Software and Software Engineering

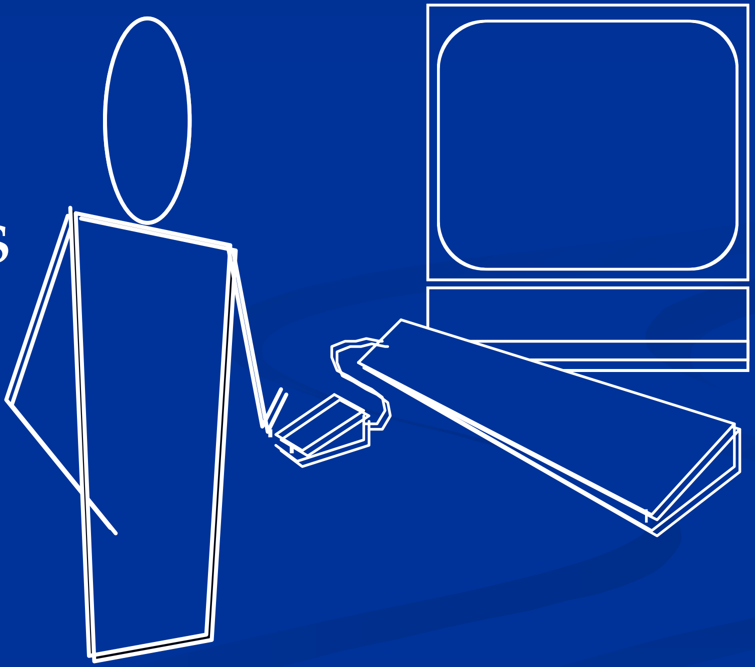
Software's Dual Role

- Software is a product
  - Delivers computing potential
  - Produces, manages, acquires, modifies, displays, or transmits information
- Software is a vehicle for delivering a product
  - Supports or directly provides system functionality
  - Controls other programs (e.g., an operating system)
  - Effects communications (e.g., networking software)
  - Helps build other software (e.g., software tools)

What is Software?

Software is a set of items or objects that form a “configuration” that includes

- programs
- documents
- data ...



Software

Software is the collection of computer programs, procedures, Rules and associate with documentation and data

Software Engineering

Software Engineering is a systematic approach to development, operation, maintenance and retirement of software.

