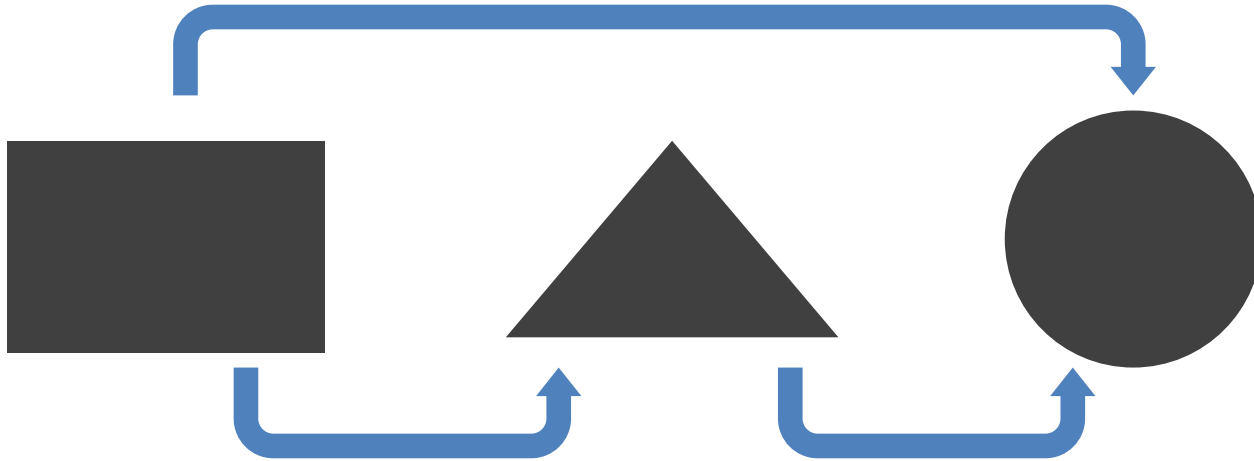
2. 설계와 개발 과정

Make sure requirement



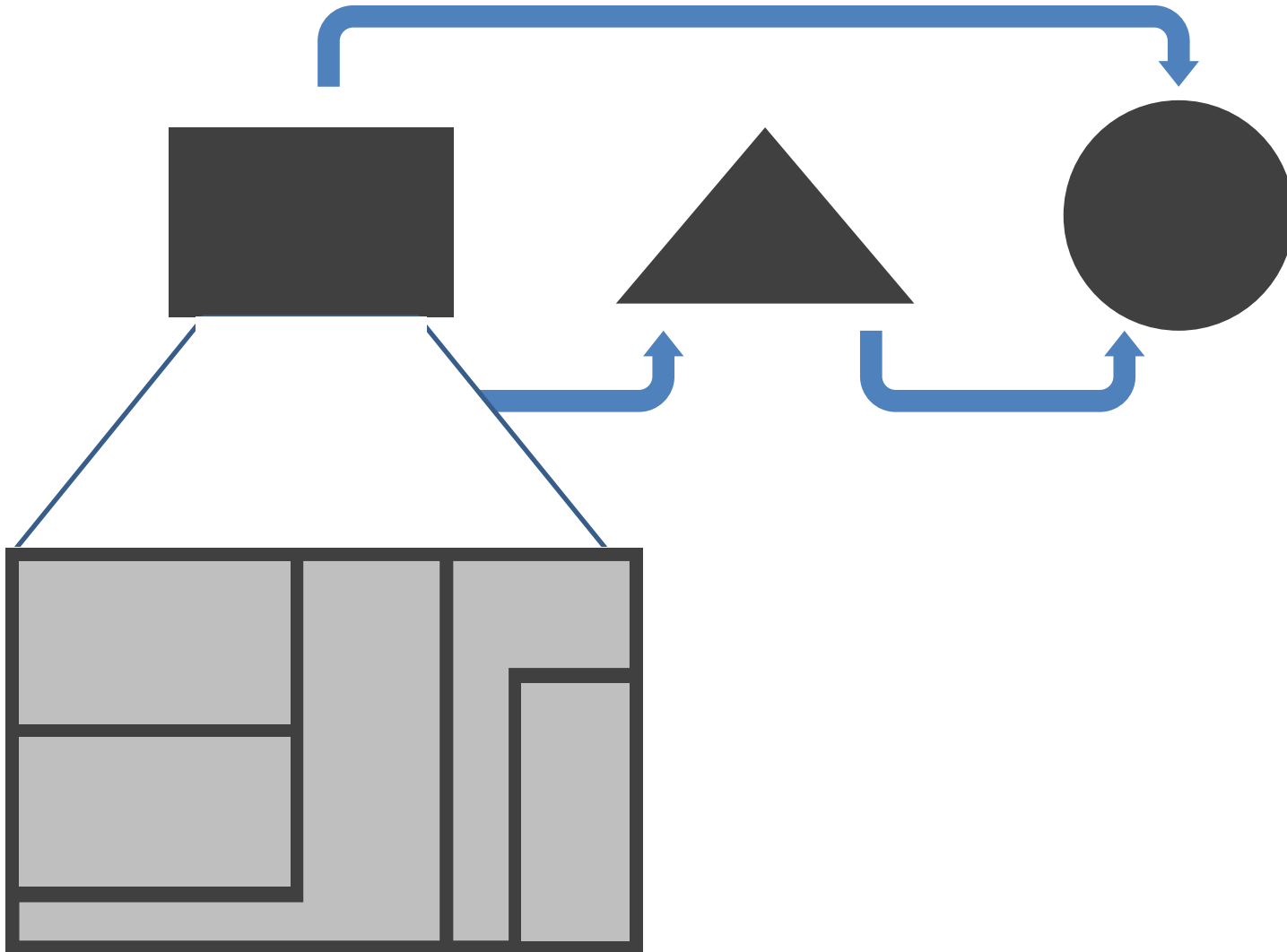
2. 설계와 개발 과정

Establish development process



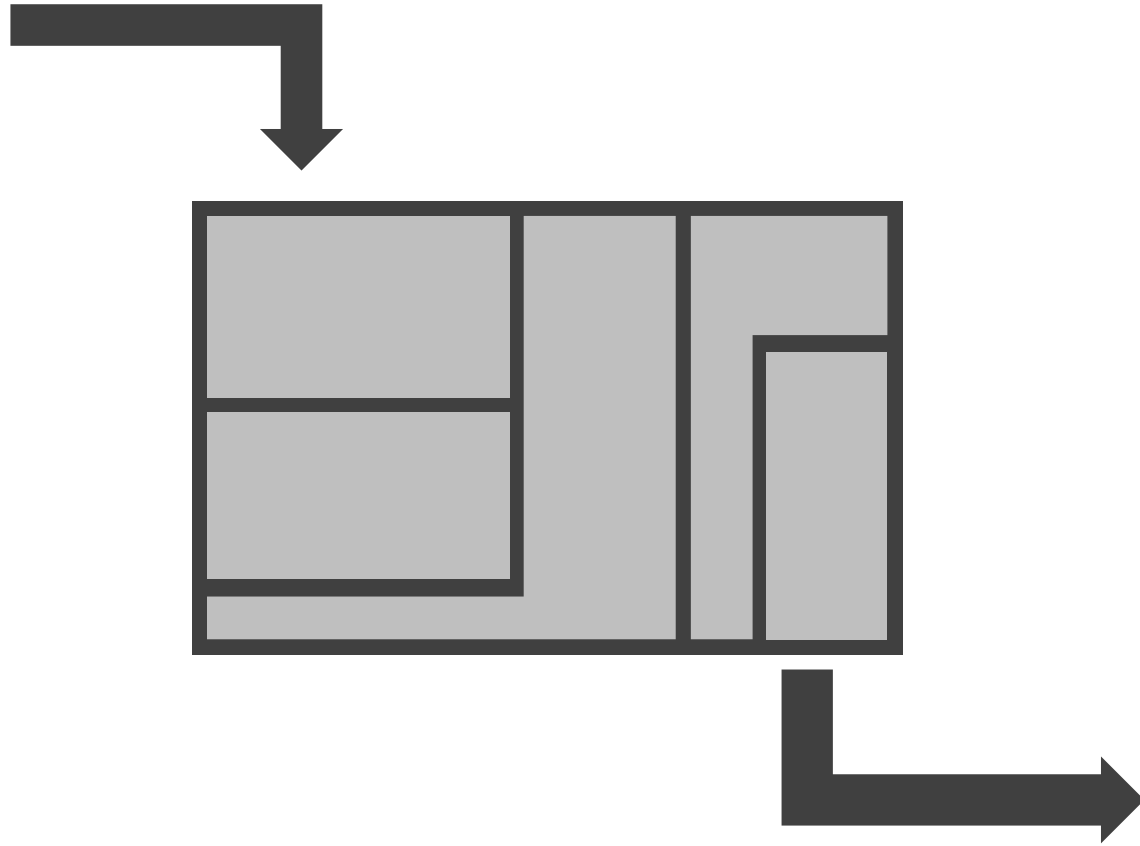
2. 설계와 개발 과정

Establish development process



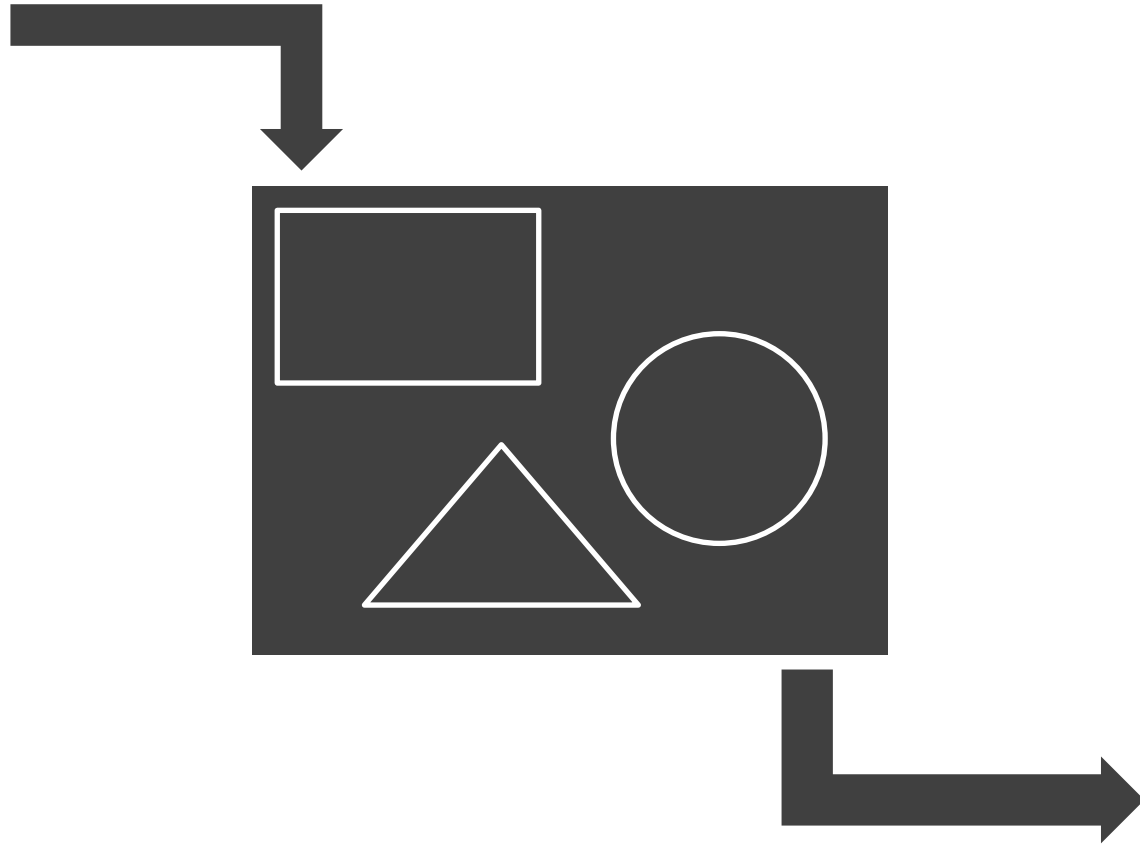
