[Experience sharing]

How to work effectively

Renesas Design Vietnam Co., Ltd.
MCU modeling, System Level Design Group
NganKTran (ngan.tran.bx@rvc.renesas.com)
Jan 28th, 2015 Rev. 1.00

Agenda

Agenda

- Introduction
- Investigate input material (e.g: HWM)
- Create documents (REQ, INT, VRF-01)
- Code implementation
- Code review
- Functional debug
- Prepare release data

Introduction

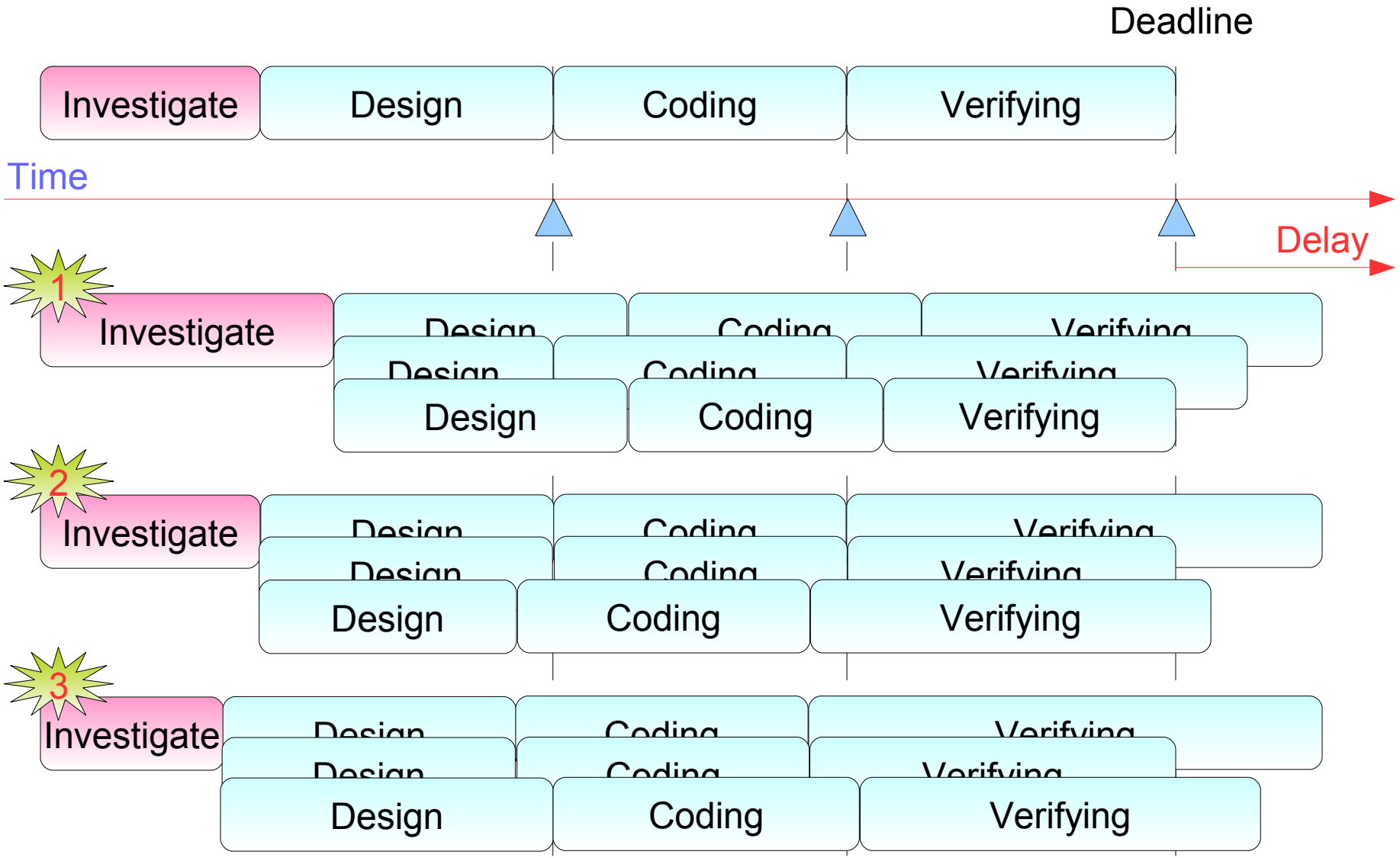
Introduction

- T.B.D
- I just want to share my experience to work effectively
- Just focus on normal model developing (with port/socket, register, ...)

Investigate input material (e.g: HWM)

- My experience based on MCS projects only. Other projects maybe different.
- The proposal is my applied way.