Or

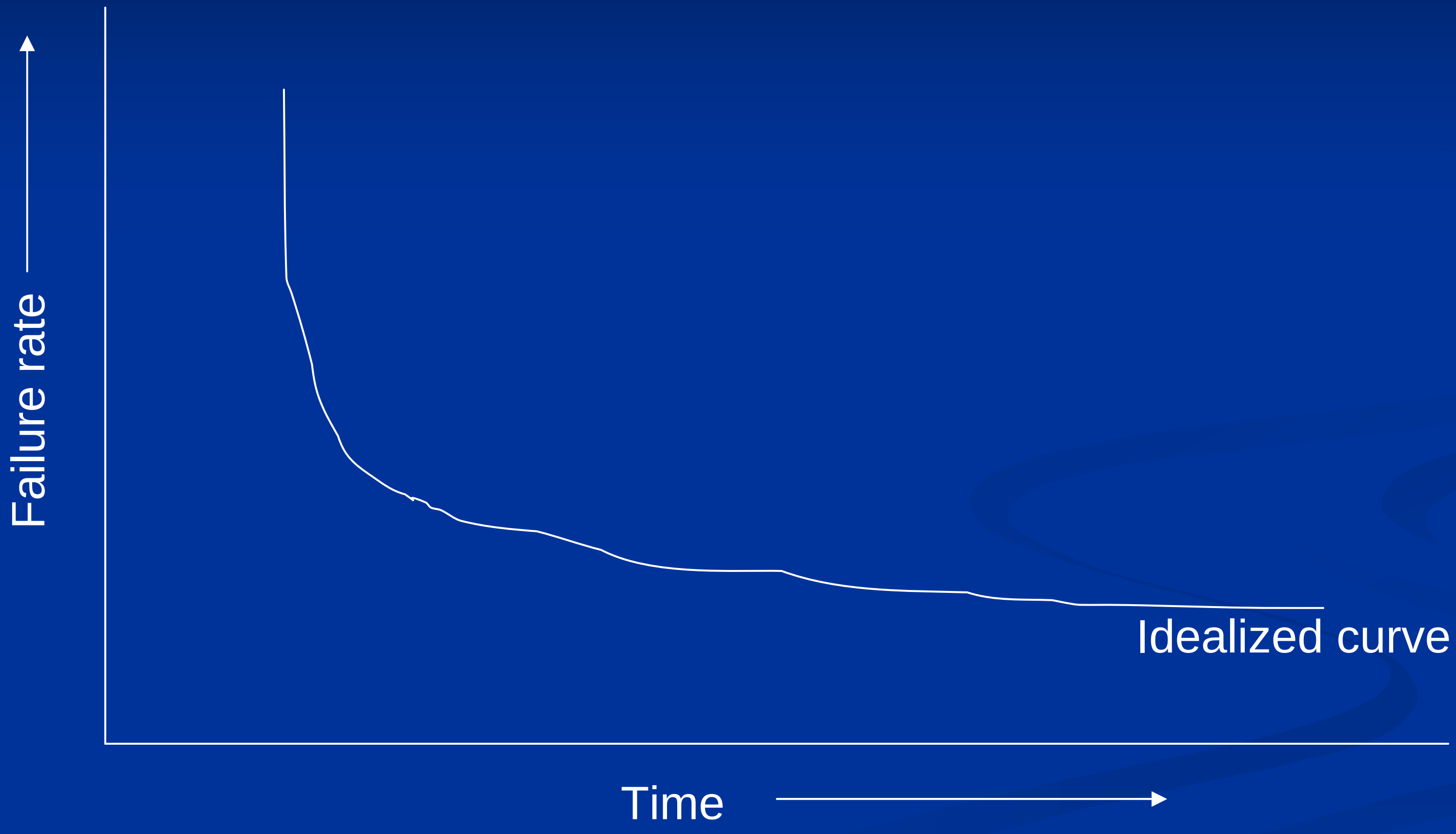
Software Engineering is the application of science and mathematic by which the capabilities of computer equipment are made useful to man via computer programs, procedures and associated with documentation.