2. 설계와 개발 과정

Unit test



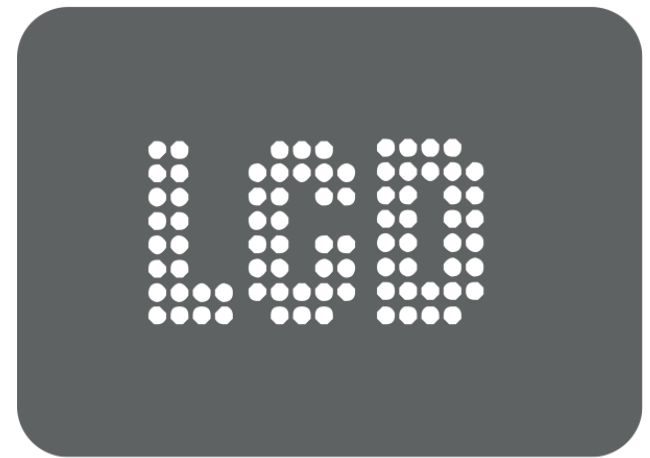
2. 설계와 개발 과정

System test



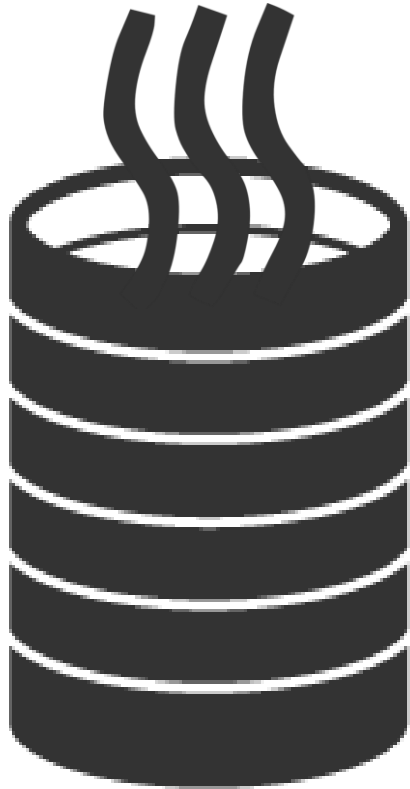
3. Software Requirement Analysis

Reservation, Concentration, Display

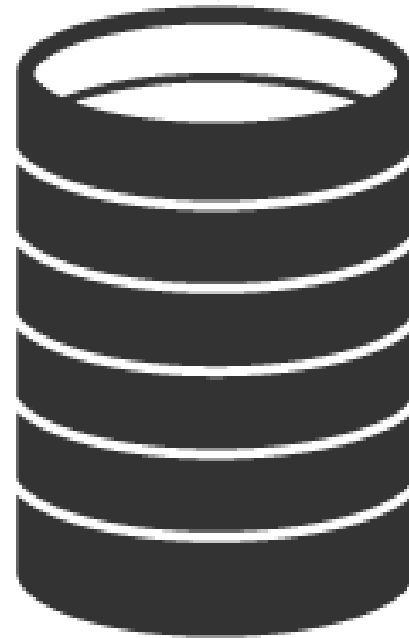


3. Software Requirement Analysis

Co-exist hot and cold water