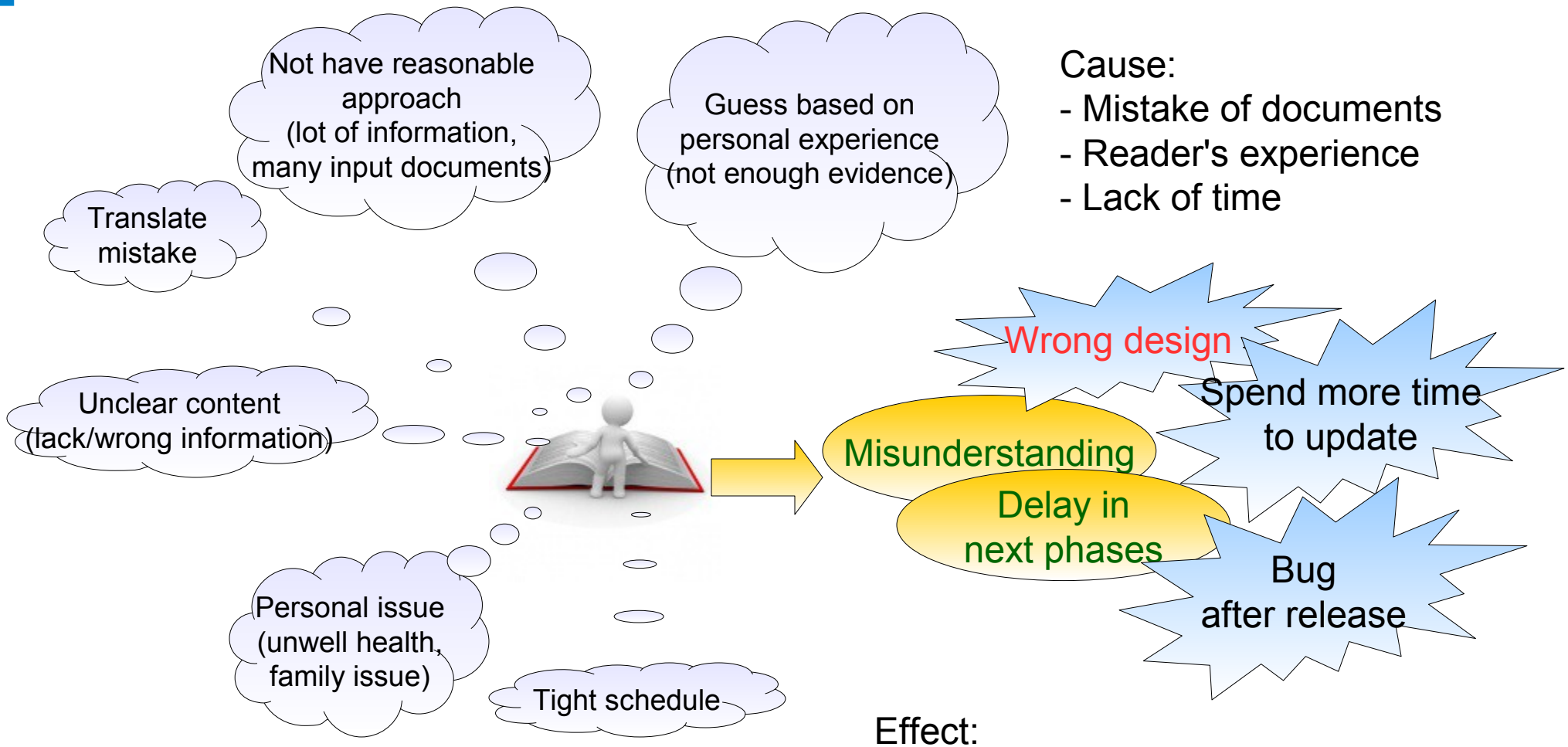
Face to issues ...



Face to issues ...

1. Real time for investigation is longer than expected plan:
    - Understand clearly: It effects to next phases (e.g: make delay) <= Wrong estimation
    - Misunderstand: It effects to next phases
        + Make delay
        + Spend more time to confirm, update, revise (depend on phase detects issue)
2. Real time for investigation is same as expected plan:
    - Understand clearly: It doesn't effect to next phase <= Best case
    - Misunderstand: It effects to next phases
        + Make delay
        + Spend more time to confirm, update, revise (depend on phase detects issue)
3. Real time for investigation is shorter than expected plan:
    - Understand clearly: It doesn't effect to next phase <= Wrong estimation
    - Misunderstand: It effects to next phases
        + Make delay
        + Spend more time to confirm, update, revise (depend on phase detects issue)

Cause and effect ...



Cause:
- Mistake of documents
- Reader's experience
- Lack of time

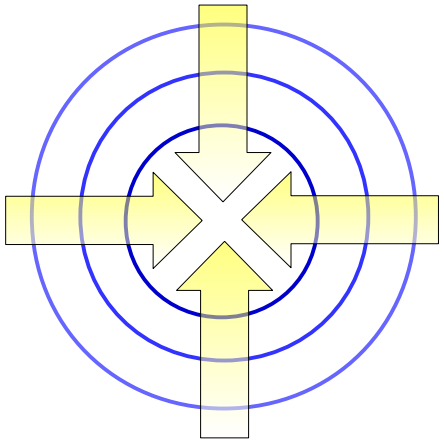
Effect:
- Delay in next phases
- Misunderstanding

How to investigate HWM effectively?