Goal of the Software Engineering

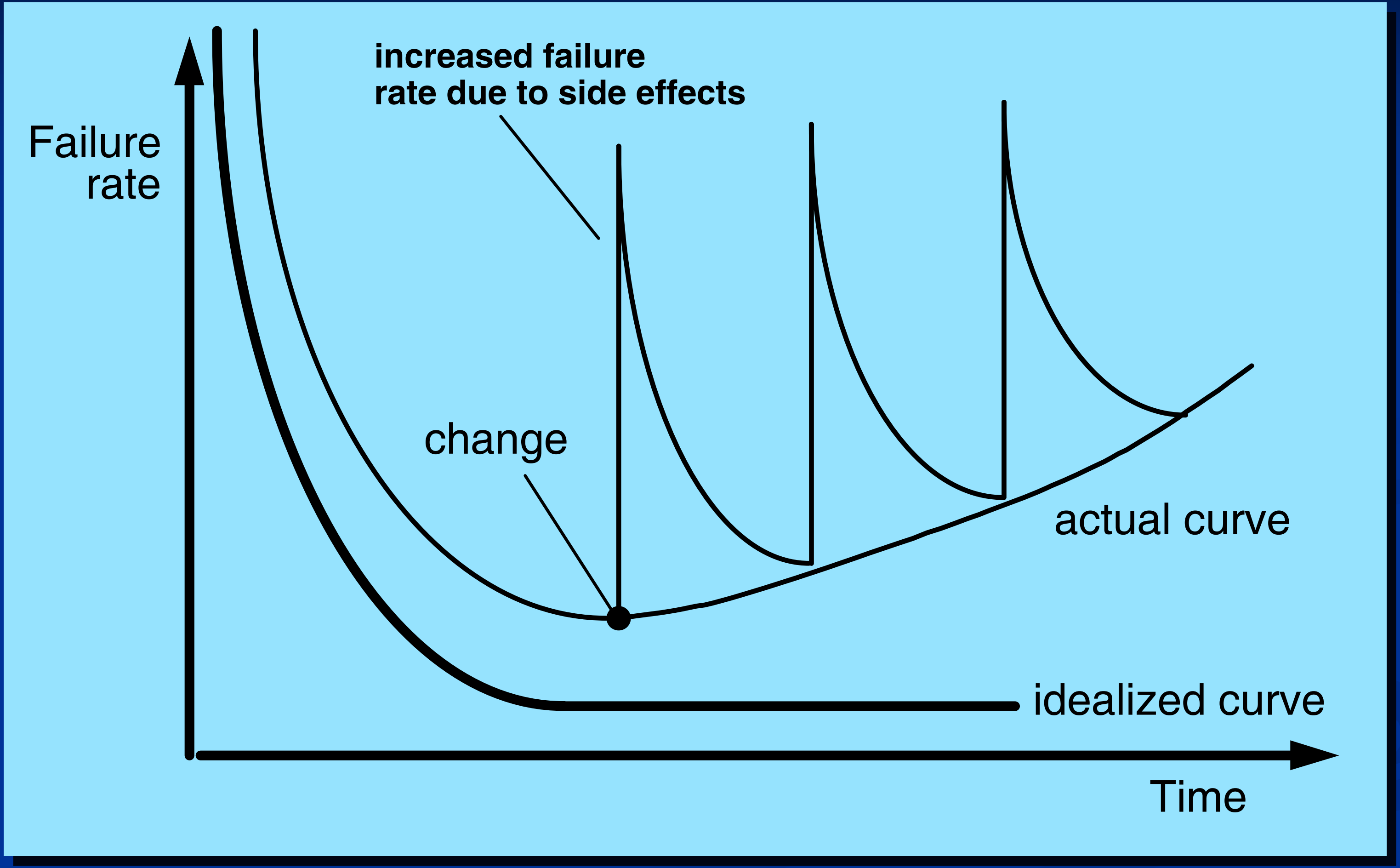
The software produce high quality software at low cost

