Software Product Assurance

Block 2 – Software PA Organizational Aspects / SW RAMS

Why SW PA?

Project Manager

«I want the SW "ready" in time and within budget»

Software Engineer

«I want to see my SW "work"»

Software PA

«I want to see the SW:

- Perform correctly in all foreseen scenarios
- Perform correctly on all foreseen platforms
- Be reliable
- Be robust
- Be maintainable
- Fulfil quality requirements
- … »

Product Assurance

*Discipline devoted to the study, planning and implementation of activities intended to **assure** that the design, controls, methods and techniques in a project result in a satisfactory degree of **quality** in a **product*** *[ECSS-S-ST-00-01]*

How SW PA?

- Apply requirements meant to ensure the quality of processes and products
- Those requirements are defined in Standards
- ESA applies ECSS ⇨ **ECSS-Q-ST-80**

- Standards' requirements are to be tailored based on criteria related to the specific project
- ECSS-Q-ST-80 includes a pre-tailoring based on software criticality (see later)

What is NOT SW PA

- Verification/Validation
- Testing
- Configuration Management
- Risk Management

Customer and Supplier

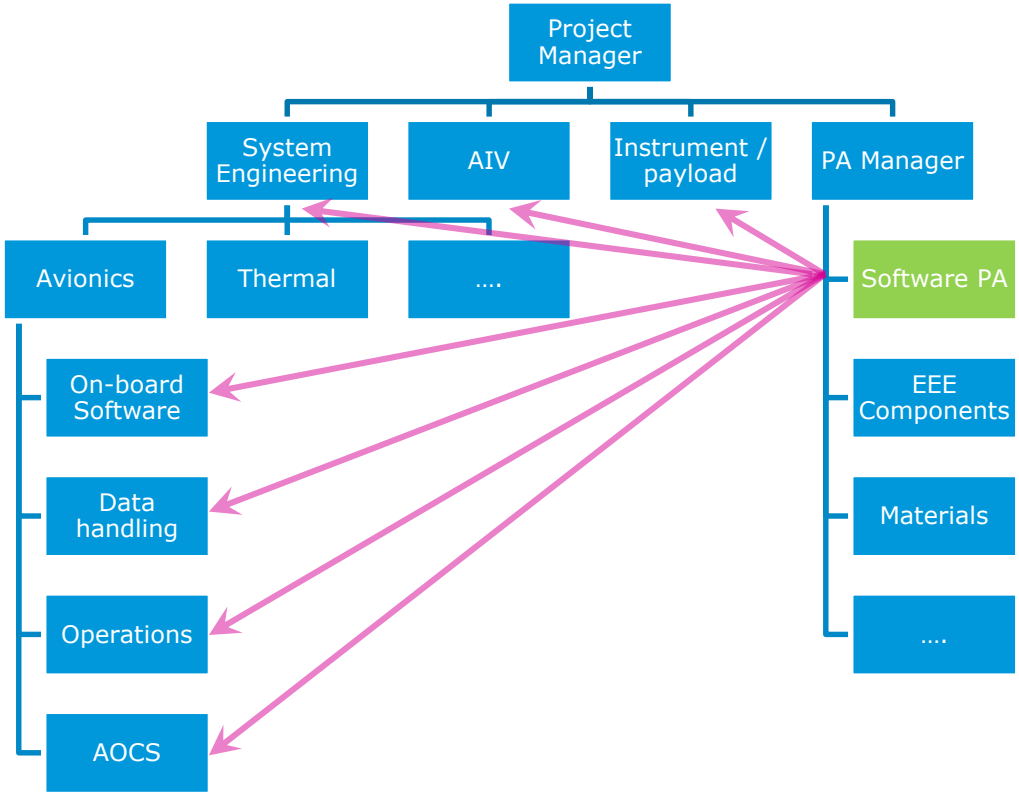
- Customer-supplier relationship, typically applied recursively (customer-supplier chain)
  - Intermediate chain levels: often both customer and supplier
- SW PA at customer level
  - Ensures suitability of procurement documentation
  - Defines software product assurance requirements
  - Monitors the suppliers' conformance to SW PA requirements

