development of safety-critical applications

- only fault tolerance depends on technical solution, the others depend on development process
- development process helps improve efficiency, efficacy and is needed for certification
- Higher level of criticality ⇒more formality
- domain specific standards/ processes

What makes a system complex?

essential complexity (Dvorak (2009))

1. *Functions to be performed:* wide range of different functions, tasks regulated by complex physical or control laws, implement complex control logic
2. *Operating Environment:* harsh environment, tight timing constraints, input from different sensors, output to different actuators
3. *Criticality:* fail-operational, implement safety functions ⇒more complex logic

What makes the development complex?

Novelty

- technology used for the first time ever (at least in domain)
- first use of technology in company

Schedule Constraints

- Critical Path: sequence of events with least lax, any delay in critical path will increase time to completion
- nearly critical paths ⇒short delay ok, more delay not ok
- with tight timing/ staffing ⇒number of nearly critical paths increase
- intentionally add slack time between events (during planing) s.t. delay not that bad

Team

- people are stupid

Geographical Distribution

- can be good or not good
- ✚ mitigate staffing constraints
- ✚ hire people not available locally
- ✚ work around the clock
- ↯ information flow can degrade
- ↯ different laws in different countries, example: A380

Organizations Maturity

- *Capability Maturity Model Integration* (CMMI), framework to measure and improve organizations maturity
  - initial to optimized
- standardized production practices

Tools

- tools support information intensive activities
- e.g. wind tunnel simulation directly from CAD

Measuring the Impact of Complexity

- COCOMO and Function Point (FP) well known to estimate effort required to develop system
- COCOMO
  - $Effort = A \cdot (SIZE)^{B+SF} \cdot EAF$

    $Effort$ effort needed to develop a system
    $SIZE$ predicted size of system in LOC
    A, B company-wide constants
    SF the *Scale Factor*: project dependent, models economy, diseconomy, exponential
    EAF the *Effort Adjustment Factor*: project dependent, multiplicative
  - parameters measuring the capability of the performing organization

    Precedentness experience of company with system
    Flexibility constraints related to project execution, schedule constraints
    Team team maturity, experience, cohesion
    Risk Resolution how are risks managed
    Process Maturity organization maturity according to CMMI model
  - adjustment factors EAF more complex
    - Early Design Model (after requirements)
    - Post-Architecture Model (after architecture definition)
    - 4 classes

      Product measure complexity of system, required reliability, database size, product complexity, required usability

      Platform measure aspects related to platform, in which system will run, execution time constraints, main storage constraints, platform volatility

      Personnel measure the capability of personnel in project, general experience, specific experience (platform, language, tool), staff turnover

      Project measure project related characteristics, communication, team geographical distribution

From System to Process

- Top-down decomposition is effective approach to manage complexity
- can be represented as a tree
- different approaches to define and organize trees
  - function trees: decomposition according to function
  - product trees: decomposition according to components
  - mixed models
- work breakdown structures (WBS): hierarchies enriched with some stuff to know what to do

Obligations and Benefits

- term from "design by contract"
- effective for allocation of functional and safety requirements
- advantages:
  - ✚ During Design: top-down fashion, validate decomposition and architectural choices
  - ✚ During Verification: bottom-up: demonstrate, that each element correctly implements the safety requirements allocated