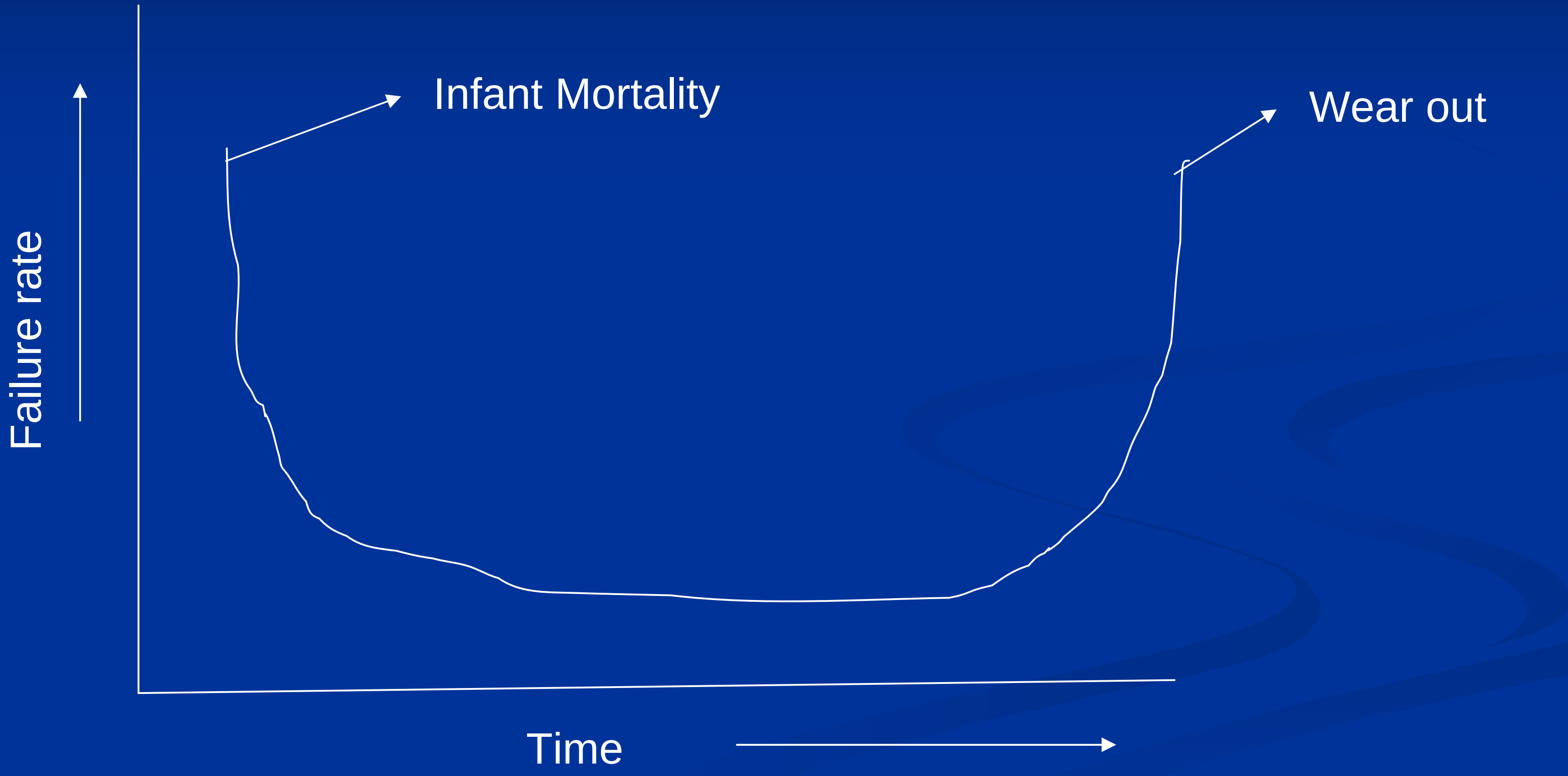
What is Software Engineering ? or characteristics

- software is engineered
- software doesn’t wear out
- software is complex


Failure rate
Idealized curve
Time

Wear vs. Deterioration




Infant Mortality
Wear out
Failure rate
Time

Software Applications

- system software
- application software
- engineering/scientific software
- embedded software
- product-line software
- WebApps (Web applications)
- AI software

Software—New Categories

- Ubiquitous computing—wireless networks
- Netsourcing—the Web as a computing engine
- Open source—"free" source code open to the computing community (a blessing, but also a potential curse!)

- Data mining

  - Grid computing
  - Cognitive machines
  - Software for nanotechnologies

Legacy Software

Why must it change?

- software must be adapted to meet the needs of new computing environments or technology.
- software must be enhanced to implement new business requirements.
- software must be extended to make it interoperable with other more modern systems or databases.
- software must be re-architected to make it viable within a network environment.

Software Evolution

- The Law of Continuing Change (1974): E-type systems must be continually adapted else they become progressively less satisfactory.
- The Law of Increasing Complexity (1974): As an E-type system evolves its complexity increases unless work is done to maintain or reduce it.
- The Law of Self Regulation (1974): The E-type system evolution process is self-regulating with distribution of product and process measures close to normal.
- The Law of Conservation of Organizational Stability (1980): The average effective global activity rate in an evolving E-type system is invariant over product lifetime.
- The Law of Conservation of Familiarity (1980): As an E-type system evolves all associated with it, developers, sales personnel, users, for example, must maintain mastery of its content and behavior to achieve satisfactory evolution.
- The Law of Continuing Growth (1980): The functional content of E-type systems must be continually increased to maintain user satisfaction over their lifetime.
- The Law of Declining Quality (1996): The quality of E-type systems will appear to be declining unless they are rigorously maintained and adapted to operational environment changes.
- The Feedback System Law (1996): E-type evolution processes constitute multi-level, multi-loop, multi-agent feedback systems and must be treated as such to achieve significant improvement over any reasonable base.