Customer

Supplier

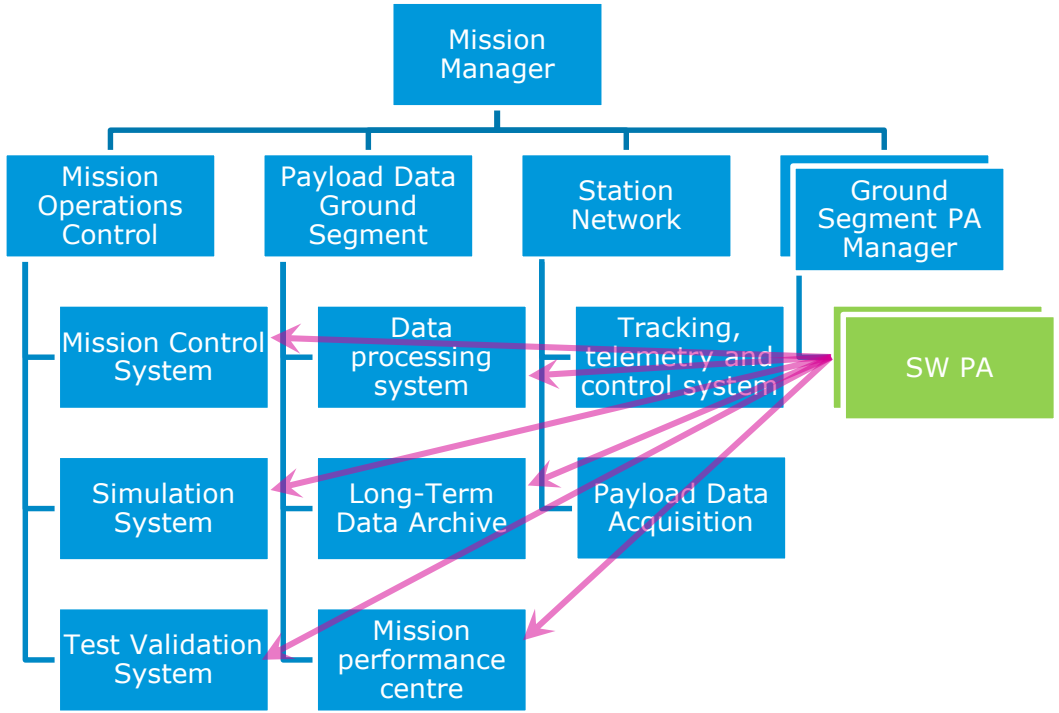
- SW PA at supplier level
  - Ensures correct implementation of software product assurance requirements
  - Defines a software product assurance programme
  - Reports to customer about implementation software product assurance programme

SW PA in a Space Segment Project



[simplified]

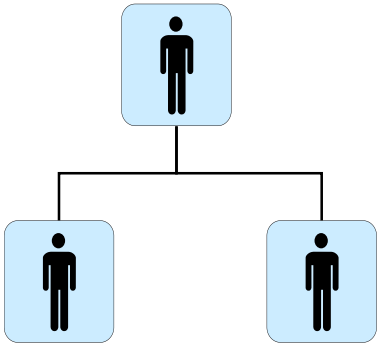
SW PA in a Ground Segment Project



[simplified]

SW PA Organization

- Software suppliers are required to:
- Define an organizational structure for software development
  - Not only PA: all personnel whose work affects quality
- Allocate and make available resources for the SW PA tasks
- Identify personnel in charge of SW PA tasks
  - Software Product Assurance Manager (or Engineer)
- Ensure authority and independence of SW PA in charge
- Grant unimpeded access to higher management



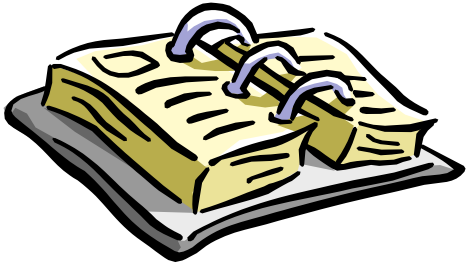
Training

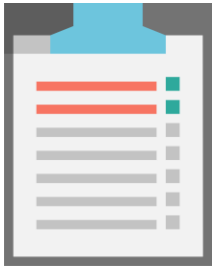
- Ensure that the right composition and categories of appropriately trained personnel are available
- Determine training subjects based on the specific tools, techniques, methodologies and computer resources to be used