The effective investigation

Overall → Detail

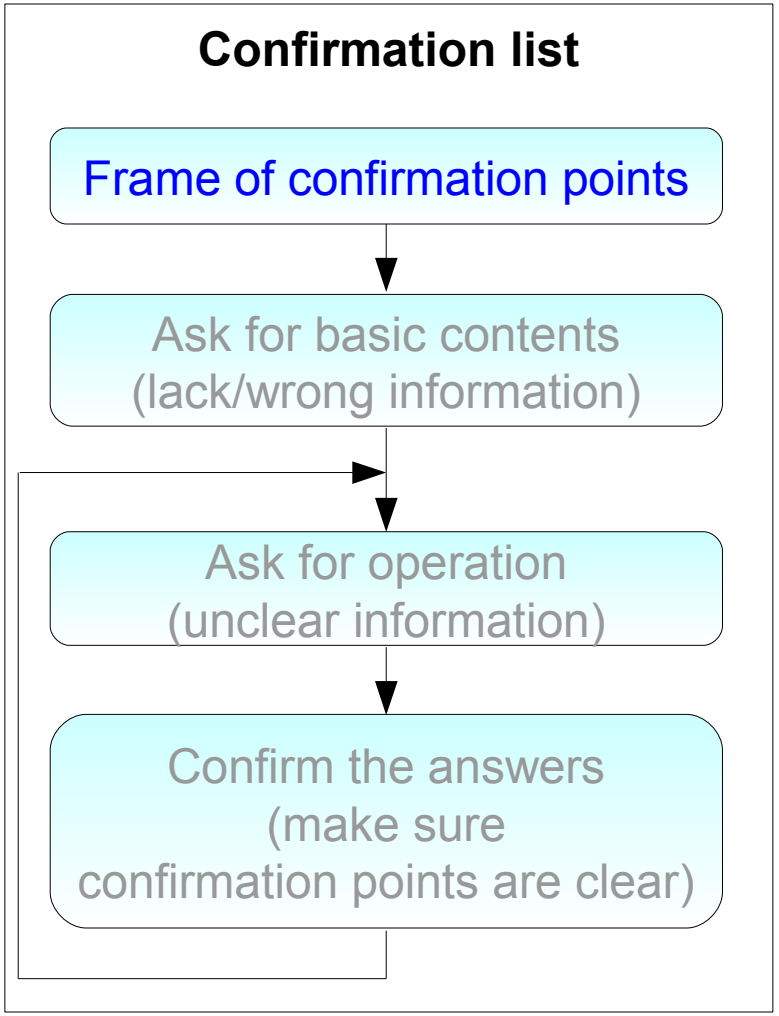
Width → Depth



Confirmation list

- Frame of confirmation points
- Ask for basic contents (lack/wrong information)
- Ask for operation (unclear information)
- Confirm the answers (make sure confirmation points are clear)

How to investigate HWM effectively?



(1) Get basic contents: port, register list

+ Do not read detail in port/register chapters (e.g: description)

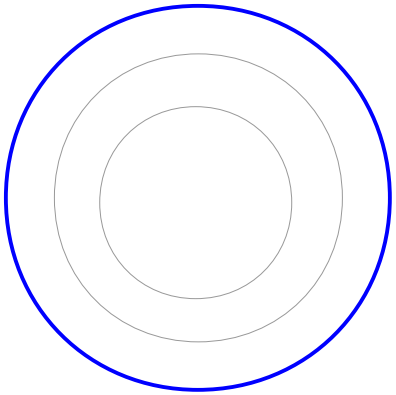
(2) Find reasonable approach

+ List features: (find the answer for below question)

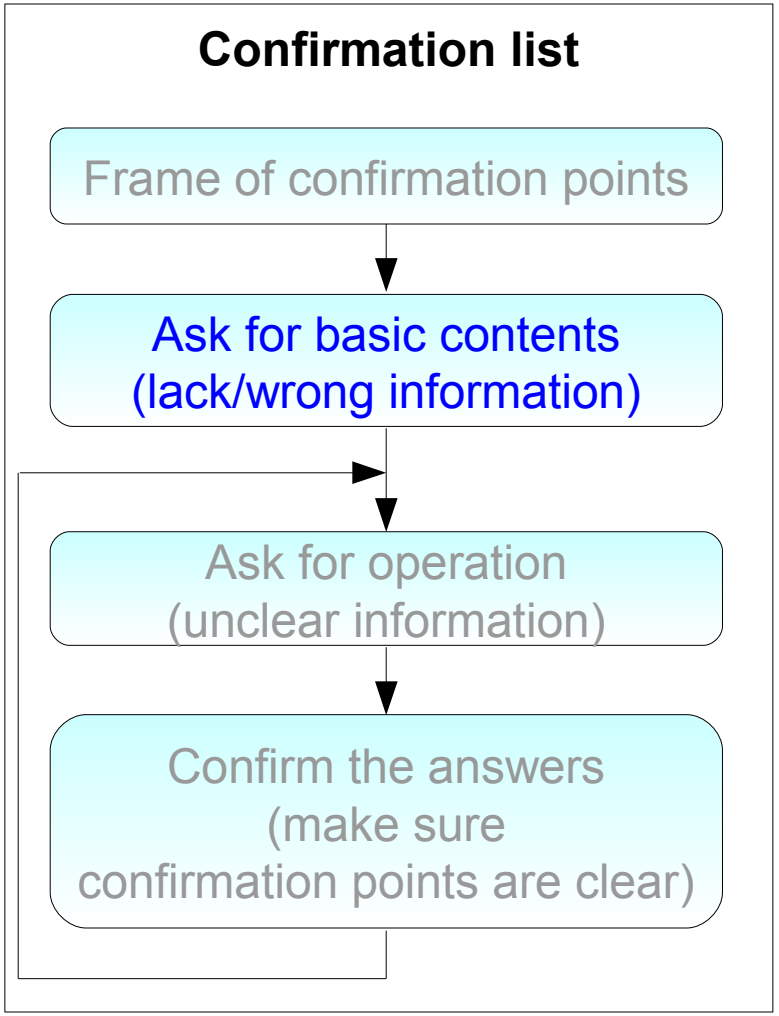
* What is the purpose this module?
* What is target application?