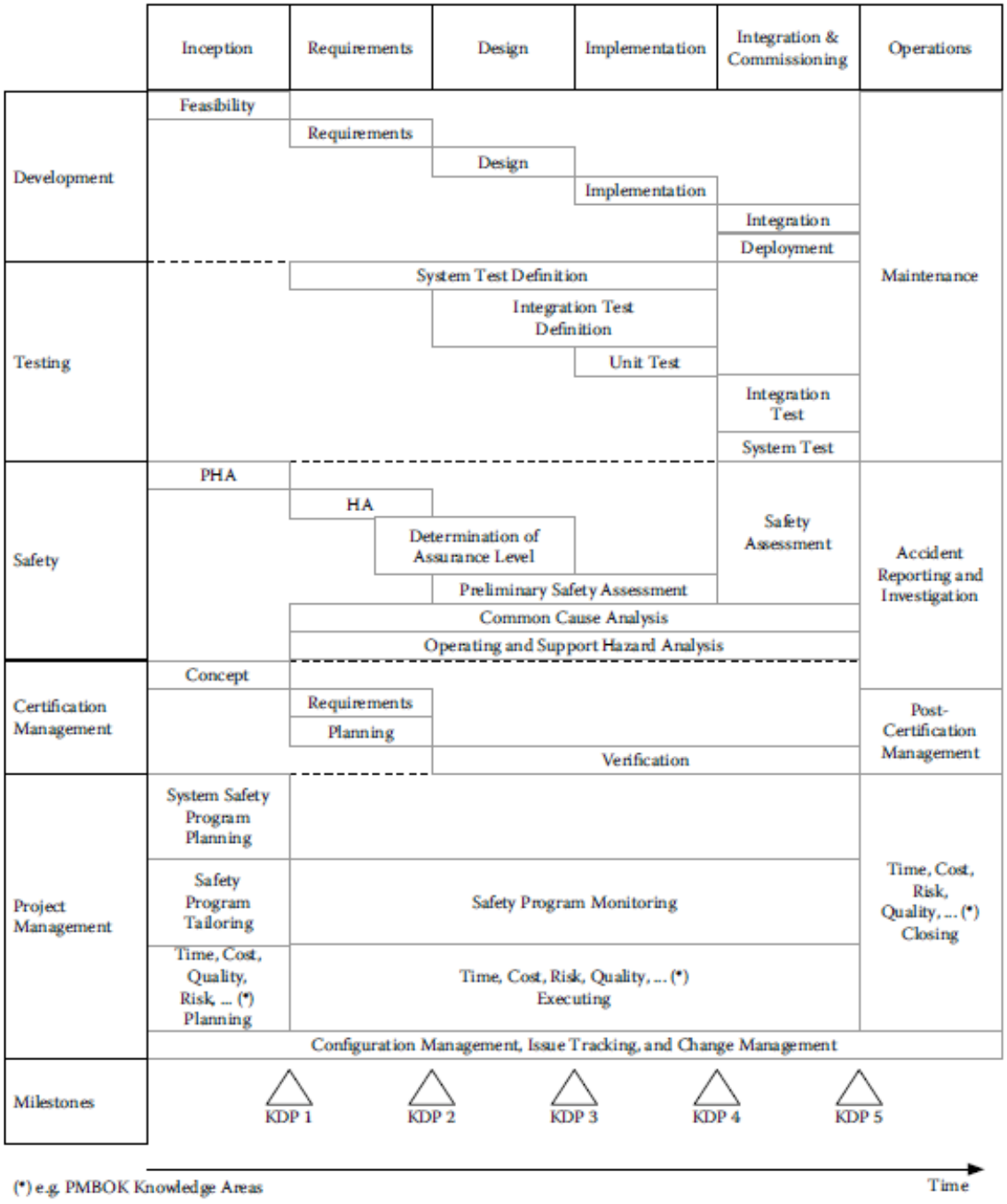
Early Assessment

- initial assessment of system, which architecture, which requirement on which component
- fault injection often by hand
- theorem proving

  increasing (TI) abstraction ⇒results in concrete space also hold in abstract space
  constant (TC) abstraction ⇒results in abstract space hold iff the hold in the concrete space
  decreasing (TD) abstraction ⇒results in the abstract space hold in the concrete space
- abstractions used as a heuristic for simplifying the proof of theorems ⇒divide and conquer

a general development framework

- time flows from right to left ⇒different stages, that characterize development of system
- by row: activities in a workflow
- by column: development progress



Phases and Phase Transition

- transition from concept to product through phases

Inception need/ opportunity for new system/ revising system is drafted, elaborated, evaluated feasibility risks, forecast profit...

Requirements characteristics of system are defined, analysis of environment

Design structure and design are drafted and evaluated, safety-critical components are individuated, design alternatives are evaluated, choices for critical and non-critical components, activities for certification

Implementation development of system according to design phase, development and testing can be in parallel, unit testing

Integration and Commissioning components are integrated into final assembly, certification takes place, system is put into operation

Operations system is maintained to ensure service

- transitions are determined by milestones
- results of verification are used with complementary goals:
  1. formal verification point to decide *transition to next phase*
  2. formal verification to *decide whether to continue project*

Comparison with Other Frameworks

Rational Unified Process (RUP)

- organize development into four phases: *Inception, Elaboration, Construction, Transition*
  - four milestones define phase transitions

    Life-Cycle Objectives concludes Inception phase, provides stakeholders with information necessary to decide, if project is to be continued

    Life-Cycle Architecture concludes elaboration phase, provides stable version of vision, requirements, architecture, prototype

    Initial Operational Capability concludes Construction phase, system ready to deploy, all other stuff (e.g. training) in place

    Product Release concludes Transition, verify that the system has initial goals and met stakeholders expectation

European Space Agency (ESA)

- seven phases

  Phase 0, Mission analysis and needs identification goals, preliminary studies for market opportunity, cost, main risk

  Phase A, Feasibility main project plan outline, technical feasibility, Phase 0 gets refined

  Phase B, Preliminary Definition previous phases refined, technical solutions committed to, pre-development on most critical technologies, safety assessment

  Phase C, Detailed Information system is developed, detailed designs, engineering models, critical components

  Phase D, Qualification and Production system is produced and qualified for development

  Phase E, Operations/ Utilization system is deployed, (ESA) ground operations, before, during, after launch, launch itself, mission goal itself