Hot

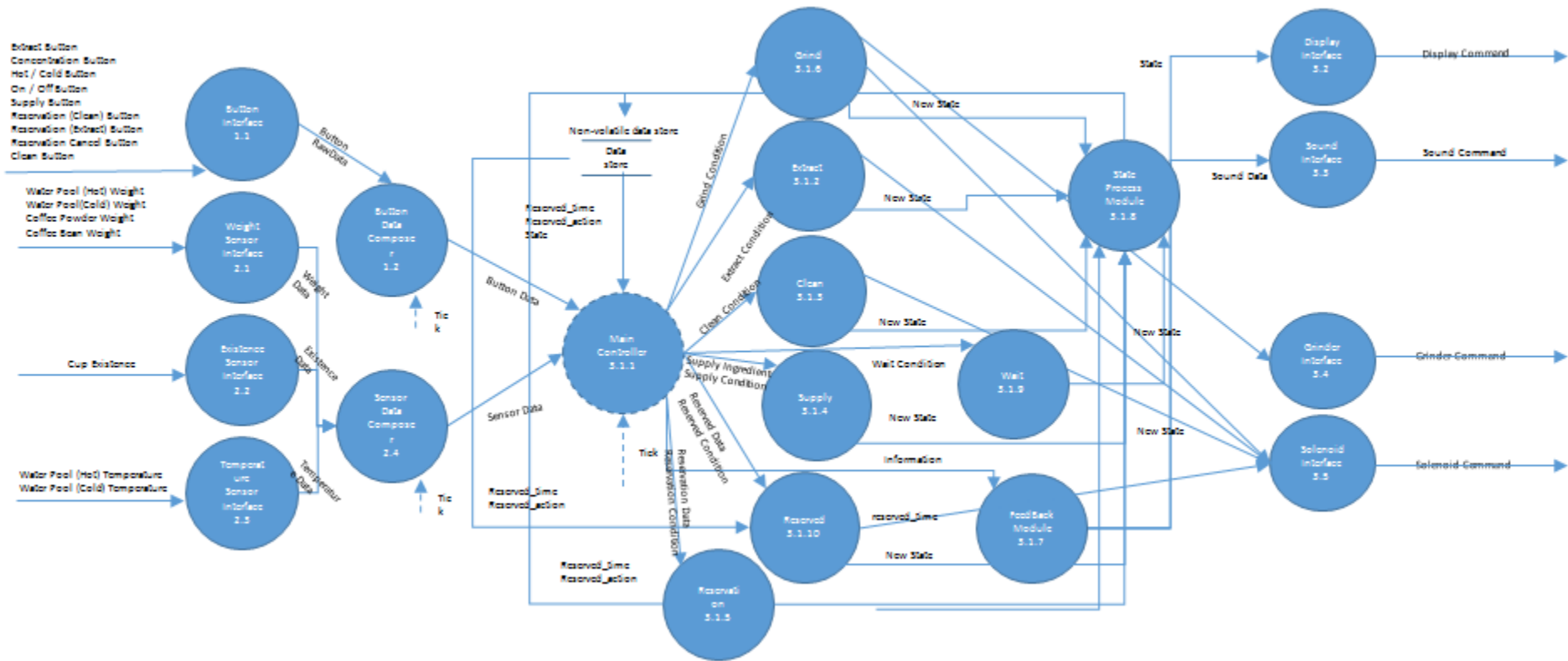


Cold

Not need to boil cold water

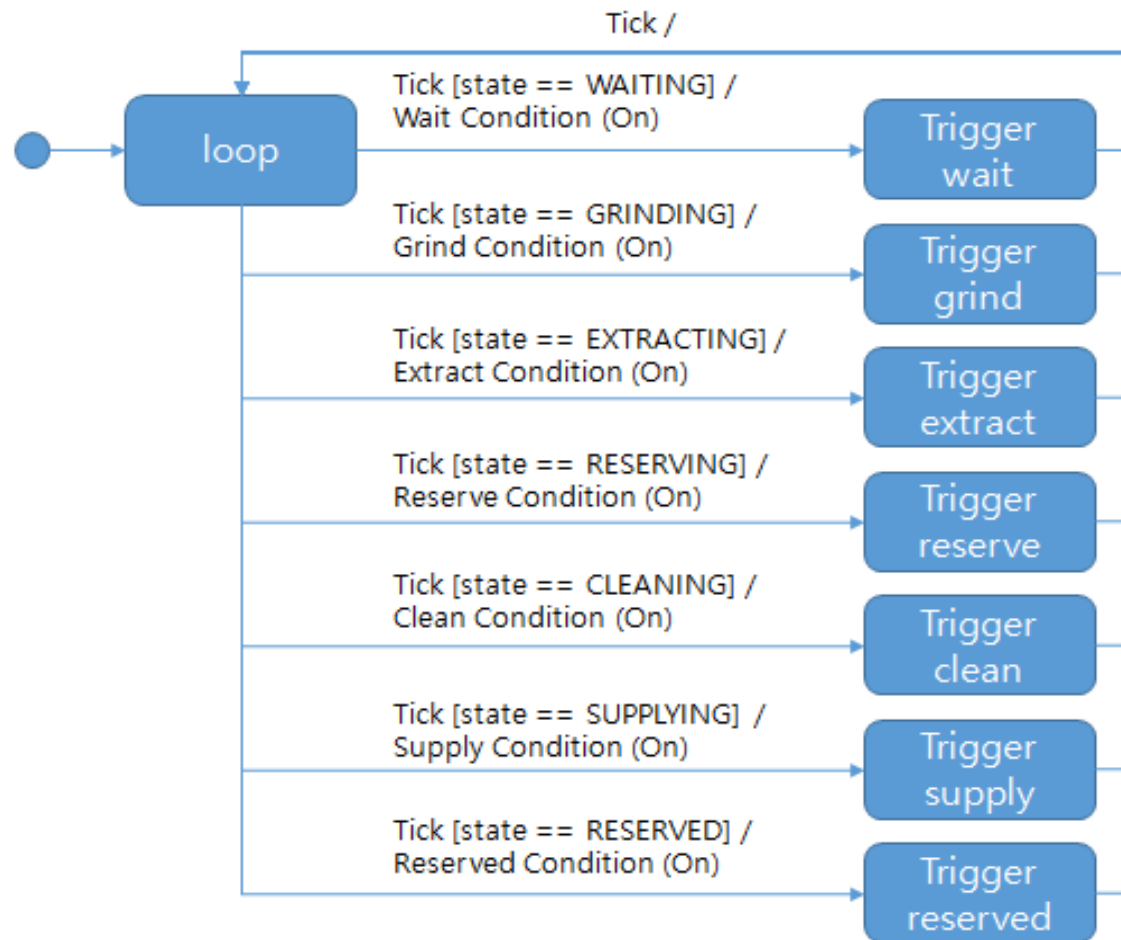
3. Software Requirement Analysis

Data Flow Diagram(DFD)



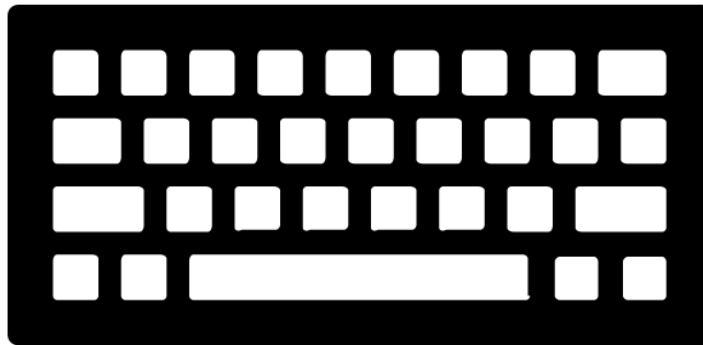
3. Software Requirement Analysis

State Transition Diagram (STD)