The answer can be found in "Overview/Introduction" chapters, ask others, search on internet, ...

+ Highlight unclear contents and write down question if any



How to investigate HWM effectively?



(1) Review unclear contents and try to find the answer by search related keys

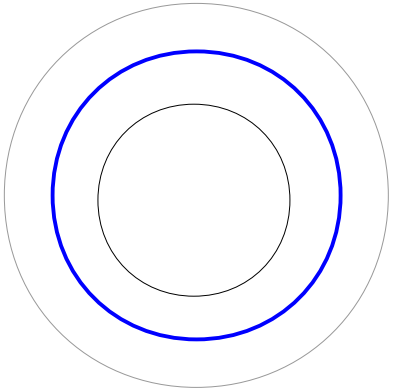
+ Check feature list: highlight unclear contents

+ Check description in port/register list: "what is this port/register used for?"

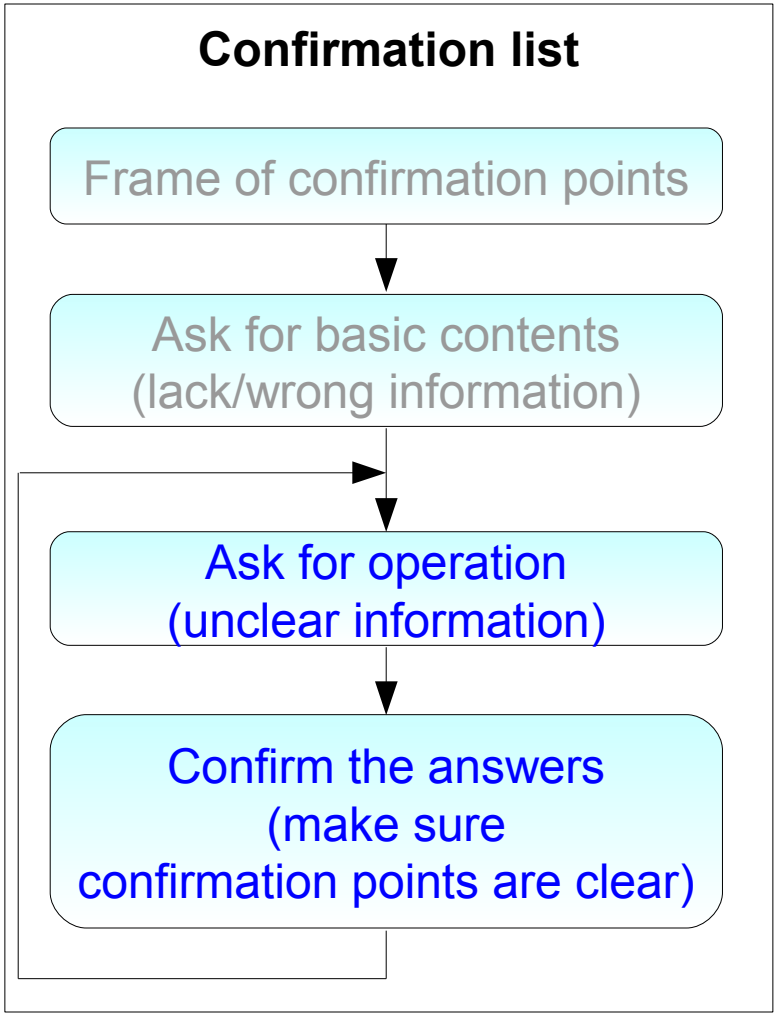
(2) Ask others and confirm their answer

+ Based on the answer, repeat (1)

(3) Ask customer if (1) and (2) are not satisfy

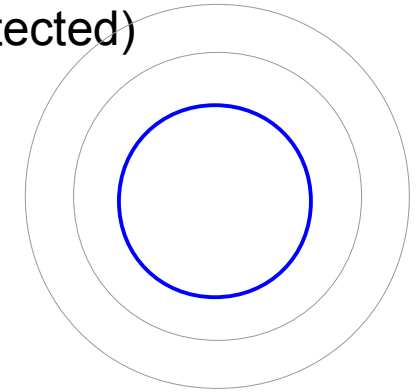


How to investigate HWM effectively?



(1) Understand operations
+ Find operation based on each feature
+ Highlight unclear contents and write down the question if any
(2) Ask others and confirm their answer
(3) Ask customer if (2) does not satisfy and confirm the answer. If the answer do not satisfy, update the question to re-confirm until all confirmation points are clear

Note: Above steps in this slide should be repeated during development process (if new issue/unclear point is detected)



Create documents

- My experience based on MCS projects only. Other projects maybe different.
- The proposal is my applied way.
- For document review, please refer to self-checklist
- For VRF-01
    + I will update template include basic test items (just title or feature)

Requirement (REQ)

- For model development in MCS projects, REQ already had a template (port/register list, feature, reset, timing, ...)
    + Get basic contents (after make clear via confirmation list): port, register, feature
    + Draw block diagram based on basic contents
    + Write detail function based on operation (after make clear via confirmation list)
    + Consider which parameter/command is supported
        * If model has interrupt signal(s) -> support parameter "Dump Interrupt"
        * If model has register for status -> support command "Dump Status Info"
        * Is there port/register can be supported by command/parameter?
        * Is there setting need to support by command/parameter?