  Phase F, Disposal disposal plan is executed

- transitions are regulated by reviews, type and number depend on phase

  MDR, Mission Definition Review concludes Phase 0

  PRR, Preliminary Requirements Review concludes Phase A

  SRR, System Requirement Review, PDR, Preliminary Design Review conclude Phase B

  CDR, Critical Design Review concludes Phase C

  QR, Qualification Review, AR, Acceptance Review, ORR, Operational Readiness Review conclude Phase D

  FRR, Flight Readiness Review, LRR, Launch Readiness Review, CRR, Commissioning Result concludes Phase E

  MCR, Mission Close-out Review concludes Phase F

Organization and Sequencing of Phases

- strict sequencing ⇒waterfall development
- critique:

  Efficiency strict sequencing does not allow parallel development

  Rigidity when features become clearer ⇒no backtracking possible

  Difficutly in managing risks assembly at the end, evaluation only at late state

Workflows

- integrate different types of activities and competences

  Development activities necessary to build system, (functional) requirements, design of architecture, integration of components

  Testing activities to confirm, that system works correctly

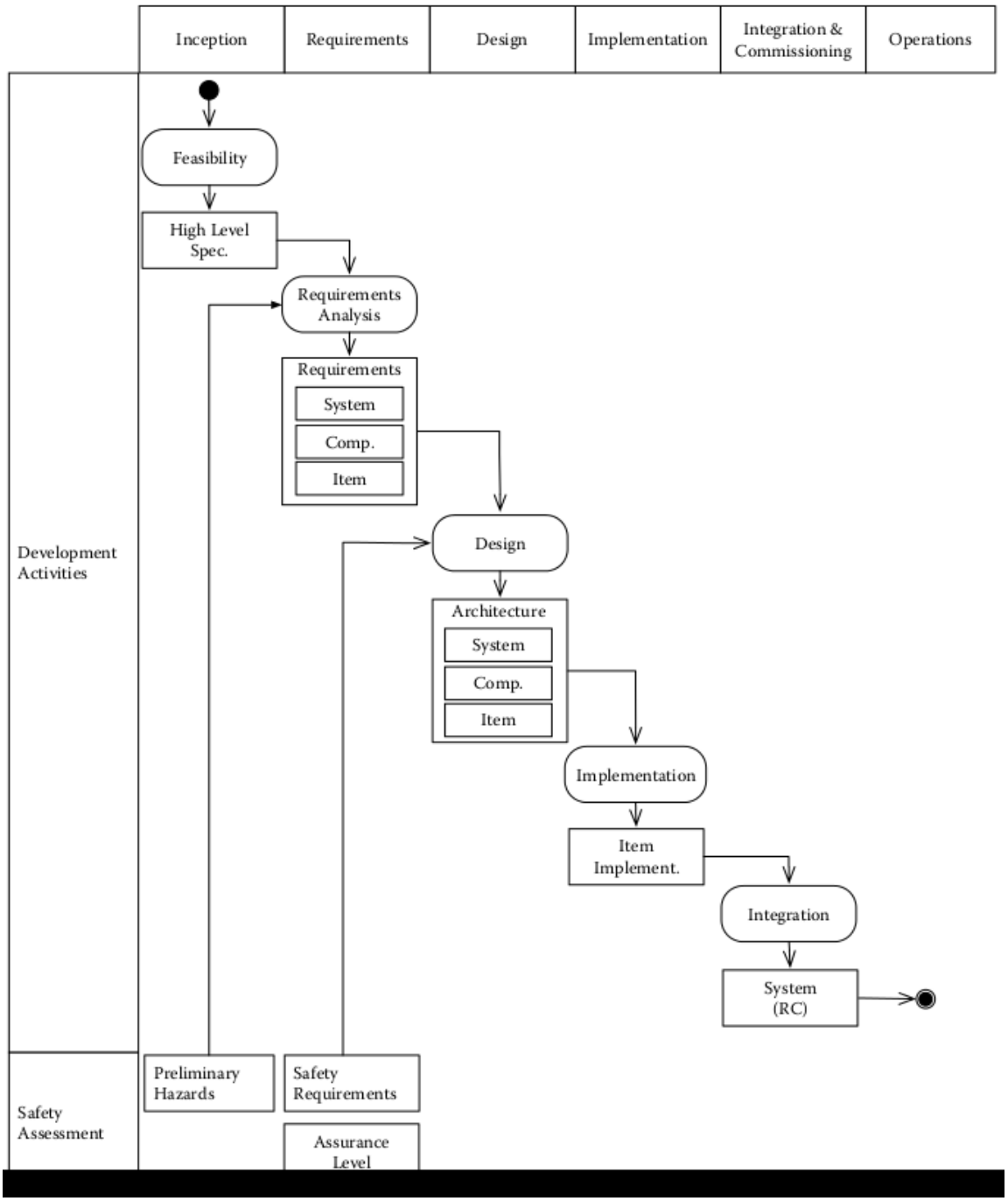
  Safety activities to ensure system under degraded conditions