Software Myths

- Affect managers, customers (and other non-technical stakeholders) and practitioners
- Are believable because they often have elements of truth,

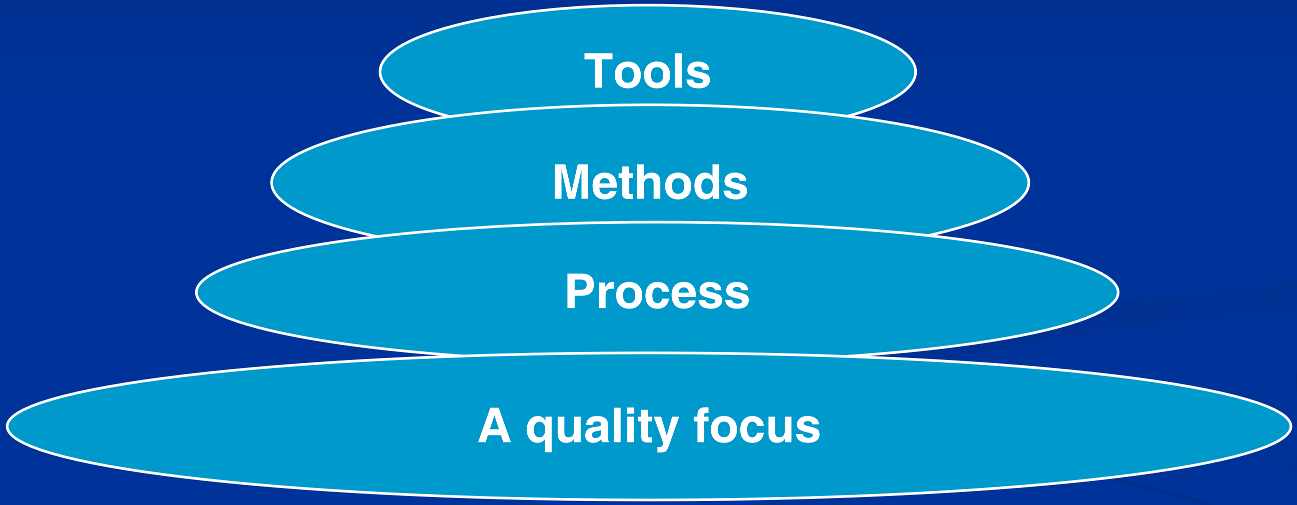
but …

- Invariably lead to bad decisions,

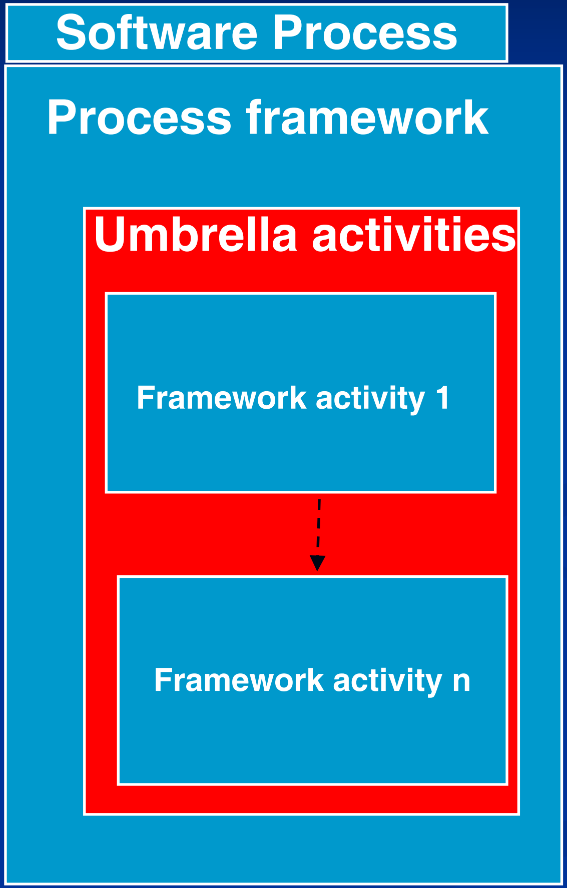
therefore …

- Insist on reality as you navigate your way through software engineering

Software Engineering Layers