Design document (INT)

- For model development in MCS projects, INT already had a frame for contents
    + Get information from REQ for basic chapters (port, register, feature, block diagram, parameter/command)
    + Draw operation diagrams based on supported function (describe in REQ), operation (make clear via confirmation list)
        * If detect more unclear point, continue add/update confirmation point via confirmation list (refer to slide 13 in detail)

Verification document (VRF-01)

- For model development in MCS projects, test items are created based on REQ, HWM and confirmation list (should not refer to INT, to keep independent mind for verifier and designer). If detect more unclear point, continue add/update confirmation point via confirmation list (refer to slide 13 in detail)
- VRF-01 includes test items (basic -> detail)
    + Test items for accessing register:
        (1) Check accessing all supported register (normal access)
            # Check reading, writing in valued bits (include read, write condition)
            # Check reading, writing in reserved bits
            # Check relationship between registers if any
            (e.g: there is 1 register control reading/writing of others)
        (2) Check accessing all supported register with wrong access size
        (3) Check accessing reserved area
            # Check "Reserved area + register area"
            # Check "Register area + reserved area"
            # Check "Register area + reserved area + register area"
            # Check "Reserved area + register area + reserved area"
        * Check accessing in debug mode (same as (1) and/or (3))
    + Test items for supported "Parameter/Command"

Verification document (VRF-01)

+ Test items for "Time Resolution"
+ Test items for reset operation:
    * Check software reset (by command), hardware reset (by port)
        # Reset in idle state
            % Model does not start yet (no setting) -> reset
            % Model started (setting some thing but not operate yet) -> reset
        # Reset in operation state
            % Model already operated -> reset
            % Reset at special points
            (e.g: when receiving data -> reset, when waiting time -> reset, when interrupt occurred -> reset, ...)
            Note: After reset a time, please check is model return to idle state or not (e.g: is there interrupt raised after reset?, ...)
        # Model operates as normal after reset
            % Reset -> setting and start operation of model
            % Model already operated -> reset -> setting and start operation of model
    * Check the interaction between hardware reset and software reset

Verification document (VRF-01)

+ Test items for operation when clock is zero:
    * Model does not operate/ stops operate
        # Not set clock -> try to start operation
        # Model already operated -> stop clock
    * Reset operation is active as normal
        # Not set clock -> reset
        # Model already operated -> stop clock -> reset
+ Test items for interrupt (if model have interrupt)
    * Make sure interrupt occurred with right condition
+ Test items for operation:
    * Check each supported feature (include right/wrong setting)
    * Check combination between features (with right setting)
    * Check start/stop operation => complex case
        # Reset multi-times: reset -> operate -> reset -> operate
        # If model have a register to control disable/enable operation, change setting of this register
        # (Stop clock ->) Set clock -> Stop clock -> Set clock
        (Note: cannot access register if the clock used for Bus is zero)
+ Test items for "Timing"

Verification document (VRF-01)

1) How can we have a common view at the beginning phase (make checklist)?

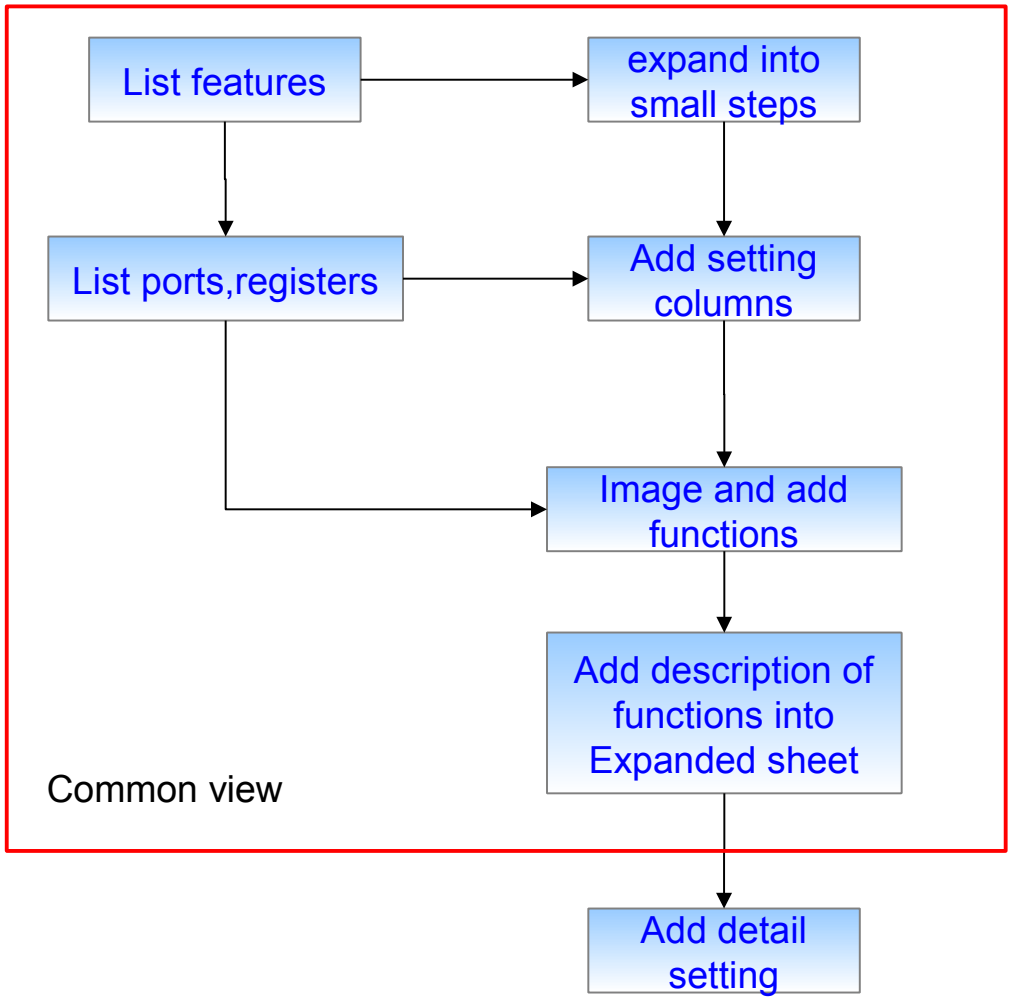
At the beginning phase, understand HWM is important!(*)