  Certification management activities to ensure the conditions for certification

  Project Management activities to coordinate effort in other workflows

Development Workflow

- show activities during development, derived from UML activity diagram



Feasibility Study

- assess technical feasibility, first high-level definition of functions

⇒ feasibility document

- 5M model

  Mission main purpose of a system, success criteria, strategic objectives

  Man human factor, risk, constraints, training

  Machine the platform that will be used to achieve mission goal

  Media environment in which system will operate

  Management set of procedures, regulations necessary for operation, maintenance and disposal of system

Requirement Analysis

- outputs of feasibility study are refined to produce sufficiently detailed and precise description

⇒ requirements document (or set thereof)

- often contractual agreement
- three tasks

  Requirements elicitation elicit system characteristics

  Requirement specification organize requirements, annotated with meta-information

  Requirement validation confirm, that requirements describe "the right system"

Design

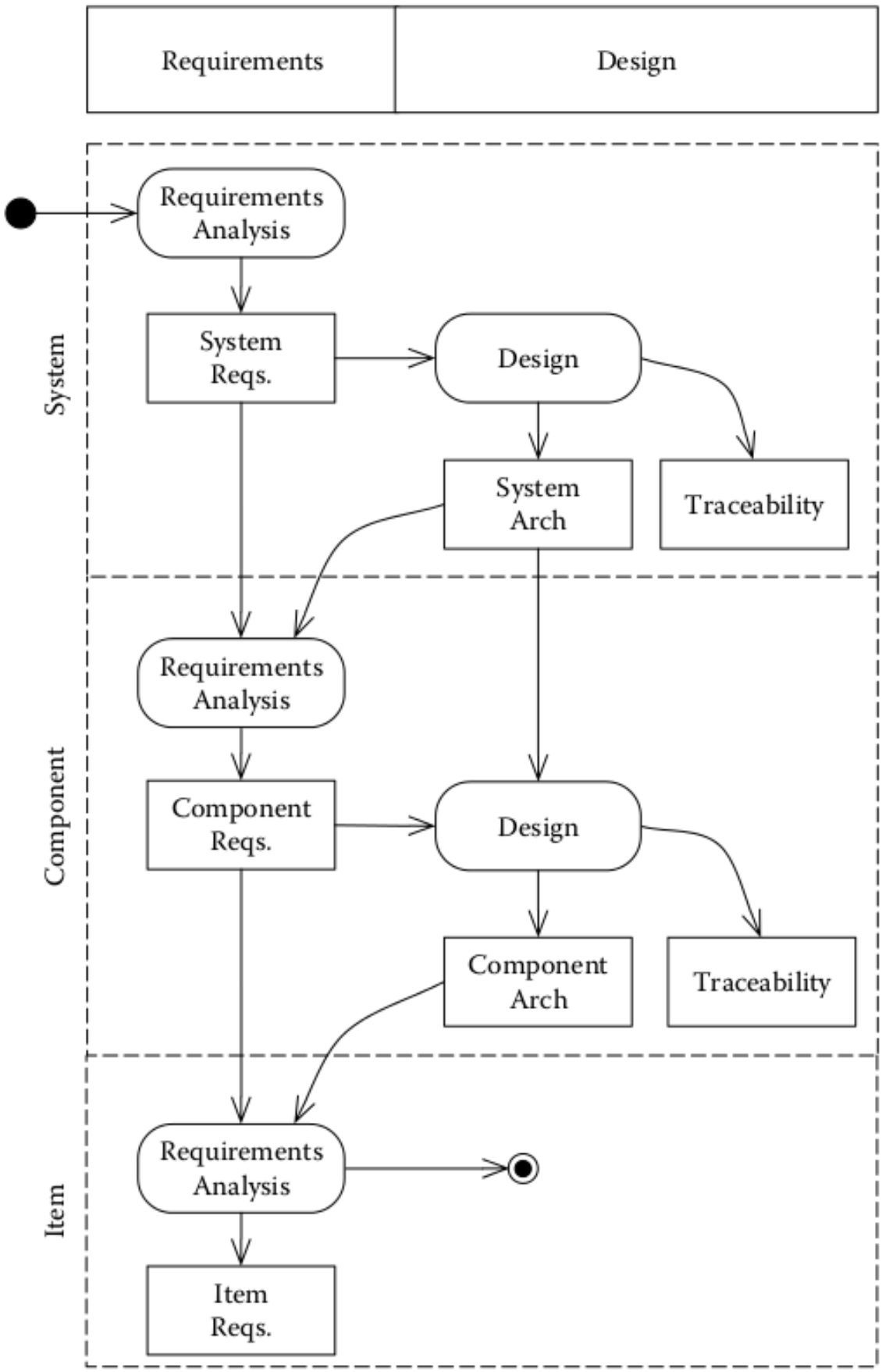
- determine architecture of system ⇒diagrams, documents, blueprints
- allocation of safety requirements has consequences:
  - imposes specific design choices, e.g. adoption of TMR for critical components
  - imposes constraints on development process that must be adopted