4. Unit Development

Command by keystroke



Created by useiconic.com
from Noun Project

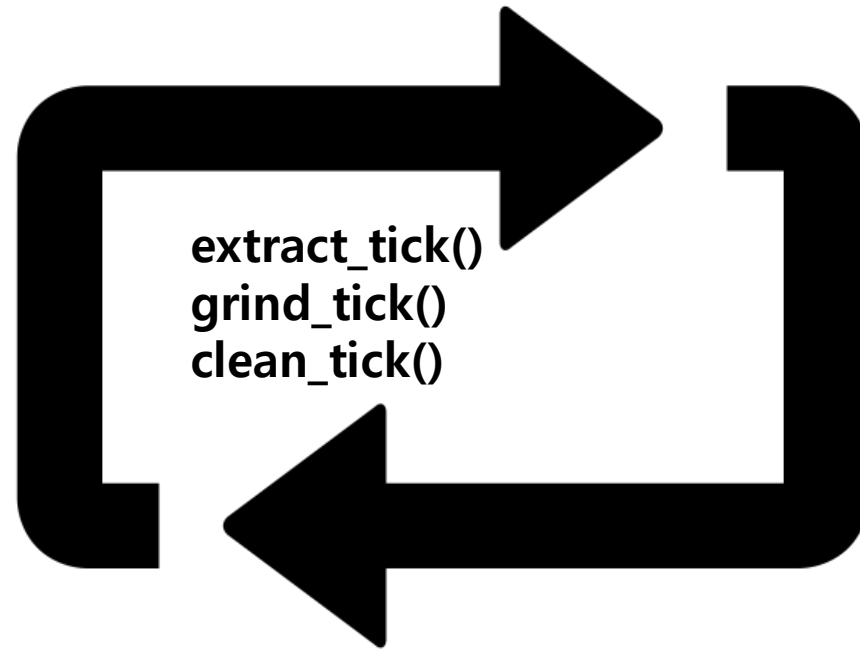
4. Unit Development

Command by keystroke



4. Unit Development

No thread, socket, fork




Created by useiconic.com
from Noun Project

4. Unit Development

Start flag by time

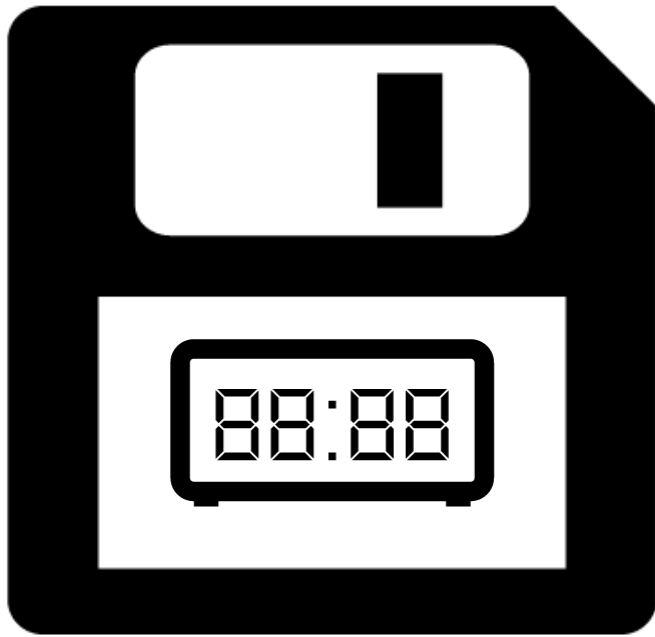



START



4. Unit Development

End flag by time



4. Unit Development

No thread, socket, fork



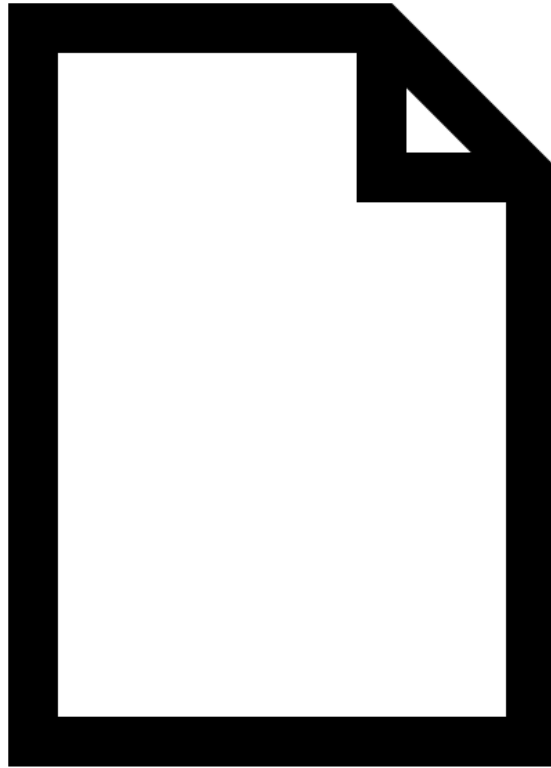
One thread

Non-blocking

Well-ported

4. Unit Development

Sensor

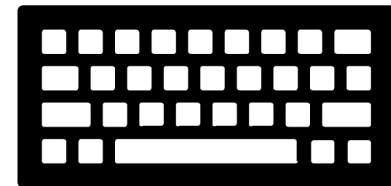


Created by Galaxicon
from Noun Project

4. Unit Development

Sensor

```
btn_init(&btn_temperature, 'm');  
btn_init(&btn_onoff, 'o');  
btn_init(&btn_extract, 'e');  
btn_init(&btn_concentration, 'c');  
btn_init(&btn_supply, 's');  
btn_init(&btn_reservation_clean, 'l');  
btn_init(&btn_reservation_extract, 't');  
btn_init(&btn_reservation_cancel, 'a');  
btn_init(&btn_clean, 'n');
```

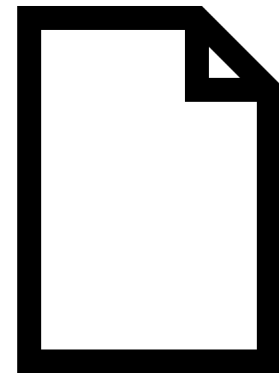


Created by useiconic.com
from Noun Project

4. Unit Development

Sensor

```
sensor_init(&sensor_hot_weight, "hot_weight.txt", 0, 500);  
sensor_init(&sensor_cold_weight, "cold_weight.txt", 0, 500);  
sensor_init(&sensor_cup_existence, "cup_existence.txt", 0, 1);  
sensor_init(&sensor_coffee_bean_weight, "coffee_bean_weight.txt", 0, 100);  
sensor_init(&sensor_coffee_powder_weight, "coffee_powder_weight.txt", 0, 100);  
sensor_init(&sensor_hot_temperature, "coffee_water_hot_temperature.txt", -100,  
           100);  
sensor_init(&sensor_cold_temperature, "coffee_water_cold_temperature.txt", -100,  
           100);  
sensor_init(&sensor_use_count, "sensor_count.txt", 0, 10);
```



4. Unit Development

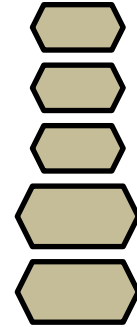
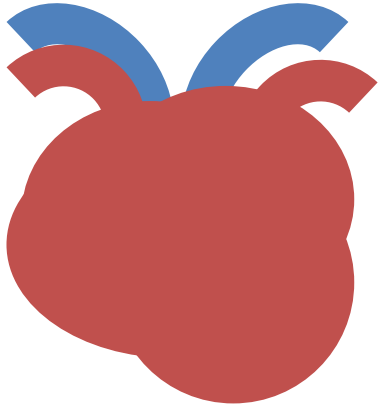
Structures

```
struct btn_ctx {  
    char key;  
    int pressed;  
}
```

```
struct sensor_ctx {  
    char * filename;  
    int min;  
    int max;  
}
```


5. Unit Test

Test each modules run as their purpose



5. Unit Test

Ctest

CTEST

ctest is a unit test framework for software written in C.

Features:

- adding tests with minimal hassle (no manual adding to suites or testlists!)
- supports suites of tests
- supports `setup()` `teardown()` per test
- output format not messed up when tests fail, so easy to parse.
- displays elapsed time, so you can keep your tests fast
- uses coloring for easy error recognition
- only use coloring if output goes to terminal (not file/process)
- it's small (a little over 300 lines of code!)
- it's easy to integrate (only 1 header file)
- has SKIP option to skip certain test (no commenting test out anymore)
- Linux + OS/X support

5. Unit Test

Ctest

```
CTEST(sensor_test, extract_test1) {
    sensor_update(&sensor_hot_weight, 500);
    sensor_update(&sensor_cup_existence, 1);
    sensor_update(&sensor_coffee_powder_weight, 0);
    new_state(STATE_EXTRACT);
    extract_tick(state);
    ASSERT_EQUAL(state, STATE_GRIND);
}
```

```
CTEST(sensor_test, extract_test2) {
    sensor_update(&sensor_hot_weight, 200);
    sensor_update(&sensor_coffee_powder_weight, 15);
    new_state(STATE_EXTRACT);
    extract_tick(state);
    ASSERT_EQUAL(state, STATE_WAIT);
}
```

```
CTEST(sensor_test, extract_test3) {
    sensor_update(&sensor_hot_weight, 500);
    sensor_update(&sensor_cup_existence, 1);
    sensor_update(&sensor_coffee_powder_weight, 10);
    new_state(STATE_EXTRACT);
    extract_tick(state);
    ASSERT_EQUAL(state, STATE_EXTRACT);
}
```

```
extern int temp_flag;
```

```
CTEST(sensor_test, test4) {
    sensor_update(&sensor_hot_weight, 500);
    sensor_update(&sensor_cold_weight, 0);
    sensor_update(&sensor_cup_existence, 1);
    sensor_update(&sensor_coffee_powder_weight, 10);
    temp_flag = 1;
```

```
int main(int argc, const char *argv[])
```

```
{
```

```
    sensor_init(&sensor_hot_weight, "hot_weight.test.txt", 0, 500);
    sensor_init(&sensor_cold_weight, "cold_weight.test.txt", 0, 500);
    sensor_init(&sensor_cup_existence, "cup_existence.test.txt", 0, 1);
    sensor_init(&sensor_coffee_bean_weight, "coffee_bean_weight.test.txt", 0, 100);
    sensor_init(&sensor_coffee_powder_weight, "coffee_powder_weight.test.txt", 0, 100);
    sensor_init(&sensor_hot_temperature, "coffee_water_hot_temperature.test.txt", -100, 100);
    sensor_init(&sensor_cold_temperature, "coffee_water_cold_temperature.test.txt", -100, 100);
    sensor_init(&sensor_use_count, "sensor_count.test.txt", 0, 10);
```

```
    int result = ctest_main(argc, argv);
    return result;
```

```
}
```