(*) - Understand HWM is not mean all clearly. That's just mean understand enough to list features. When understand more: (1). → Have more features → Update checklist;

(2). → Miss understand → Wrong features → Update checklist

- I reused 3 slides in "Sharing_GenTM.pdf" file which I shared to proposal "How to make a good checklist"
    # Easy for review
    # Clear category
    # Spend less time
    # Can have full test item with more corner case

Verification document (VRF-01)



1. List features

2.1. From each feature → expand into small steps

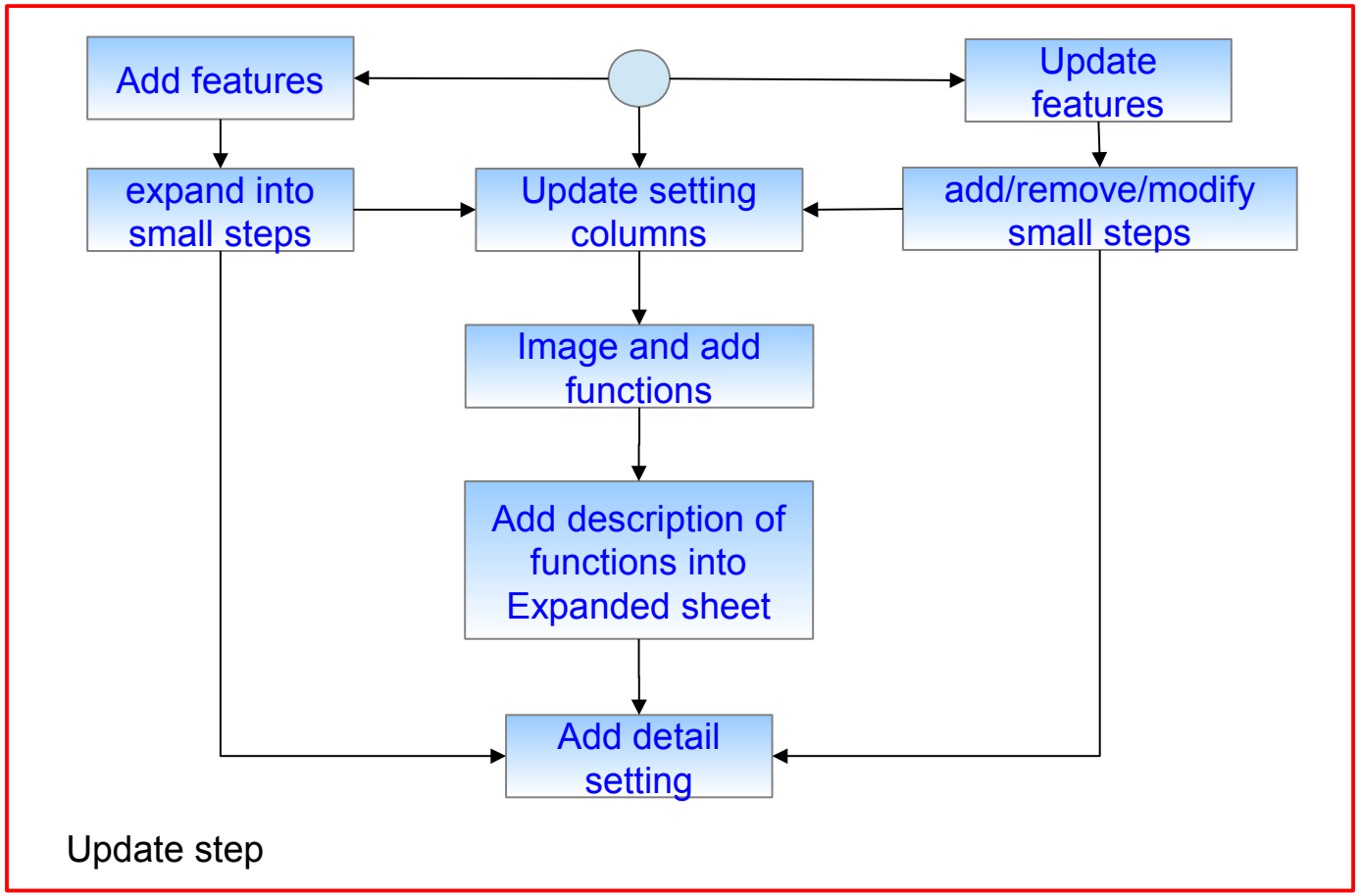
2.2. List all ports, registers → Consider to add setting columns (each port/register is 1 column or collect setting group into functions)

3. From each step → add setting columns

4. From setting column → image functions → add to Expanded sheet

5. Add detail setting

Verification document (VRF-01)



Update step

Code implementation

- My experience based on MCS projects only. Other projects maybe different.
- The proposal is my applied way.