Implementation and Integration

- build leaf elements of system during *implementation*
- assemble and integrate components during *integration*

Hierarchical Design of Systems

- time flows from top to bottom
- at first level of detail (system level) requirements are for the whole system
- at second level of detail (component level) architecture and requirements are decomposed, refined and allocated to components
- at third level (item level, items are assumed to be simple), refine components into requirements, basis for item implementation



Testing Workflow

- testing activities span throughout system development, may continue in production

development testing during development

verification testing verification of final product

production testing ensure every does not have fabrication defects

- testing contains different stages

  planning stage scope, plan general objectives are drafted

  definition phase testing strategy is defined, test cases are constructed

  execution phase test campaign is conducted, results are reported

Acceptance Test Definition

- verify system compliance with requirements
- testing performed in test cases
- tests according to development stage

  Analysis use of mathematical modeling and analytical techniques

  Demonstration providing means to demonstrate a requirement is satisfied

  Inspection visual examination of realized end product

  Test use end product to obtain detailed data needed to verify compliance with the requirements

Integration Test Definition

- ensure, that subsystems interact as expected ⇒test integration document

Unit Test Definition

- verify module or item
- typically look at actual implementation

Test Execution

- require delivery of implemented items, components and system:

  Unit Test Execution execute unit tests, often automated, code that breaks previous assumptions, fail the test

  Integration Tests execute integration tests

  Acceptance Tests ensure, that delivered system correctly implements the requirements

- tests produce several version:

  alpha release early system releases

  beta release more mature system

  RC-N Release Candidate, number N

- testing can not prove absence of faults

Safety Assessment Workflow

- two different phases, complementary goals
  - during system development (requirements, design, implementation) ⇒"constructive" role: guide design activities, ensure safety goal
  - during integration ⇒"destructive role", demonstrate whether system is safe, actually meets safety requirements ⇒question all design hypotheses
- techniques are the same (e.g. fault tress, FMECAs) but purpose is different

Preliminary Hazard Analysis (PHA) and Hazard Analysis (HA)

- goal: identify potential hazards and safety risks

  PHA when initial information is available

  HA when more detailed and more consolidated information is available

- strategies to manage hazards

  1. *Design for minimum risk*: design to eliminate risks or reduce to acceptable level
  2. *Incorporate safety devices*: if risks cannot be eliminated, reduce risk via use of fixed automatic or other safety design features, provisions for periodic functional checks
  3. *Provide warning devices*: neither design nor safety devices can effectively eliminate identified risks or adequately reduce risk, produce adequate warning signal
  4. *Develop procedures and training* where risk elimination is impractical through design selection or safety and warning devices ⇒procedures and training

Determination of Assurance Levels

- components are not equally critical ⇒rigor of development due to *assurance levels*
- classify for effects of violation of safety requirements

  Catastrophic malfunction causes unsafe operational conditions

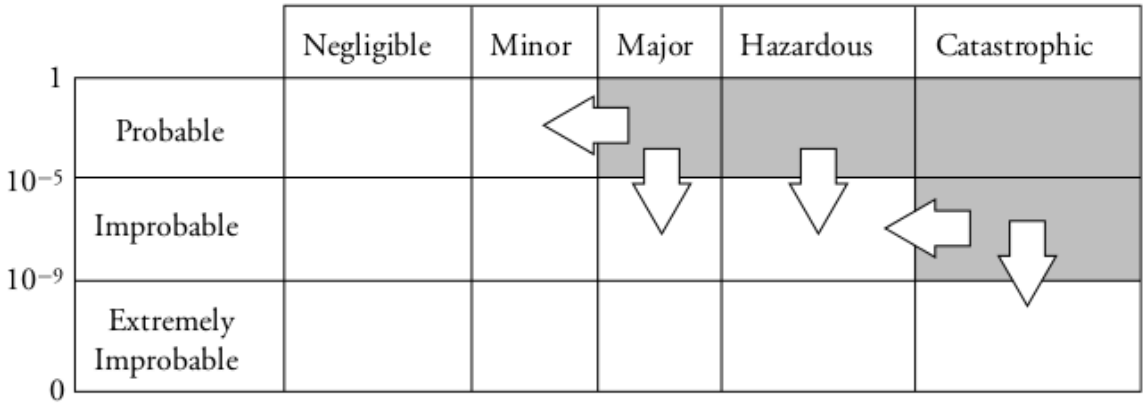
  Hazardous malfunction severely reduces operational safety, might cause severe stress on operators and have adverse effects on occupants

