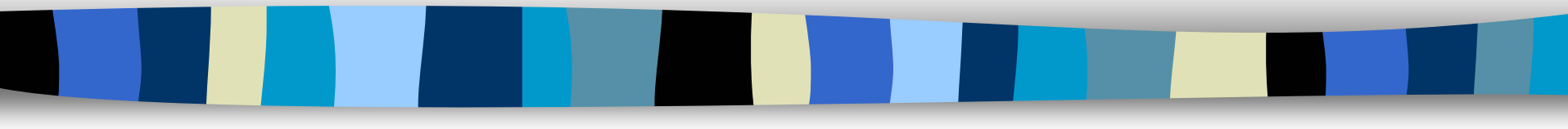
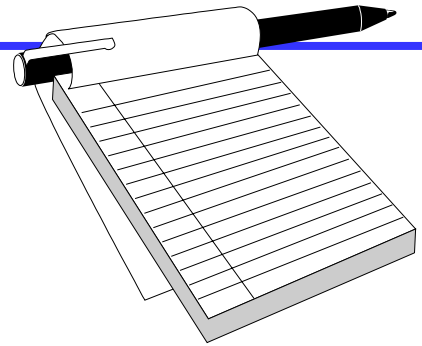
Lessons Learnt in Software Projects Review Processes

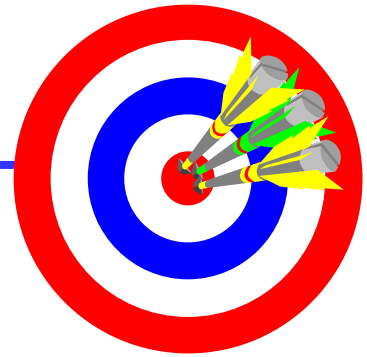


Agenda



- Objective
- Overview: Review perspective
- TOP Faults of the sequential Review Processes (and life cycle):
- Consequences
- TOP Generic Faults
- Generic Recommendations
- Conclusions

Objective



WHAT:

Contribute to improve future software processes and products from lessons learnt in previous reviews.

HOW :

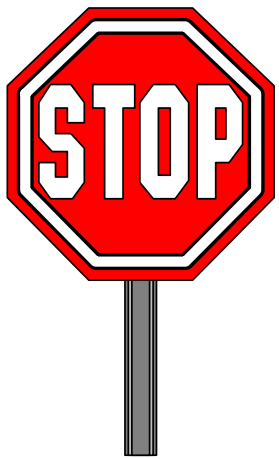
Examining Reviews Outputs

Selecting Recomendations.

(Criteria: generality, criticality, repetitive)

Sw Projects seen from Review Perspective

Technically: OK



Deficiency in **application** of methods, principles, guidelines

Planification: TOP 4 Faults

Items	Contingency actions
Focus on scientific tasks	Describe and plan sw engineering tasks (including SQA and detailed GANTT)
Mgment activities poorly described and planned	Specify procedures, activities, internal reviews, responsibilities, conflict resolution.
Risks are identified but no contingency actions foreseen.	Perform a complete risk management plan
CM activities start too late	CM activities must start at the beginning of the life cycle with the definition of all configuration items.

Requirements: TOP 4 Faults

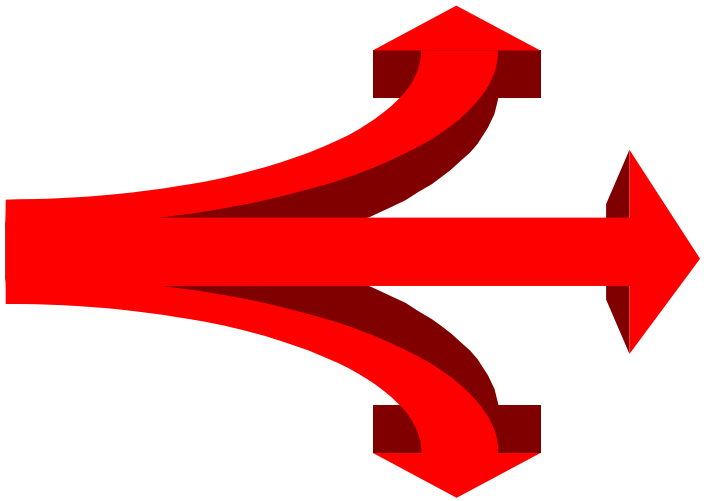
Items	Contingency actions
No products definition	Identify products (obj, results, req., etc.)
No definition of product quality level	Define product quality level taken into account type of products, constraints, sources, dependencies, incompatibilities, etc.
Non measurable requirements	Avoid “as much as possible”, “at the earliest”, etc. Use measures
No distinction of users (if applicable)	Distinguish different users or user interface (e.g. via ftp, direct request, distribution list, etc.)