Software Process Framework



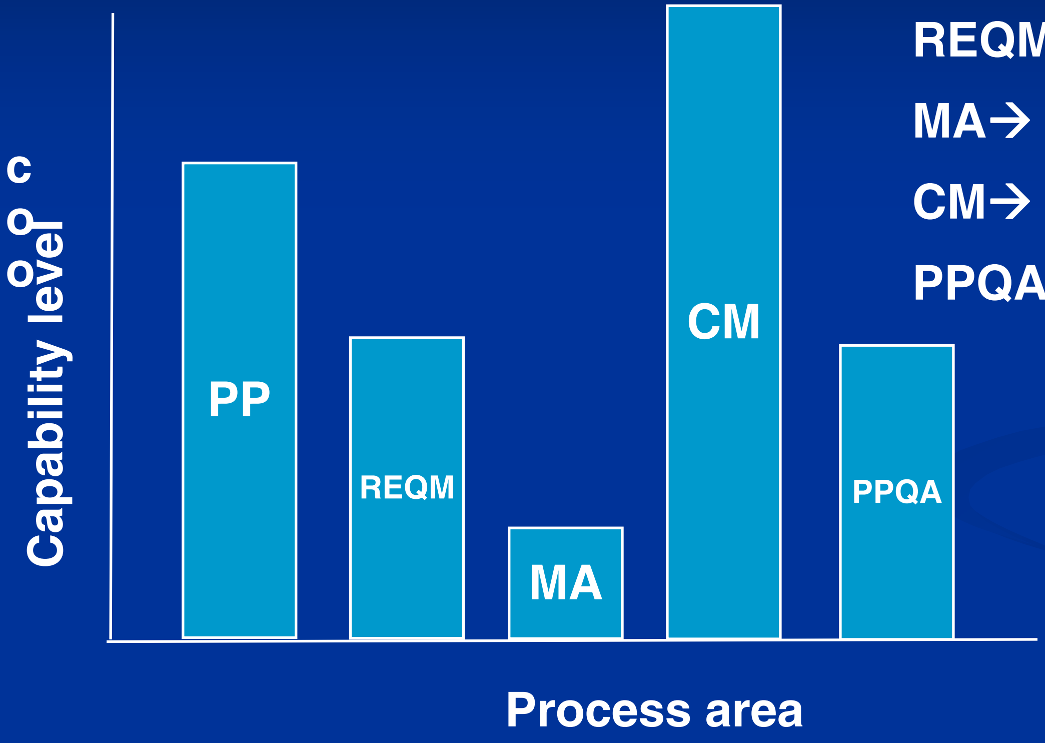
Generic process framework activities

- Communication
- Planning
- Modeling
- Construction
- Deployment

Software project tracking and control

- Risk management
- Software quality assurance
- Formal technical reviews
- Measurement
- Reusability management
- Work product preparation and production

The capability maturity model integration (CMMI)