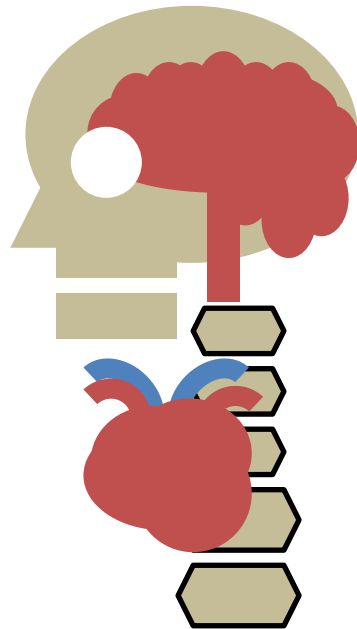
5. Unit Test

Ctest

```
[OK]
TEST 4/25 sensor_test:extract_test3 NORMAL
[OK]
TEST 5/25 sensor_test:test4 ERROR_COLD
[OK]
TEST 6/25 sensor_test:test5 ERROR_CUP
[OK]
TEST 7/25 sensor_test:test6 ERROR_COFFEE_BEAN
[OK]
TEST 8/25 sensor_test:test7 [OK]
TEST 9/25 sensor_test:test8 [OK]
TEST 10/25 sensor_test:test9 [OK]
TEST 11/25 sensor_test:test10 [OK]
TEST 12/25 sensor_test:test11 [OK]
TEST 13/25 sensor_test:test12 [OK]
TEST 14/25 sensor_test:test13 [OK]
TEST 15/25 sensor_test:test14 [OK]
TEST 16/25 sensor_test:test15 ERROR_SOLENOID
[OK]
TEST 17/25 sensor_test:test16 [OK]
TEST 18/25 sensor_test:test17 [OK]
TEST 19/25 sensor_test:test18 [OK]
TEST 20/25 sensor_test:test19 ERROR_GRIND
[OK]
TEST 21/25 sensor_test:test20 [OK]
TEST 22/25 sensor_test:test21 [OK]
TEST 23/25 sensor_test:test22 [OK]
TEST 24/25 sensor_test:test23 [OK]
TEST 25/25 sensor_test:test24 [OK]
RESULTS: 25 tests (25 ok, 0 failed, 0 skipped) ran in 117 ms
C:\WUsers\WPJY\Desktop\W2016se-master>
```

6. Full Development

Merge each modules



6. Full Development

Merge each modules

```
extern struct btn_ctx btn_ctxs[9];

void btn_init(struct btn_ctx *c, char init_c) {
    c->key = init_c;
    c->pressed = 0;
}

void btns_update() {
    char c = getch();
    if(c == btn_temperature.key) {
        btn_press(&btn_temperature);
    } else if(c == btn_extract.key) {
        btn_press(&btn_extract);
    } else if(c == btn_supply.key) {
        btn_press(&btn_supply);
    } else if(c == btn_concentration.key) {
        btn_press(&btn_concentration);
    } else if(c == btn_reservation_clean.key) {
        btn_press(&btn_reservation_clean);
    } else if(c == btn_reservation_extract.key) {
        btn_press(&btn_reservation_extract);
    } else if(c == btn_reservation_cancel.key) {
        btn_press(&btn_reservation_cancel);
    } else if(c == btn_clean.key) {
        btn_press(&btn_clean);
    } else if(c == btn_onoff.key) {
        btn_press(&btn_onoff);
    } else {
        ungetch(c);
    }
    state_process();
}
```



6. Full Development

Merge each modules

```
struct solenoid {
    int element;
    float quantity;
};

struct solenoid solenoid_data;
int temp_flag = 0; // 0 : hot / 1 : cold
int timer = -1; // mysleep_init()

void extract_tick(int now_state) {
    int Count = sensor_get(&sensor_use_count);
    if (now_state == STATE_EXTRACT) {
        if (sensor_get(&sensor_coffee_powder_weight) < 10) {
            new_state(STATE_GRIND);
        } else if (sensor_get(&sensor_coffee_powder_weight) >= 10) {
            if(temp_flag == 0){ //Hot
                if(sensor_get(&sensor_hot_weight) >= 300 && sensor_get(&sensor_cup_existence) == true && Count < 10){
                    Solenoid_Command(On);

                    if(mysleep(&timer, 3)) {
                        Solenoid_Command(Off);
                        new_state(STATE_WAIT);
                        sensor_sub(&sensor_hot_weight, (concentration+1)*100);
                        sensor_sub(&sensor_coffee_powder_weight, 10);
                        sensor_add(&sensor_use_count, 1);
                        return;
                    }
                }
            }
            if(sensor_get(&sensor_hot_weight) < 300 || sensor_get(&sensor_cup_existence) == false || Count >= 10){
                if(mysleep(&timer, 2)) {
                    new_state(STATE_WAIT);
                    error_msg = NULL;
                    werase(stdscr);
                } else {
                    if(!error_msg) werase(stdscr);
                    if(sensor_get(&sensor_hot_weight) < 300){
                        error_msg = "물(근)이 부족합니다.";
                    }else if(sensor_get(&sensor_cup_existence) == false){
                        error_msg = "컵이 부족합니다.";
                    }else if(Count >= 10){
                        error_msg = "정소가 필요합니다.";
                    }
                }
            }
        }
    }
}
```

6. Full Development

Merge each modules

```
struct solenoid solenoid_data;

int grind_timer = -1;
void grind_tick(int now_state)
{
    if(now_state == STATE_GRIND){
        if (sensor_get(&sensor_coffee_bean_weight) < 10)
        {
            if(mysleep(&grind_timer, 1)) {
                new_state(STATE_WAIT);
                error_msg = NULL;
                werase(stdscr);
            } else {
                if(!error_msg) werase(stdscr);
                error_msg = "원두가 부족합니다.";
            }
        }
        else if(sensor_get(&sensor_coffee_bean_weight) >= 10)
        {
            Solenoid_Command(On);
            if(mysleep(&grind_timer, 3)) {
                sensor_sub(&sensor_coffee_bean_weight,10); // 소모되는 원두의 양
                sensor_add(&sensor_coffee_powder_weight,10);
                Solenoid_Command(Off);
                new_state(STATE_EXTRACT);
            }
        }
    }
}
```



6. Full Development

Merge each modules

```
void state_process() {  
  
    if(btn_is_pressed(&btn_temperature)){  
        temp_flag = !temp_flag;  
        btn_release(&btn_temperature);  
    }else if(btn_is_pressed(&btn_extract)){  
        new_state(STATE_EXTRACT);  
        btn_release(&btn_extract);  
    }else if(btn_is_pressed(&btn_supply)){  
        new_state(STATE_SUPPLY);  
        btn_release(&btn_supply);  
  
        supply_type = 0;  
        supply_amount = 0;  
    }else if(btn_is_pressed(&btn_concentration)){  
        btn_release(&btn_concentration);  
        concentration++;  
        concentration = concentration%3;  
    }else if(btn_is_pressed(&btn_reservation_clean)){  
        if(state == STATE_WAIT){  
            new_state(STATE_RESERVE);  
            reserve_action = CLEAN;  
        }  
  
        btn_release(&btn_reservation_clean);  
    }else if(btn_is_pressed(&btn_reservation_extract)){  
        if(state == STATE_WAIT){  
            new_state(STATE_RESERVE);  
            reserve_action = MK_COFFEE;  
        }  
        btn_release(&btn_reservation_extract);  
    }else if(btn_is_pressed(&btn_reservation_cancel)){  
        if(state == STATE_WAIT){  
            new_state(STATE_RESERVE);  
            reserve_action = CLEAN;  
        }  
    }  
}
```