Arch. Design: TOP 4 Faults

Item	Contingency actions
Non-functional reqs are not implemented	Take into account performance, operational, resource, security, maintainability, quality, reliability, etc. reqs
No sw & data errors mgment and system failure mgment	Analyse and design errors and faults management
No reasoned decision on method and derived issues	Be consistent among design methods, notations, tool and programming language
No verification against SRD	Check consistency

Consequences

Uncompleted plan --> Delays, uncontrolled project...



Implementation starts with no complete and clear definition --> No expected results

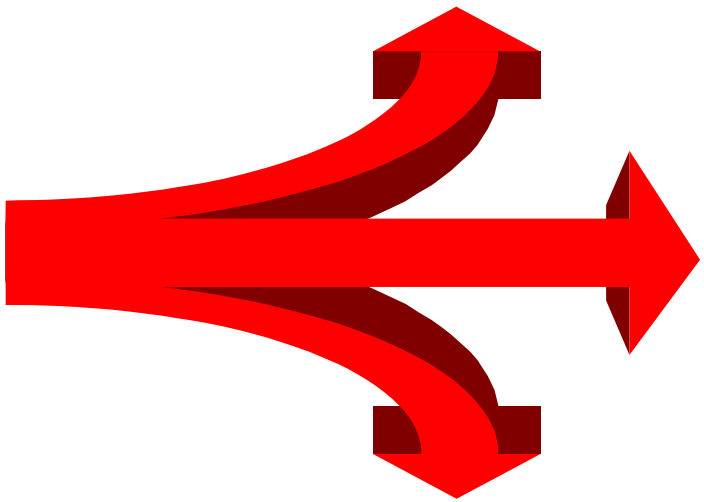
No best quality in products

DD and Unit Testing: TOP Faults

Item	Contingency actions
Uncompleted list of configuration items	Include all sw config items, hw and COTS if necessary
No test coverage defined	Define invalid & valid classes, border cases, ...
No testing responsible defined	Define a no programmer participant responsible

Consequences

Products not fully validated



Undefined configurations -->
No possible versions recovery

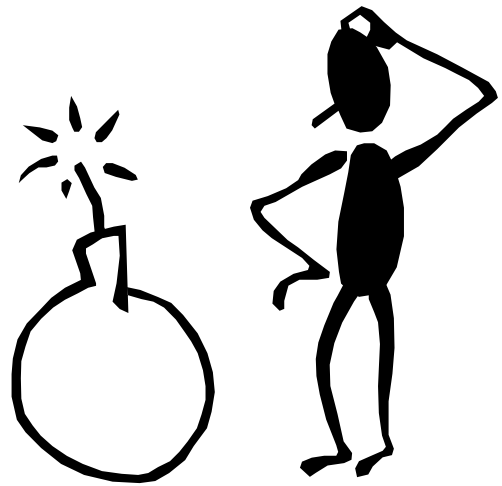
No control on items evolution

V&V TOP 4 Faults

Item	Contingency actions
Impossible to compare expected results with observed results	Focus on visibility of results and its correctness
Impossible to trace tests	Include traceability matrices (to check against reqs)
Sw robustness is not tested	Define test to stress the system, introduce errors, etc.
No definition of validation environment	Define environment, conditions, emulators if nec., etc.