  Major malfunction significantly reduces safety margins and increases workload of operators, causing distress to occupants

  Minor if malfunction causes slight reduction in safety margins, slight increase in workload, some inconvenience for occupants

  Negligible malfunction has no effect on safety

- probability is another factor



- NASA assurance level

  1. Determine *risk index* of software component, probability and impact of failures
  2. Determine *control category*, look at characteristics of software
  3. Determine *software risk index*, number from 1 to 5, correspond to required safety effort necessary for its development

Preliminary Safety Assessment (PSA)

- perform safety analysis on system while its being developed
- interest between preliminary safety assessment and design in the case of hierarchical development

process

- outputs are requirements, architecture and list of hazards
- activities proceed to next level of refinement (component level) ⇒specification of each component

Safety Assessment (SA)

- proceed in bottom-up fashion, from component to system level, verfiy the hypotheses made during design
- safety verification include

  Item and Component Safety Assessment lower levels of architecture are verified, Fault Tree Analysis, FME(C)A and Common Cause Analysis

  System Safety Assessment verify architecture, Fault Tree Analysis, FME(C)A and Common Cause Analysis

Common Cause Analysis (CCA)

- in case of systematic failures ⇒redundancy not useful
- three kinds of common cause analysis (SAE)

  Common Mode Analysis systematic errors in development process, usually done via checklists

  Particular Risk Analysis considers external events that invalidate the independence of components (e.g. fire, snow, lightning, ruptures, leakages, ...)

  1. Determine scope of analysis (the risk to analyze)
  2. Define the models regulating the risk
  3. Define the effects of the risk base on the model
  4. Determine the failures on the systems components (based on effects of risk)
  5. Determine how the failures propagate to the rest of the system

  Zonal Safety Analysis physical proximity can invalidate the independence of components

  Identify system's zone the different zones of the system are identified, for many systems such zones are predefined and standardized

  list components all components of zones

  list hazards all hazards pertaining to the zone are listed

  perform analyses identify effect of hazard (FMEA, FTA)

  define mitigation rules strategies how to deal with hazard due to zonal installation

- redundancy based upon different designs

Common Cause Analysis and Software

- redundancy does not really improve with software
  - failures in software primarily design problems
  - no variations, all copies are identical, will fail in the same way
- use *diversity*, independent software versions by independent teams, different tools and programming languages ⇒different teams may still produce coincident errors

Operating and Support Hazard Analysis (O&SHA)

- identify, analyze and evaluate the possible hazard to people, environment and property associated to system life cycle
- outputs are changes to design, procedures, testing, training, ...

Certification Management Workflow

- safety-critical systems may be certified in certain areas (EASA - European Aviation Safety Agency, NRC - Nuclear Regulatory Commission)
- product certification workflow helps ensure that certification proceeds as smoothly as possible, involve certification agency as soon as possible
- documents:

  Contractual activities ensure, that all legal aspects and agreements between certification authority and applicant are properly taken care of

  Managerial activities all activities related to managing the certification process, includes: planning, monitoring, ensuring a proper integration of technical certification activities in other workflows

  Technical activities activities related to evaluating compliance of system with target regulations and safety goals.

- certification phases, transitions happen with audits/ checklists

  1. conceptual design phase
     - *ensure early , value added, joint involvement with an expectation to surface critical areas and the related regulatory issues*

       process orientation partnership between applicant and certification authority is established ⇒"Partnership for Safety Plan" (PSP)

       pre-project guidance, familiarization briefings discussion about policies, regulations, product characteristics ⇒identify potential issues, risks, points of attention

       certification planning outlines certification plan, includes data to be submitted, methods of compliance, schedule
  2. requirement definition phase
     - ensure that certification project is up and running

       project setup set up organizational structure, team, tools

       issues book repository of *issue papers*, which describe technical and regulatory issues

       certification basis how compliance with safety regulations will be shown
  3. compliance planning phase
     - define activities for actual implementation of certification
     - participation of certification authority needed on every decision on critical stuff
  4. implementation phase
     - actual implementation of certification takes place
     - *conformity inspection*: provide documentation of compliance with design,
     - tests, inspections, analyses
  5. post-certification phase
     - define stuff for managing and maintaining certification through life-cycle