6. Full Development

Merge each modules

```
extern char input_buf[];
extern char allowed_charset[];

int main() {
    time_init();
    power_flag = 1;

    //now_state = 1;
    btn_init(&btn_temperature, 'm');
    btn_init(&btn_onoff, 'o');
    btn_init(&btn_extract, 'e');
    btn_init(&btn_concentration, 'c');
    btn_init(&btn_supply, 's');
    btn_init(&btn_reservation_clean, 'l');
    btn_init(&btn_reservation_extract, 't');
    btn_init(&btn_reservation_cancel, 'a');
    btn_init(&btn_clean, 'n');

    sensor_init(&sensor_hot_weight, "hot_weight.txt", 0, 500);
    sensor_init(&sensor_cold_weight, "cold_weight.txt", 0, 500);
    sensor_init(&sensor_cup_existence, "cup_existence.txt", 0, 1);
    sensor_init(&sensor_coffee_bean_weight, "coffee_bean_weight.txt", 0, 100);
    sensor_init(&sensor_coffee_powder_weight, "coffee_powder_weight.txt", 0, 100);
    sensor_init(&sensor_hot_temperature, "coffee_water_hot_temperature.txt", -100,
    .....
    100);
    sensor_init(&sensor_cold_temperature, "coffee_water_cold_temperature.txt", -100,
    .....
    100);
    sensor_init(&sensor_use_count, "sensor_count.txt", 0, 10);
    sensor_update(&sensor_use_count, 0);
```



6. Full Development

Main loop classify by flag

```
while (1) {  
    print_state();  
    btns_update();  
  
    if(power_flag){  
        wait_tick(state);  
        grind_tick(state);  
        extract_tick(state);  
        reserve_tick(state);  
        clean_tick(state);  
        supply_tick(state);  
        reserved_tick(state);  
    }  
}
```

6. Full Development

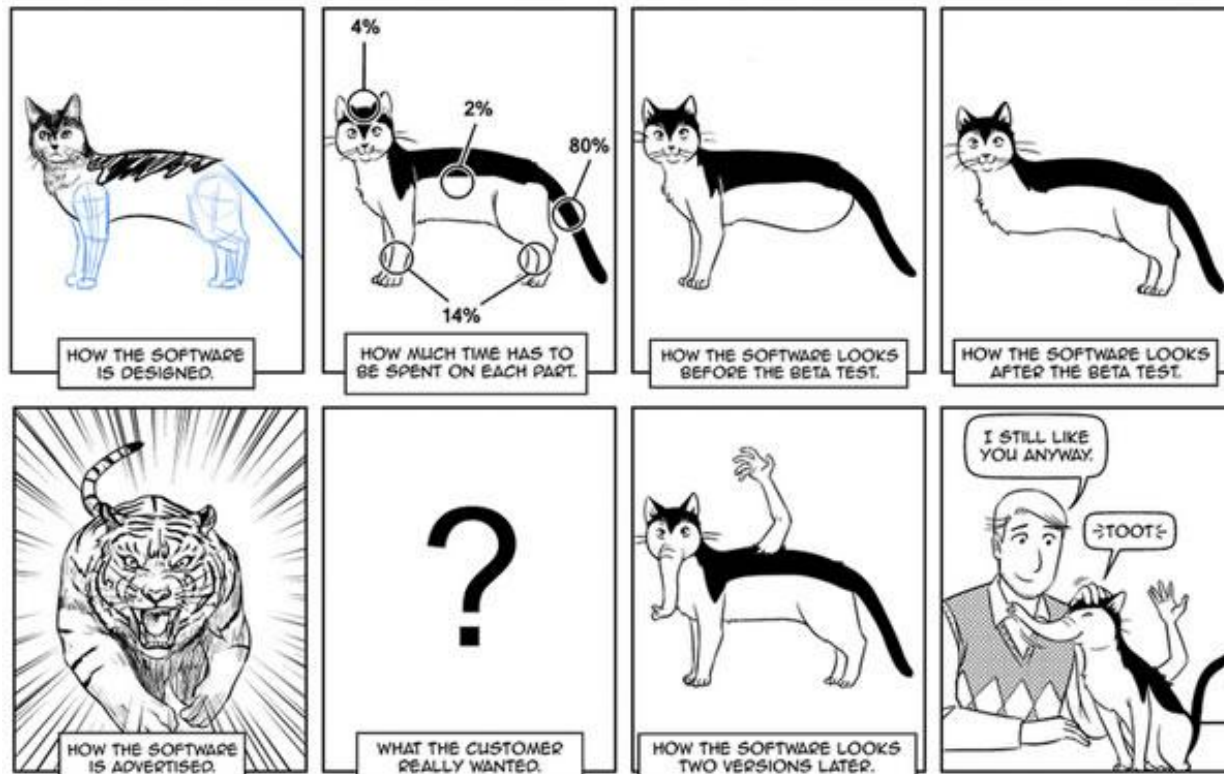
Merge each modules

```
┌───┴───┐
U
현재 시간                15:11
현재 상태              대기 중
선택 온도              중간
선택 온도              온
물(온) 잔량            100ml
물(냉) 잔량            500ml
커피 잔량 (원두)      30g
가루 유무              X
커피 가루 청소 필요   X
커피 예약 시간        0
청소 예약 시간
└───┬───┘
C
┌───┴───┐
U
q   료          버튼
o   /오        버튼
c   온도 세팅  버튼
m   온도 세팅  버튼
e   온도 세팅  버튼
s   온도 세팅  버튼
n   온도 세팅  버튼
i   온도 세팅  버튼
t   온도 세팅  버튼
a   온도 세팅  버튼
└───┬───┘
C
```

7. System Test

Reason why we should test entire system

Richard's guide to software development



7. System Test

Test list

< Figure 1 Test List >

Test ID.	Description
1	커피가 추출되는지에 대한 여부
2	자원이 공급되는지에 대한 여부 (물, 원두)
3	청소가 실행되는지에 대한 여부
4	기능이 예약되는지에 대한 여부
5	ON/OFF가 작동되는지에 대한 여부
6	커피가 추출 될 때 자원이 감소하는지에 대한 여부
7	청소가 실행 될 때 자원이 감소하는지에 대한 여부

8	예약 작업에 대해 제대로 실행되는지에 대한 여부
9	Requirement에 작성된 대로 추출하는데 걸리는 시간이 맞는지?
10	Requirement에 작성된 대로 청소에 걸리는 시간이 맞는지?
11	커피가루가 없는 경우 커피가루를 갈아내는지
12	커피가루가 없는데 원두도 없는 경우 requirement대로 동작하는지
13	청소가 필요한 시점에 청소를 요청하는지
14	컵이 없는 경우 커피 추출을 안하고 알려주는지
15	청소가 필요한대 청소를 안하고 추출하면 어떻게 되는지
16	커피 추출 중 전원을 종료하면 어떻게 되는지

7. System Test

Test result

Identifier	Input Specification	Output Specification	Result
CM.STC.100	keyStroke - o	CM에 전원이 공급되고 메뉴가 출력됨	P
CM.STC.110	keyStroke - s	어떤 자원을 공급할 것인지 메뉴를 출력함	P
CM.STC.120	keyStroke - n	청소를 실행 중이라는 창이 출력됨	P
CM.STC.121	청소가 실행 중인 상태 (state = STATE_CLEAN)	자원이 감소되는 창이 출력됨	P
CM.STC.122	청소가 실행 중인 상태	정해진 시간 동안 청소 중인 상태가 됨	P
CM.STC.123	자원을 계속 사용하여 청소가 필요한 상태	청소가 필요하다는 창을 띄워줌	P
CM.STC.130	keyStroke - e	커피가 추출되고 커피 추출 중인 창이 출력됨	P
CM.STC.131	추출중인 상태 (state = STATE_EXTRACT)	커피가 추출되면 requirement에 따른 양 만큼 커피원료가 줄어듦	P
CM.STC.132	커피 추출	Requirement에 따른	P

8. Cross Testing

From Team 7

Identifier	Feature	Valid value
CMS_STP_000	커피 추출이 가능하다	온도, 농도 설정 값이 입력되었다는 전제 하에 설정 값에 따른 커피를 추출
CMS_STP_001	조건에 따라 커피추출이 불가능하다.	커피 머신이 Ready상태가 아닐 때 커피 머신 동작 불가 / 커피 추출 후 커피 찌꺼기가 남아있을 때 커피 추출 불가
CMS_STP_002	사용자의 기호에 따라 설정이 가능하다.	입력받은 온도와 농도 상태 확인
CMS_STP_003	커피 추출 또는 청소 예약이 가능하다	원하는 예약 작업과 시간 상태 확인 → Display 화면의 해당 작업에 가장 먼저 실행되어야 할 작업 시간이 출력된다.
CMS_STP_004	커피가루가 없을 경우 원두를 자동 분쇄 후 커피를 추출한다.	커피가루 상태에 따른 커피 머신의 동작 확인
CMS_STP_005	물과 커피의 잔량을 지속적으로 사용자에게 알려준다.	커피 머신이 해당 재료를 사용할 때마다 최신 정보를 화면에 디스플레이
CMS_STP_006	물과 커피는 사용자에게 의해 충전된다.	입력받은 재료량 상태 확인
CMS_STP_007	머신 내부 청소가 가능하다.	청소 명령시의 커피머신 상태 확인
CMS_STP_008	머신은 경고 화면을 보여주고 경고음을 낼 수도 있다.	사용자의 명령을 수행 할 수 없을 때 적절한 경고명령을 Display에 출력하고 경고음으로 알린다.
CMS_STP_009	머신은 언제든지 전원이 꺼질 수 있다.	전원이 꺼졌을 때 모든 정보가 초기화된다.