- No university degrees in Software Product Assurance around
- Build up SW PA skills through training, experience in SW development and PA in general

SW PA Planning

- Develop a Software Product Assurance Plan in response to applicable software product assurance requirements
  - May be part of the overall project PA plan
  - Not necessarily a tome: only what is realistically feasible
- Ensure Plan is up-to-date at each milestone





- Include a compliance matrix vs. the applicable software product assurance requirements
- Include references to the project documentation that will contain the output of the implemented requirements

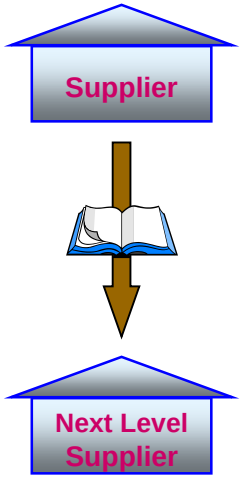
SW PA Reporting

- Regular software product assurance reporting to be provided as part of the overall project reporting
- Specific reporting to be provided at milestone reviews

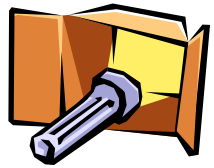
- Main reporting topics
  - Assessment of product and process quality
  - Verifications undertaken
  - Problems detected and resolved

Supplier Requirements and Monitoring

- PA should be involved in the selection of lower-level suppliers
- When selecting lower-level suppliers that claim (massive) software reuse, a preliminary software reuse file (see later) should be required as part of the proposal



- Software product assurance requirements shall be established for lower-level suppliers
  - To be approved by the customer
- Lower-level suppliers shall be monitored
  - Approve SW PA plan
  - Verify definition and implementation of software development processes, in accordance with SW PA requirements, and quality of products

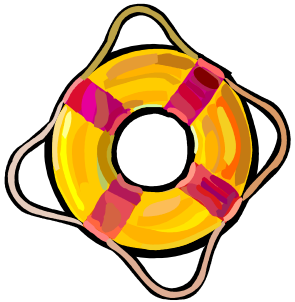


SW Dependability and Safety

- Software RAMS
  - Reliability
  - Availability
  - Maintainability
  - Safety

} Dependability

Software RAMS activities start at system level and continue at software level, with mutual feedback



- Main objectives
  - Classify software based on criticality
  - Define and implement measures to handle critical software (including pre-tailoring)

ECSS-Q-HB-80-03
Software Dependability and Safety

Software failures and faults

Human mistake

```
if(1)
{
   x++;
}
```

(Software) Fault

```
try{...}
catch(error)
{...}
```

(Software) Error

(System) Failure

- Software is a purely intellectual artefact
  - Behaves as programmed
- Do software failures exist?
- Software faults, hence software-caused failures, are systematic
  - No hardware-like wear-out
- Software-caused failures occur randomly
  - Under specific conditions
  - Difficult to predict (much like hardware)

Software Reliability

- Property of being "free from faults"
- Achieved through a set of activities at system and at software level
- Software reliability requirements are derived from system ones
- Compliance with software quantitative requirements can hardly be demonstrated
  - Software reliability models exist
  - Based on assumptions that have proven to be unjustified for most of bespoke software

Software Availability & Maintainability

- Maintainability: capability of the software to be retained or restored to a state in which it can perform a required function, when maintenance is performed
- Especially important for SW with long lifetime