- *issues book* properly trace and deal with issues
  compliance can be shown by

  standard techniques issue is shown to be properly handled via safety analysis technique

  equivalent level of safety compensating factors are put in place

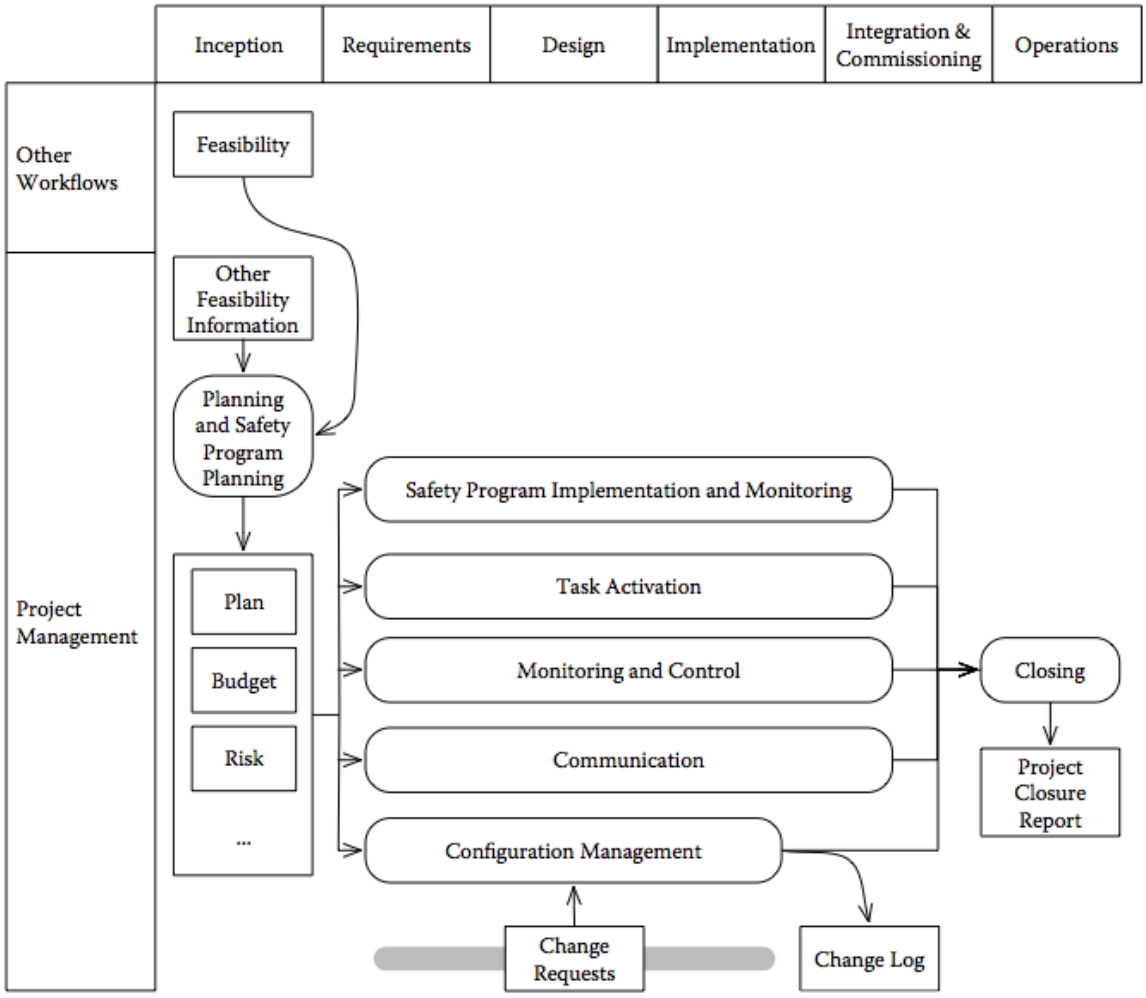
  special condition adoption of unusual design feature or technique does not allow standard practices

Project Management Workflow

"The application of knowledge, skills, tools and techniques to project activities to meet project requirements."

- *PRoject IN a Controlled Environment* (PRINCE2) (British)

- *Project Management BOdy of Knowledge* (PMBOK) (American)



Safety Process Definition and Tailoring

- customize development process, adapt it to level of formality required
- output is a document that describes (minimum):
  - how safety program will be implemented, techniques and standards
  - resources responsible for implementation of safety plan (⇒dedicated safety team)
  - how safety program and safety team fit into overall project structure, lines of communication

Safety Program Implementation and Monitoring

- ensure that safety program is implemented according to standards and rules defined for project
- safety manager responsible for
  - maintain log of identified hazards and residual hazards
  - formally approving any residual hazard
  - communicate hazards and residual hazards to end user

Other Management Activities

planning forecasting the dimensions of project

task activation project manager activates and verifies termination of tasks according to plan