8. Cross Testing

From Team 7 Result

CMS_STP_000_000	커피추출 명령 (ice)	커피 추출 시간 미달
CMS_STP_000_001	커피추출 명령 (hot)	커피 추출 시간 미달

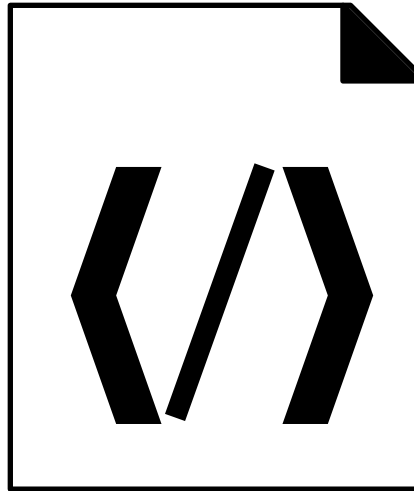
8. Cross Testing

From Team 7 Result - Sol

		고)
CM.STC.132	완성본이 아닌 테스트용 코 드여서 Timer이 3초로 설정 되어 있었음	Timer을 3에서 10초로 수정

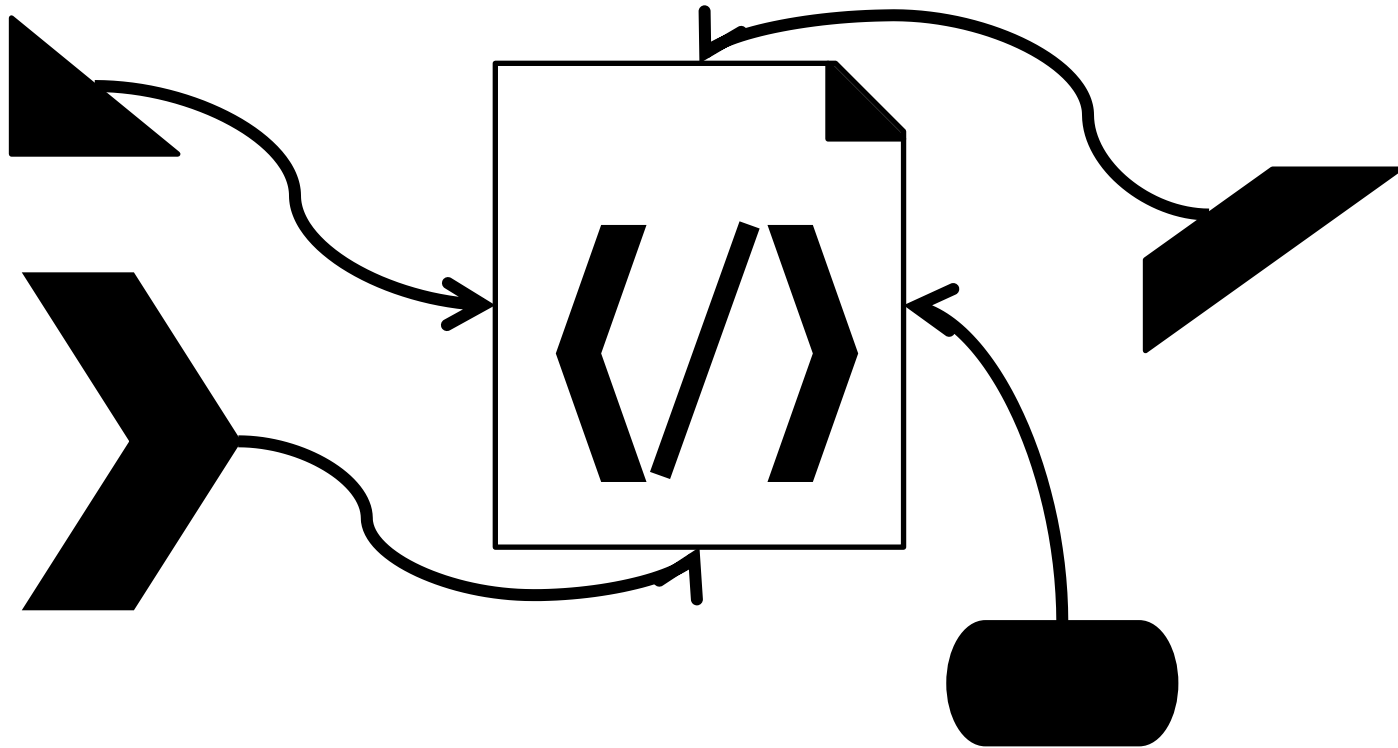
9. Summary

About software engineering



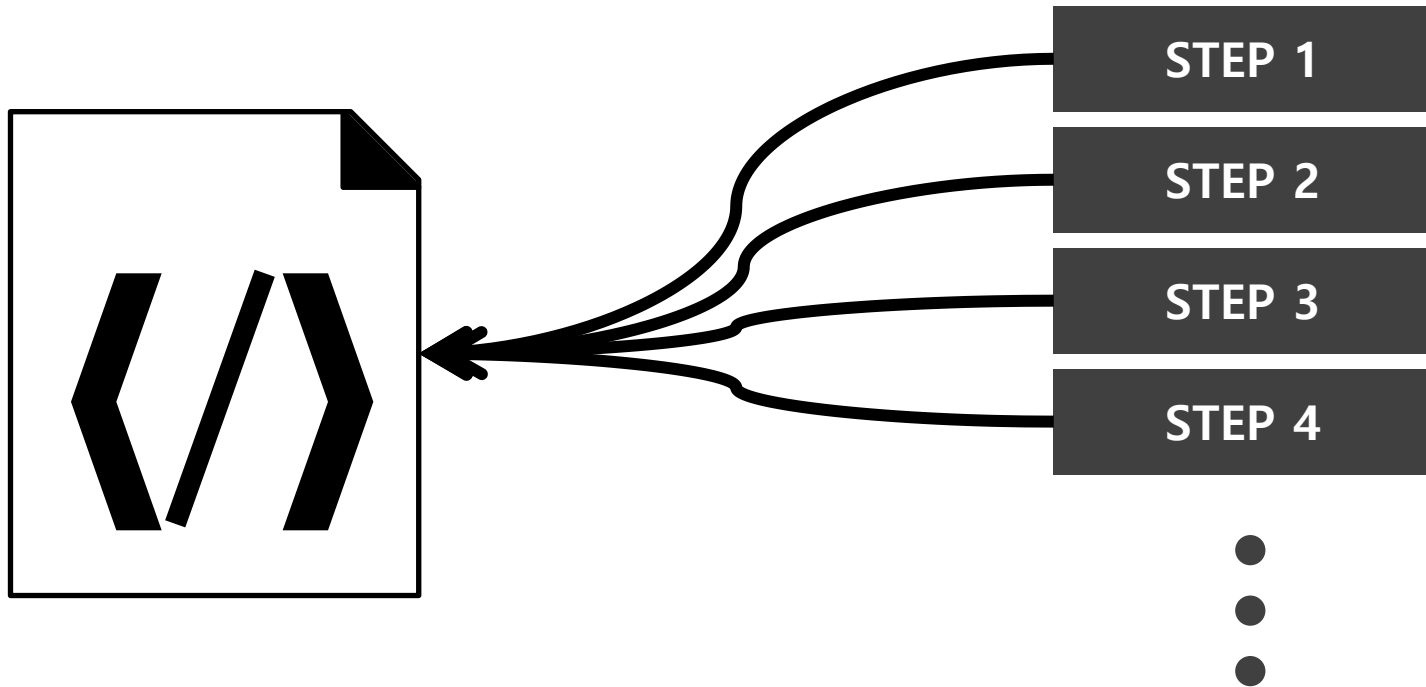
9. Summary