PP→ Project planning

REQM→ Requirements management

MA→ Measurement and analysis

CM→ Configuration management

PPQA→ Process and product QA

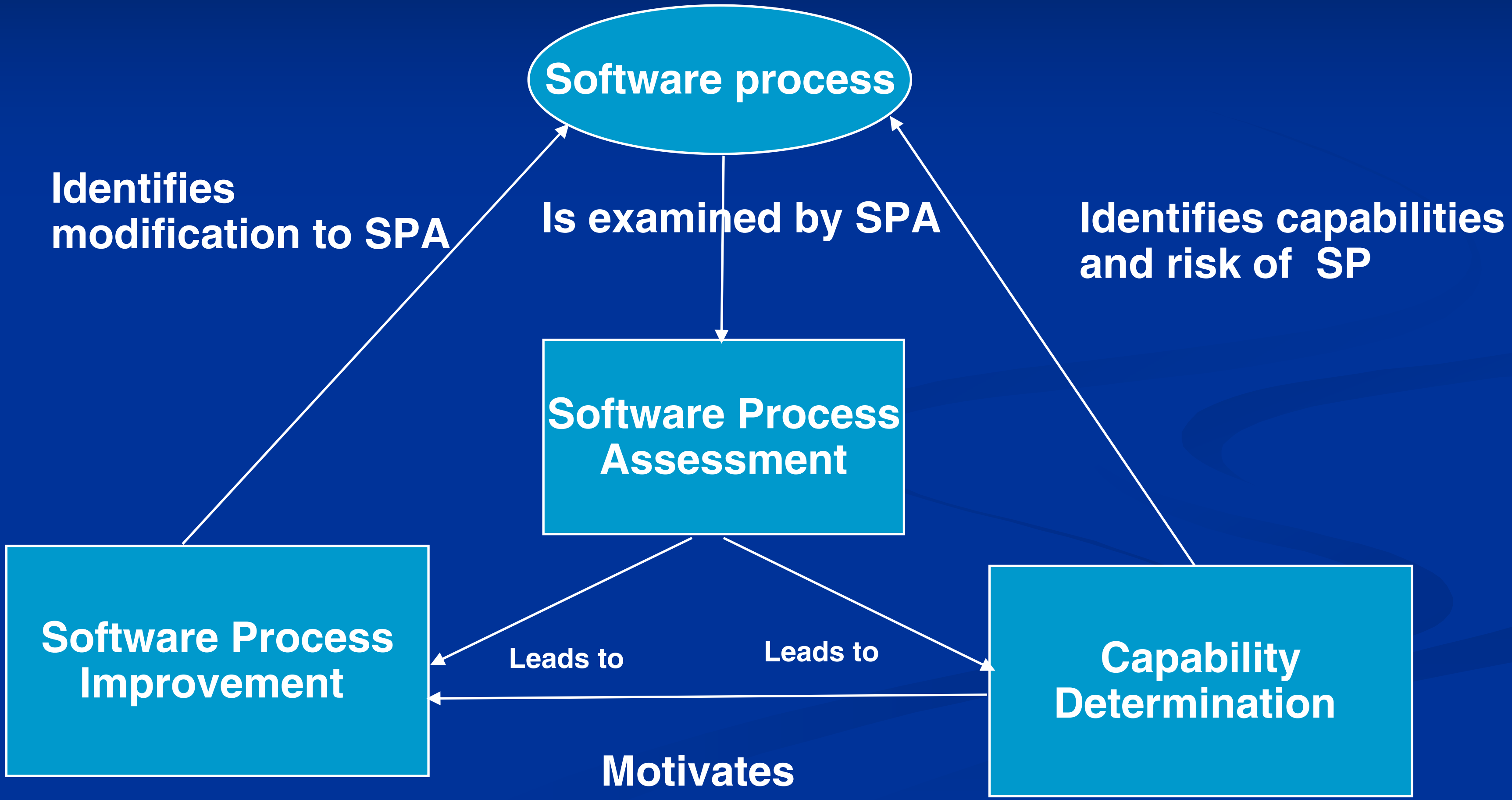
Level of CMMI

- Incomplete
- Performed
- Managed
- Defined
- Quantitatively managed
- Optimized

Process Patterns

- Customer communication
- Requirements gathering
- Spiral model or prototyping model
- Resulting context
- deployment

Process Assessment


Software process
Identifies modification to SPA
Is examined by SPA
Identifies capabilities and risk of SP
Software Process Assessment
Software Process Improvement
Leads to
Leads to
Capability Determination
Motivates

Personal and Team Process Models

- Personal Software Process (PSP)

  Planning, High level design, High level design review, Development, Postmortem.

- Team Software Process (TSP)

  Build self-directed, Managers, Software process, improvement guidance, teaching