How to ...

- For model development in MCS projects, source code is created based on REQ, HWM and confirmation list, INT. If detect more unclear point, continue add/update confirmation point via confirmation list (refer to slide 13 in detail)
- Coding includes steps (simple -> complex)
    + Reuse some common functions (ConvertClockFreq, CheckClockPeriod, CalculateCLKPosEdge, SeparateString, GetTimeResolution) (1*)
    + Define functions for supported parameters and commands (refer to command ans parameter list in REQ)
        # Declare argument for parameter in constructor
        # Some functions can be reused from previous models (2*)
    + Declare event, method to handle clock, reset, interrupt (refer to previous model, HWM, confirmation list) (3*)
    + Use tool to generate registerIF and CommandIF classes (we must prepare ..if.txt files of course) (refer to register list in REQ)

Note: (1*), (2*), (3*) - I expected we can use tool to generate (I already started to do this)

How to ...

+ Implement for register callback functions (refer to HWM, confirmation list, REQ)
    # Relationship between registers if any
    (e.g: there is 1 register control reading/writing of others)
    # Affection when the register is written (based on description in HWM)
+ Implement function for each feature (based on basic idea in INT)
    # Consider condition for "reset or zero clock" checking
    # Refer to HWM to review feature to make sure idea in INT is useful

Code review

- My experience based on MCS projects only. Other projects maybe different.
- The proposal is my applied way.
- If the proposal is useful, will we add it into self-checklist?

How to ...

- Verifier should join reviewing (in some project, due to lack of resource, reviewer is not verifier)
    + Misunderstand between designer and verifier can be detected (as much as possible)
        # If detect more unclear point, continue add/update confirmation point via confirmation list (refer to slide 13 in detail)
    + Designer and verifier should have unify mind
- Reviewer should understand basic operation

Note: Mismatch between source code and design documents (INT, REQ) can be detected too

How to ...

- Review contents (basic -> detail)
    + Register list (refer to HWM, REQ, INT)
        # Check register defining
            * Wrong name, access size, bit initial value, access role?
            * Lack of register?
        # Check callback function of register
            * Lack of callback? (there is no callback function for register needs)
            * Odd callback? (there is unnecessary callback function)
            * Wrong/misunderstand implementation?
    + Command/parameter (refer to REQ, INT, Confirmation list)
        # Command
            * Lack/Odd of command?
            * Wrong/misunderstand implementation?
        # Parameter
            * Lack of its definition in constructor?
            * Does it NOT effect to model?
            (there is no argument assigned by its value)
            * Is argument NOT updated by its value after reset/restart model?
            (after reset, the argument is not reset to parameter value but zero value)

How to ...

- Review contents (basic -> detail) *(cont)*
    + Argument definition
        # Odd argument? (there is unused argument) <- can detect by 1Team check!?!
    + Operation (refer to INT, HWM, Confirmation list)
        # Check start/stop operation
        # Check reset operation (initial value)
        # Check operation flow of each feature
            * Wrong/misunderstand implementation?
            * Lack of condition for "reset or zero clock" checking?
            (e.g: model does not change state when reset is active or clock is zero)
            * Different with flow in INT?
        # Check timing
            * Wrong formula?
        # Check optimization
            * Can optimize for long code?
    + Check description (comments): suitable, clear
    + Check grammar, typo, tab/space
- Compare source code when it updated with odder

Functional debug

- My experience based on MCS projects only. Other projects maybe different.
- The proposal is my applied way.

How to ...

- Compare source code when it updated with odder. If necessary (source code is updated so much), repeat "Code review"
- Execute all test items
    + Check "PASS" test items
        # Make sure right setting
        # Make sure right operation
        # Re-execute all "PASS" test items to make sure updated source code does not effect to previous results

How to ...

+ Check "FAIL" test items (1 by 1)
    # Confirm failed case (by view of verifier)
        (1) Wrong test item?
        (2) Wrong environment configure?
        (3) Unexpected/Wrong implementation?
    # If detect more unclear point between verifier and designer, continue add/update confirmation point via confirmation list (refer to slide 13 in detail).
    DO NOT confirm face to face.
    # Find root cause
        * If (1) or (2), verifier need to update setting/configure and re-execute
        * If (3), verifier post issue on Redmine system
            $ Symptom
            $ Expected
            $ Cause (if any)
            $ Suggestion (if any)
            $ Environment and TM fail (if any) <= I didn't suggest this because I think the "symptom and expected" are enough for designer can detect issue by review their coding operation. In the case complex TM, verifier need to spend more time to prepare another with more simple for designer debug.

Prepare released data

- My experience based on MCS projects only. Other projects maybe different.
- The proposal is my applied way.

How to ...