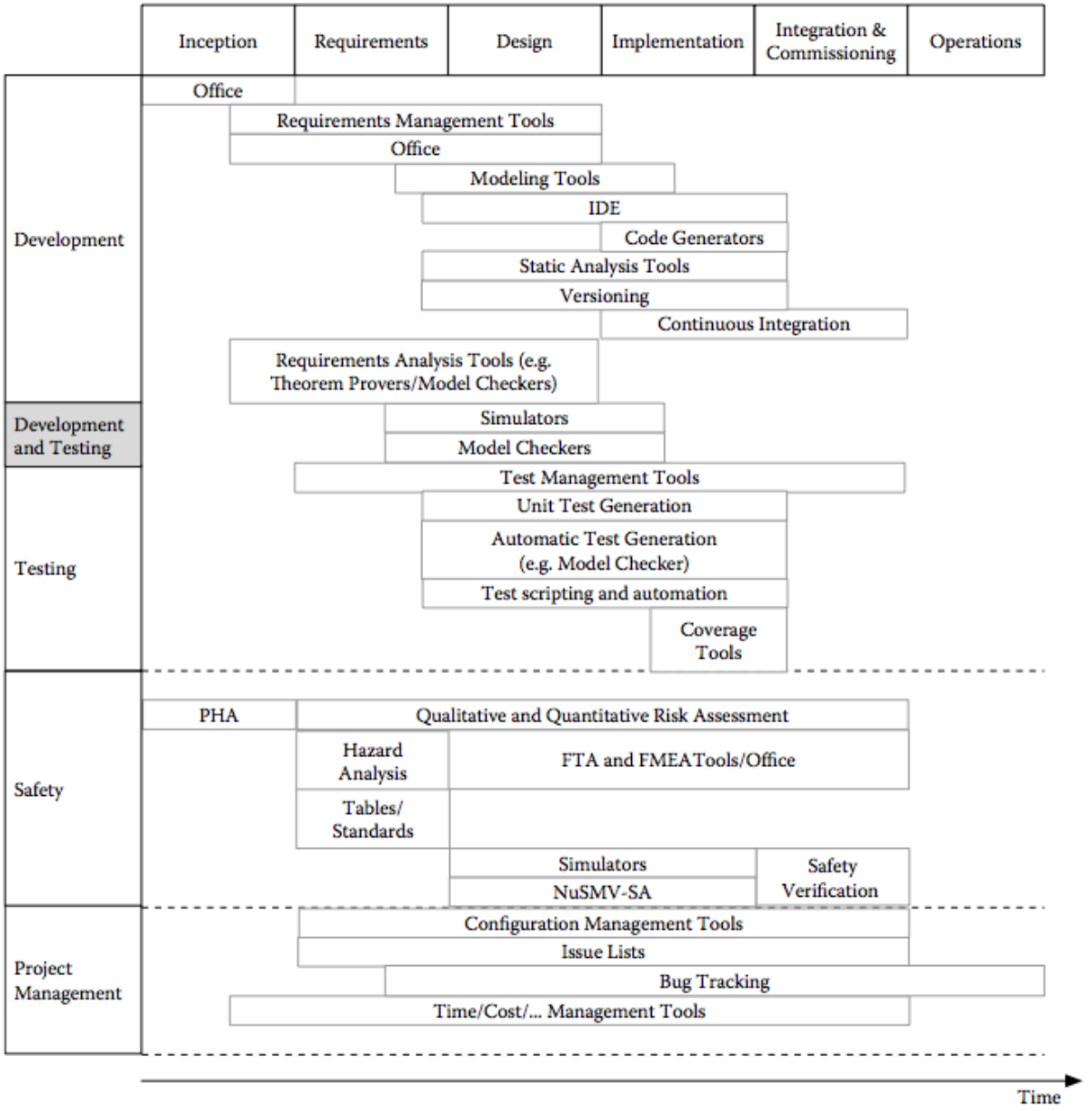
monitoring and control data about progress is collected and compared with baseline plans

communication project manager is responsible for information flow

configuration management provide visibility and control over changes

closing asses project after end

Tool Support



Supporting the Development Workflow

requirement management tools

- annotate enrich requirements with meta-information
- maintain traceability

requirement analysis tool reason about requirements, provide information about consistency/ completeness, often with formal representations

modeling tools, simulators, code generators blur distinction between design, development and testing

- simulation: SPICE, MATLAB Simulink
- formal verification: SPIN; NuSMV, theorem provers

static analysis tools IDEs, certified compilers

versioning systems coherent management of whole sets of files

build and continuous integration simplify management of complex software systems

configuration management

issue and bug-tracking management

Supporting the Testing Workflow

- tools, that help maintain specification of tests, often web based, trace test results, integration with bug-tracking tools
- tools to support automatic generation and execution of tests, based on model checkers and static analysis, automatic generation of test cases
- tools to support evaluation of tests, verify completeness of tests, code coverage

Supporting the Safety Analysis Workflow

- tools for management of safety-related information, FaulTree+, semantically meaningful way of drawing and managing fault trees
- tools for quantitative assessment of risks and hazards, based on probability theory, monte carlo simulation, markov analysis
- tools for qualitative assessment of risks and hazards, tools like NuSMV-SA, automatic generation of fault trees (?)