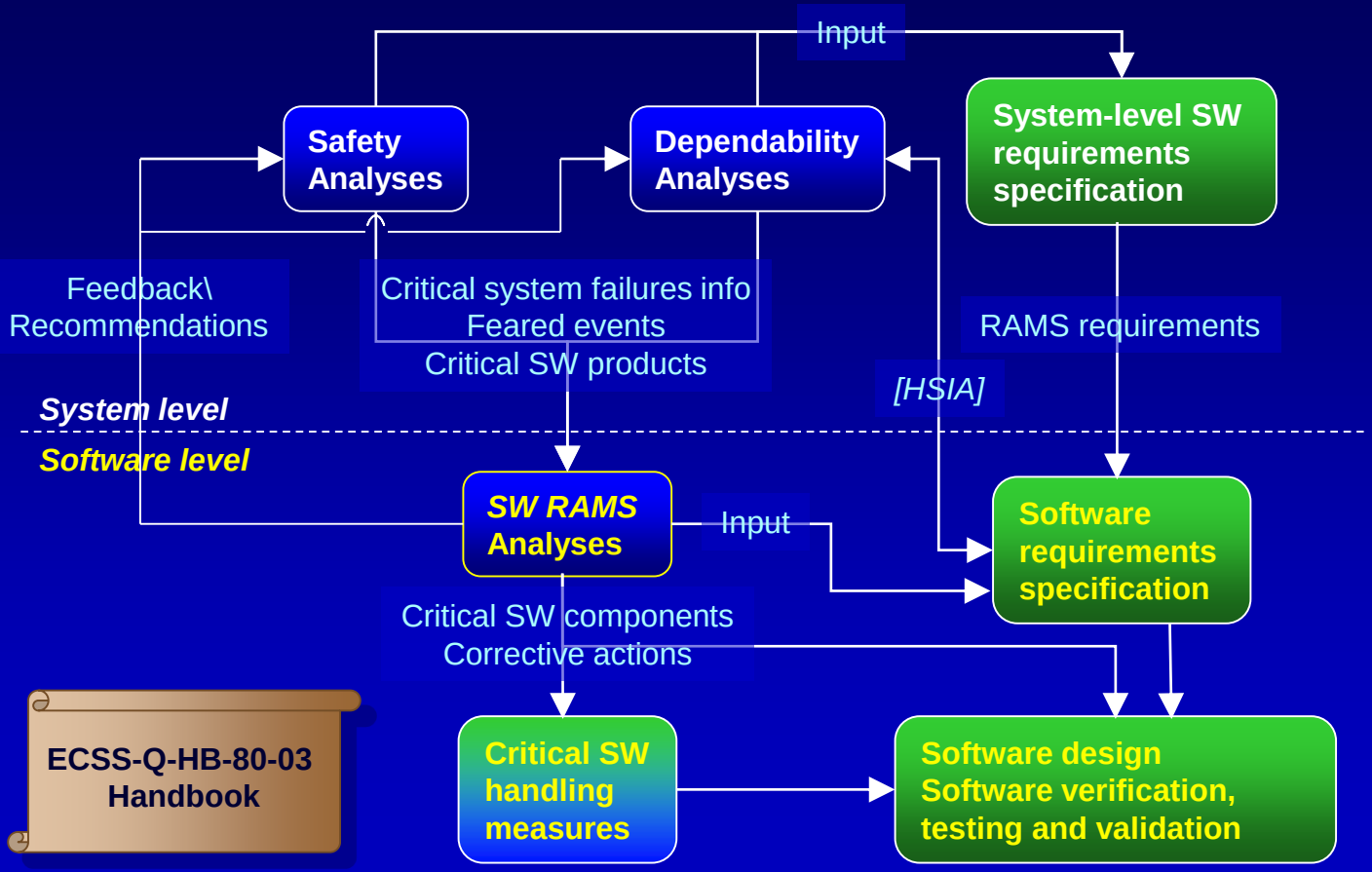
- Availability: capability of the software to perform its function at a given instant or for a time interval
  - It is a function of reliability and maintainability

Software Safety

- Safety is a system property
  - Software in itself cannot cause or prevent harm to human beings, system loss or damage to environment
- Safety and reliability are different concepts
  - A system can be reliable but not safe, and vice-versa
- Software safety is the contribution of software to the system safety

- Compliance of software with numerical safety targets cannot be analytically demonstrated
- Approach: design for minimum risk

Software RAMS overview



Software Criticality Classification

- At system level:
  - Dependability analyses (e.g. FME(C)A, FTA)
  - Safety analyses (e.g. Hazard Analysis)

Severity classification of failures/hazardous events

Criticality classification of system functions

Criticality classification of hardware and operations

Consideration of compensating provisions

Criticality classification of **software products**

Function Criticality Classification

- Function criticality is directly linked to the severity of failure/hazard consequences, without consideration of compensating provisions