Consequences

No reliable products

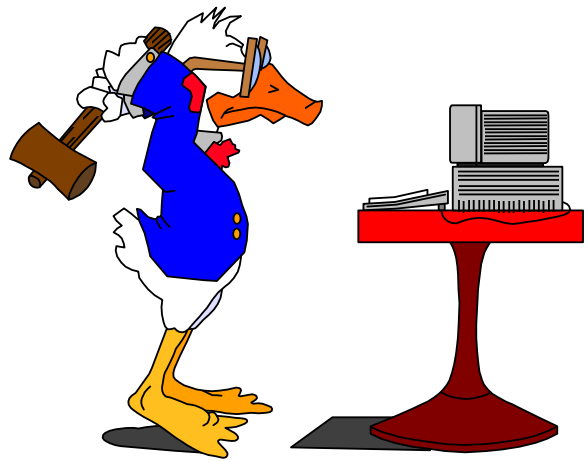


User Manual TOP 4 Faults

Item	Contingency actions
Different users not identified	Define users profiles and distinguish operations
Incomprehensibility	To be reviewed by different users
Too narrative	Include screens, graphics, tables, etc.
Partner-dependent style	Review the doc to guarantee homogeneity

Consequences

Unusable user manuals



Generic TOP 8 Faults

Item	Contingency actions
Non consistent techniques along life cycle	Feasibility analysis and consequent decision
Lack of traceability and crossed references in docs	Build life and consistent documents. Reference previous docs. Define doc hierarchy
PP not continuously updated	Plan review procedures and execute them
Provide rationales supporting taken decisions	Do not assume background decisions. Adapt, not adopt

Generic TOP 8 Faults

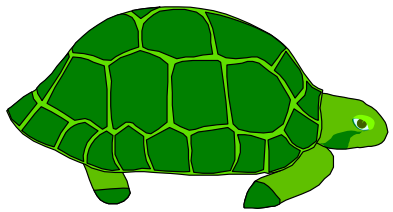
Items	Contingency actions
Too many critical TBD's TBC's	Take critical decisions on due time
Late requirements identification	Build prototypes and maquettes
Late commencement of CM activities	Initial items are not covered by CM
Lack of baseline control along config items	Execute CMP

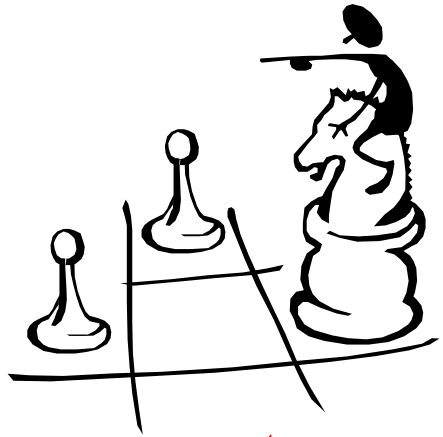
Generic Consequence

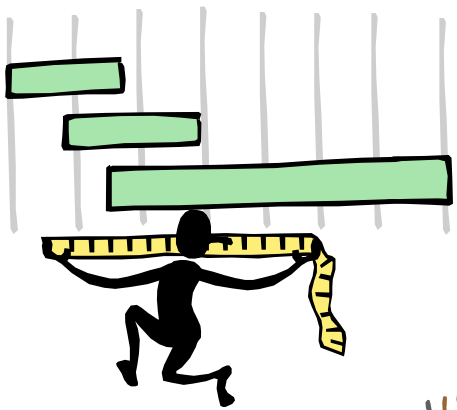
Scientific Software Projects finalise and derived products are finally operative, **but**

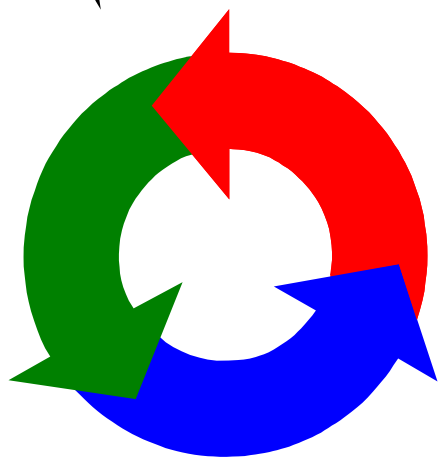
NOT as “well” as they could during and after the development