No process



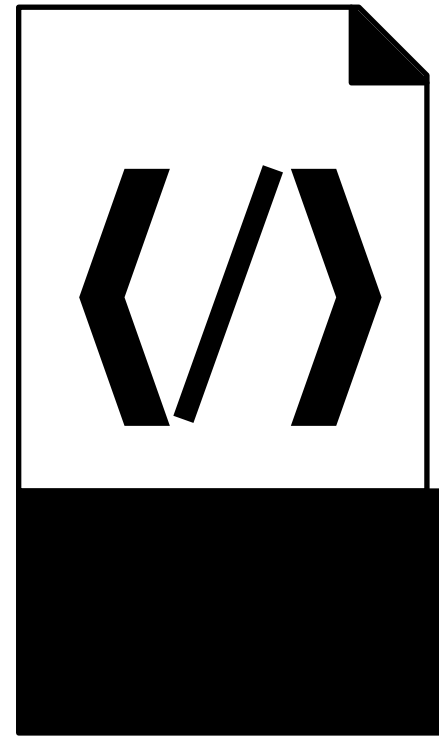
9. Summary

Follow process



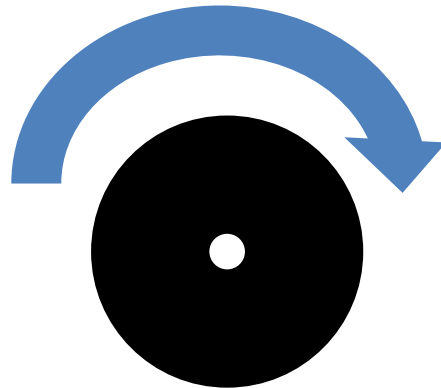
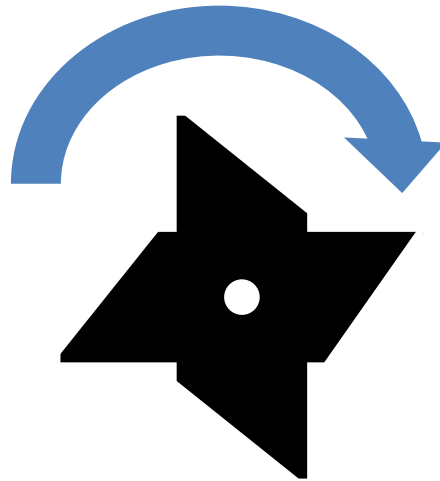
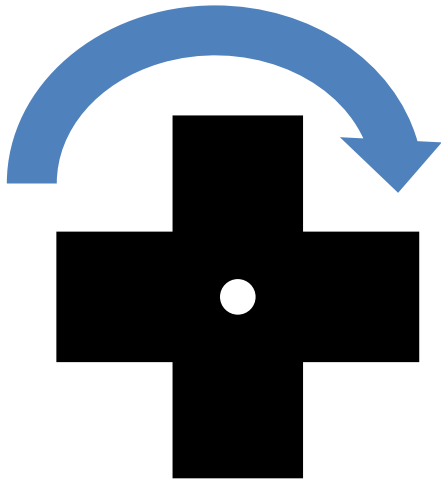
9. Summary

Follow process



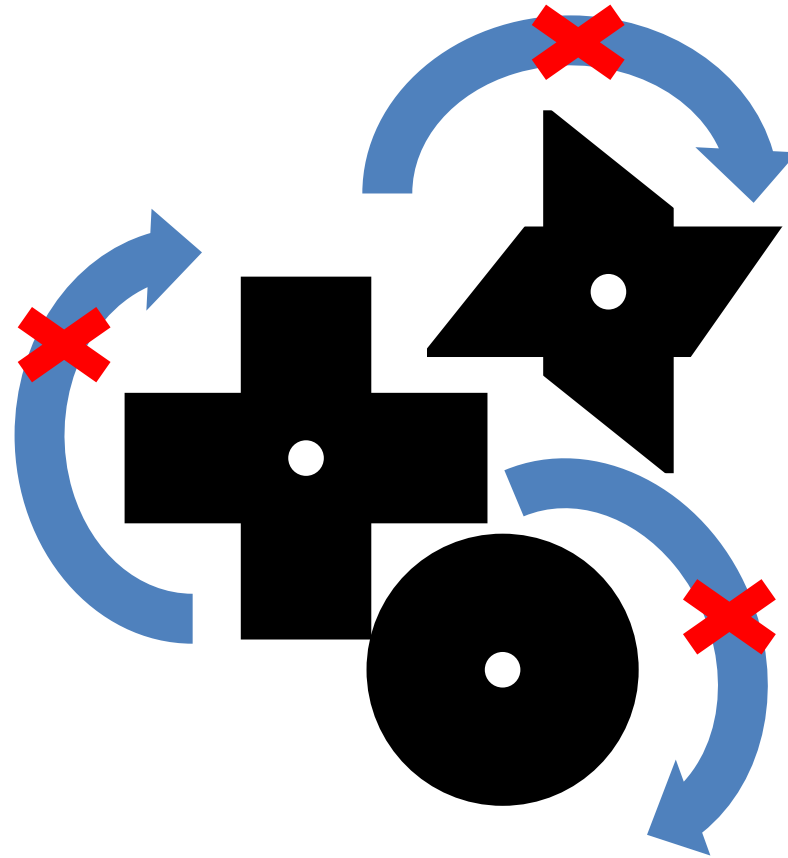
9. Summary

About Testing



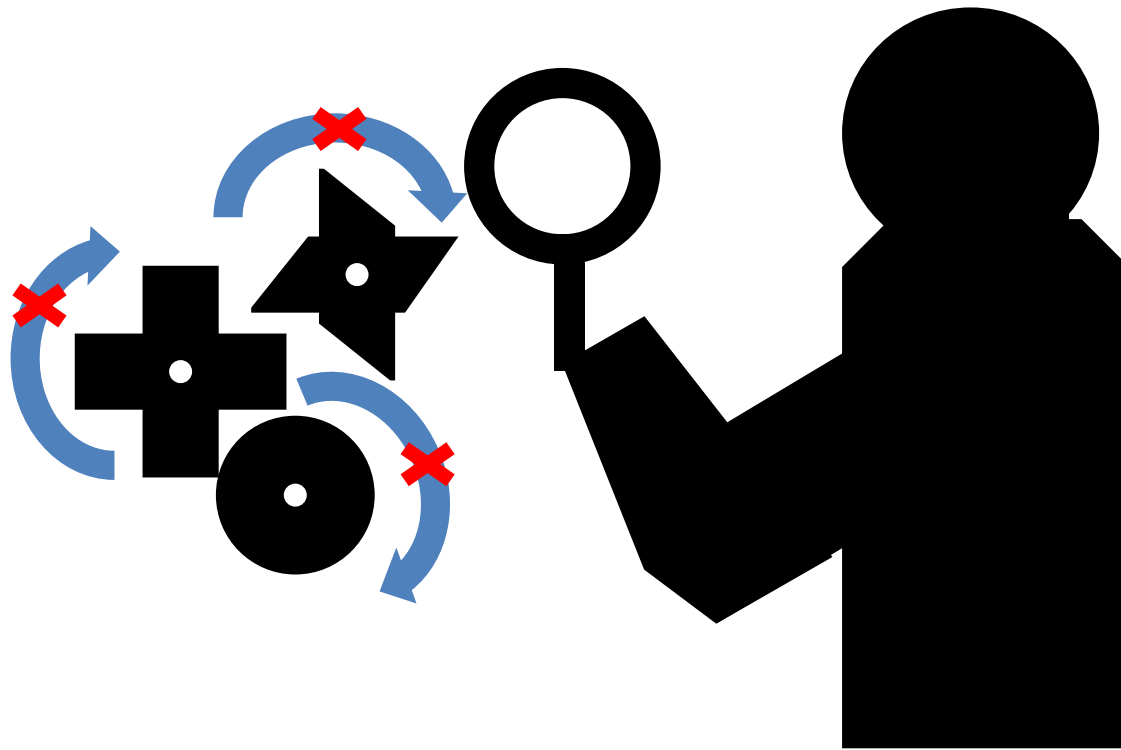
9. Summary

About Testing



9. Summary

Self Test



9. Summary

Cross Test