SEVERITY	***FUNCTION CRITICALITY***	***CRITERIA TO ASSIGN CRITICALITY CATEGORIES TO FUNCTIONS***
CATASTROPHIC (LEVEL 1)	**I**	A FUNCTION THAT IF NOT OR INCORRECTLY PERFORMED, OR WHOSE ANOMALOUS BEHAVIOUR CAN CAUSE ONE OR MORE FEARED EVENTS RESULTING IN CATASTROPHIC CONSEQUENCES
CRITICAL (LEVEL 2)	**II**	A FUNCTION THAT IF NOT OR INCORRECTLY PERFORMED, OR WHOSE ANOMALOUS BEHAVIOUR CAN CAUSE ONE OR MORE FEARED EVENTS RESULTING IN CRITICAL CONSEQUENCES
…	…	…

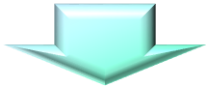
SW Dependability and Safety

NAME	LEVEL	DEPENDABILITY (ECSS-Q-30)	SAFETY (ECSS-Q-40)
CATASTROPHIC	1		• LOSS OF LIFE, LIFE-THREATENING OR PERMANENTLY DISABLING INJURY OR OCCUPATIONAL ILLNESS. • LOSS OF AN INTERFACING MANNED FLIGHT SYSTEM • SEVERE DETRIMENTAL ENVIRONMENTAL EFFECTS. • LOSS OF LAUNCH SITE FACILITIES. • LOSS OF SYSTEM
CRITICAL	2	COMPLETE LOSS OF MISSION	• TEMPORARILY DISABLING BUT NOT LIFE-THREATENING INJURY, OR TEMPORARY OCCUPATIONAL ILLNESS . • MAJOR DETRIMENTAL ENVIRONMENTAL EFFECTS. • MAJOR DAMAGE TO PUBLIC OR PRIVATE PROPERTIES. • MAJOR DAMAGE TO INTERFACING FLIGHT SYSTEMS, • MAJOR DAMAGE TO GROUND FACILITIES.
MAJOR	3	MAJOR MISSION DEGRADATION	
MINOR OR NEGLIGIBLE	4	MINOR MISSION DEGRADATION OR ANY OTHER EFFECT	

Failure/hazard consequences severity categories

HW/SW Products Criticality

- Criticality of hardware and operations is determined in accordance with the highest criticality of functions implemented
- Criticality of software is assigned considering the overall system design



- In particular whether compensating provisions exist that can prevent or mitigate failure consequences (e.g. inhibits, monitors, back-ups, operational procedures)
- Compensating provisions allow to "*downgrade*" the software criticality (of 1 category only)

Software Criticality Categories (I)

FUNCTION CRITICALITY	CRITICALITY CATEGORY TO BE ASSIGNED TO A SOFTWARE PRODUCT
I	CRITICALITY CATEGORY **A** IF THE SOFTWARE PRODUCT IS THE SOLE MEANS TO IMPLEMENT THE FUNCTION
	CRITICALITY CATEGORY **B** IF, IN ADDITION, AT LEAST ONE OF THE FOLLOWING COMPENSATING PROVISIONS IS AVAILABLE, MEETING THE REQUIREMENTS DEFINED IN CLAUSE 5.4.2: - A HARDWARE IMPLEMENTATION - A SOFTWARE IMPLEMENTATION; THIS SOFTWARE IMPLEMENTATION SHALL BE CLASSIFIED AS CRITICALITY **A** - AN OPERATIONAL PROCEDURE
II	CRITICALITY CATEGORY **B** IF THE SOFTWARE PRODUCT IS THE SOLE MEANS TO IMPLEMENT THE FUNCTION
	CRITICALITY CATEGORY **C** IF, IN ADDITION, AT LEAST ONE OF THE FOLLOWING COMPENSATING PROVISIONS IS AVAILABLE, MEETING THE REQUIREMENTS DEFINED IN CLAUSE 5.4.2: - A HARDWARE IMPLEMENTATION - A SOFTWARE IMPLEMENTATION; THIS SOFTWARE IMPLEMENTATION SHALL BE CLASSIFIED AS CRITICALITY **B** - AN OPERATIONAL PROCEDURE

Software Criticality Categories (II)