Generic Recommendations







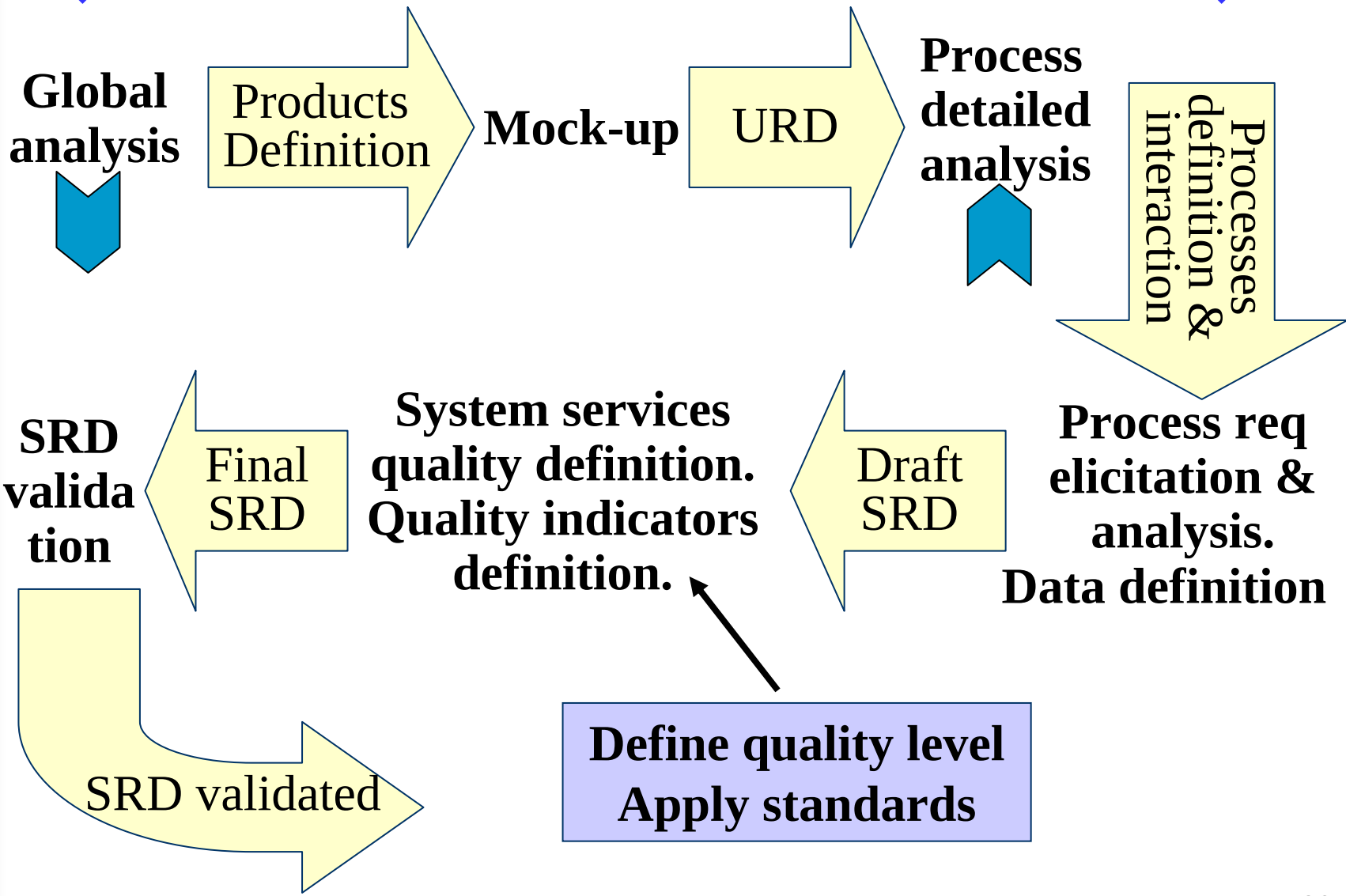


Conclusion

Key for success:

Continuous application of guidelines, principles, methods

Analysis: SRD generation



Design: ADD & DDD generation