- In MCS's project, we have clean_all.csh file used to clean environment
- Below step is necessary for preparing data (step by step):
    + Make sure last execution with latest source code
        # Is all source code committed CVS?
        # Is committed source code with wrong version?
        # Last check for description/comments (suitable, clear), grammar, typo, tab/space (Do not need to repeat "Code review" due to "Functional debug" already done)
        # Is there "DEBUG" message?
    + Make sure verify component is removed "DEBUG" message
    + Make sure all expected logs are PASS (no TM FAIL or cannot finish execution)
        # Is "PASS" TM number same as total TM (in each mode)?
        # Is **execution result** expected? (PASS TM, coverage result, summarize report)
    + Clean environment by using clean_all.csh file
        # Clean object/built files (gmake clean)
        # Clean folders need to empty or copied files
        # Clean "log" files used for debug
        # Clean coverage files (coverage result already copied to report folder)

How to ...

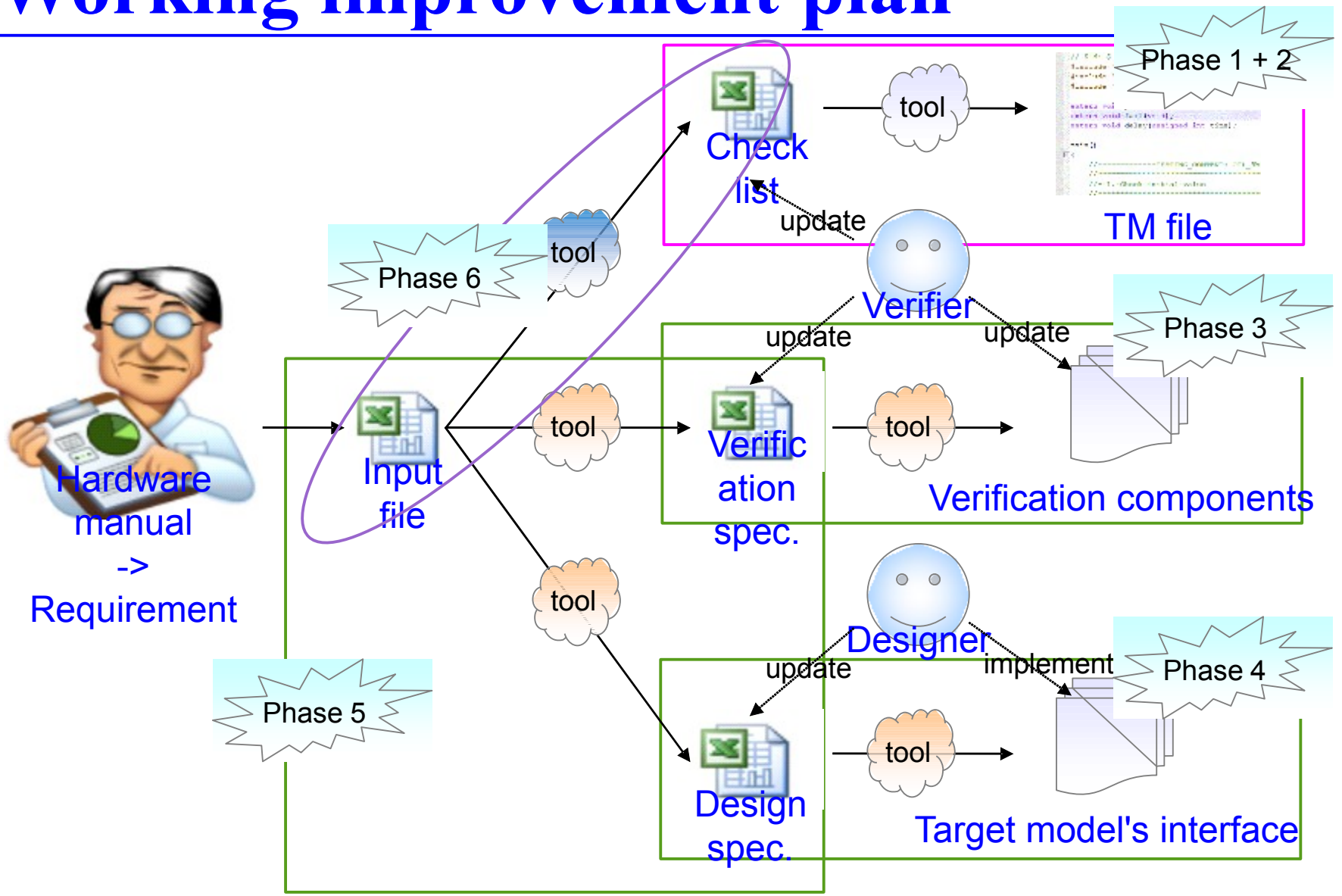
+ Make sure readme.txt file is last update
    # Test item number, pass/fail number
    # Model name
    # Description for setting
+ Check execution files (run_all*.csh/run_all*.bat)
    # Is there line for execution be commented?
    # Is there line for coverage be not commented? (we should comment line for coverage)
+ Compress data
+ Checking data (reviewer should be a other)
    # Make sure data packet can execute as normal
    # Make sure execution result is same as expected
    # Re-check readme.txt, execution files as above steps

- That's my working check list (excel file is better for checking in each phase)
- If we can collect more ideas, working flow can be improved.

Please share your ideas!

Renesas Design Vietnam Co., Ltd.

Working improvement plan



Action item

- Update template of checklist.	P.I.C: T.B.D	D/L: T.B.D
- Update self-checklist.	P.I.C: T.B.D	D/L: T.B.D
- Create tool to check automatically	P.I.C: T.B.D	D/L: T.B.D
- ...		