FUNCTION CRITICALITY	***CRITICALITY CATEGORY TO BE ASSIGNED TO A SOFTWARE PRODUCT***
III	CRITICALITY CATEGORY **C** IF THE SOFTWARE PRODUCT IS THE SOLE MEANS TO IMPLEMENT THE FUNCTION
	CRITICALITY CATEGORY **D** IF, IN ADDITION, AT LEAST ONE OF THE FOLLOWING COMPENSATING PROVISIONS IS AVAILABLE, MEETING THE REQUIREMENTS DEFINED IN CLAUSE 5.4.2: - A HARDWARE IMPLEMENTATION - A SOFTWARE IMPLEMENTATION; THIS SOFTWARE IMPLEMENTATION SHALL BE CLASSIFIED AS CRITICALITY **C** - AN OPERATIONAL PROCEDURE
IV	CRITICALITY CATEGORY **D**

NOTE: IT SHOULD BE NOTED THAT A TOO HIGH LEVEL/INCOMPLETE FUNCTIONAL DECOMPOSITION, POORLY ACCOUNTING FOR SAFETY AND DEPENDABILITY ASPECTS, COULD LEAD TO A UNNECESSARILY CONSERVATIVE SOFTWARE CATEGORY CLASSIFICATION.

Compensating Provisions

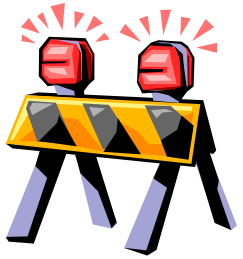
- Conditions are established for acceptable compensating provisions in the SW criticality assignment
  - Probabilistic assessment cannot be used as a criterion for SW criticality classification
  - Effectiveness of compensating provisions (for the purpose of "downgrading") must be demonstrated in all conditions
  - There must be sufficient time to intervene in all situations
  - In case the compensating provisions contain software, this software shall be classified at the criticality category corresponding to the highest severity of the failure consequences that they prevent or mitigate

SW Dependability and Safety

- The supplier shall apply engineering measures to reduce the number of critical software components
- Propagation of failures from low-criticality to high-criticality SW components shall be prevented
  - If not possible, all involved components shall be classified at the highest criticality level among them
- Contribution of software to Hardware-Software Integration Analysis
  - Identify, for each hardware failure included in the HSIA, the requirements that specify the software behaviour in the event of that hardware failure

Handling of Critical Software (I)

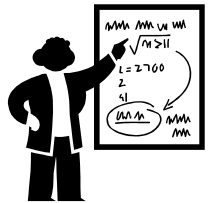
- The supplier shall define, justify and apply measures to assure the dependability and safety of critical software
  - Measure proposed by the supplier and agreed with the customer, e.g.:
  - insertion of features for failure isolation and handling;
  - defensive programming techniques;
  - use of a “safe subset” of programming language;
  - full inspection of source code; etc.



- The correct implementation of the chosen measures shall be verified and reported on

Handling of Critical Software (II)

- Specific requirements for critical software
  - Mandatory regression testing in case of change of hardware or development tools
  - Potential need for additional verification and validation to be analysed in case of change of hardware and environment
  - Remove unreachable code
  - Testing to be (re-)executed on non-instrumented code



Besides the tailoring of engineering and PA requirements

Tailoring of SW PA requirements

- For most projects, making all ECSS-Q-ST-80C requirements applicable is neither sensible nor feasible
  - … and supplier claiming compliance is not credible
- SW PA requirements should always be tailored to the specific project's needs
  - Tailoring is a customer's responsibility!
- Different tailoring drivers may (co-)exist
  - Dependability and safety aspects
  - Software development constraints
  - Product quality objectives and business objectives
- In general, budget should not be the main driver

Pre-tailoring based on criticality

Clause	Description	A	B	C	D
…	…	…	…	…	…
7	**Software product quality assurance**	**-**	**-**	**-**	**-**
7.1	Product quality objectives and metrication	-	-	-	-
7.1.1	Deriving of requirements	Y	Y	Y	Y
7.1.2	Quantitative definition of quality requirements	Y	Y	Y	Y
7.1.3	Assurance activities for product quality requirements	Y	Y	Y	Y
7.1.4	Product metrics	Y	Y	Y	Bullet 4.(a) not applicable
7.1.5	Basic metrics	Y	Y	Y	Design-relevant and fault density/failure intensity metrics not required
7.1.6	Reporting of metrics	Y	Y	Y	Y
7.1.7	Numerical accuracy	Y	Y	Y	Y
7.1.8	Analysis of software maturity	Y	Y	Y	N
…	…	…	…	…	…

Disclaimer