



NRC-CNRC

*Institute for
Information
Technology*

Emerging essentials of software development

Hakan Erdogmus

October 17, 2007



National Research
Council Canada

Conseil national
de recherches Canada

Canada

Two historical views of software development

- **Engineering/research:** software is “hard”, software development a scientific, engineering plan-oriented process

Plan



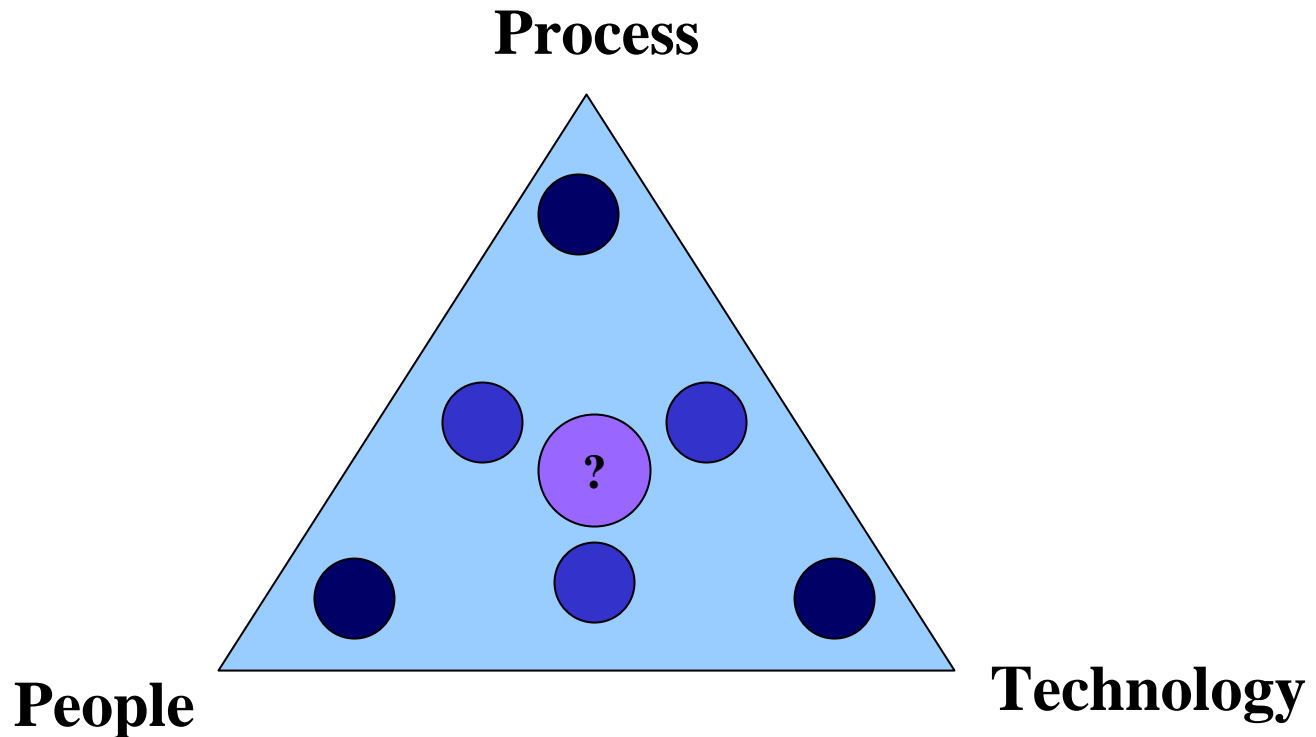
Evolve

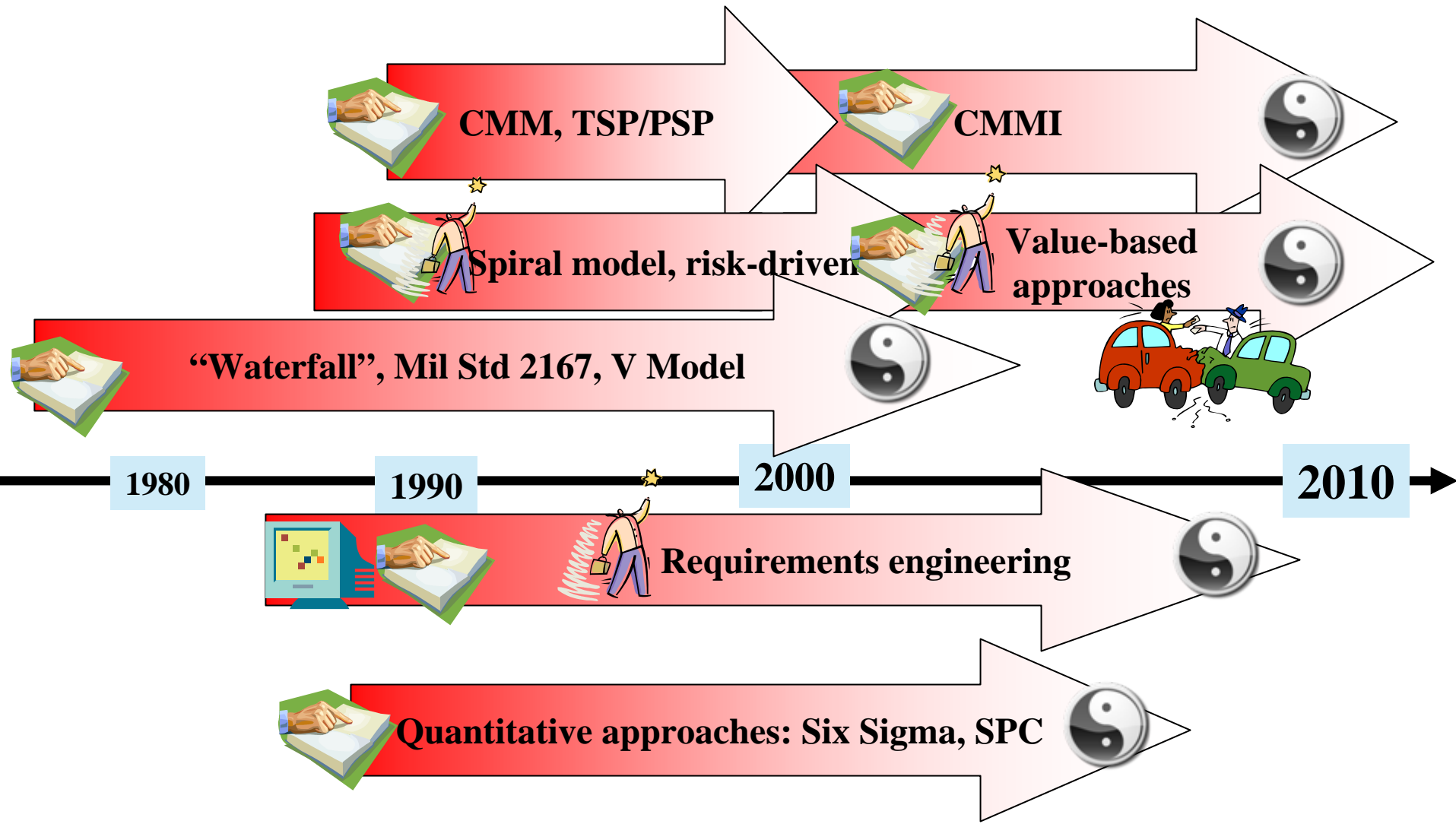
- **Grassroots/practitioner:** Software is “soft”, software development is a crafty, organic, skills- and technology-driven, organic process

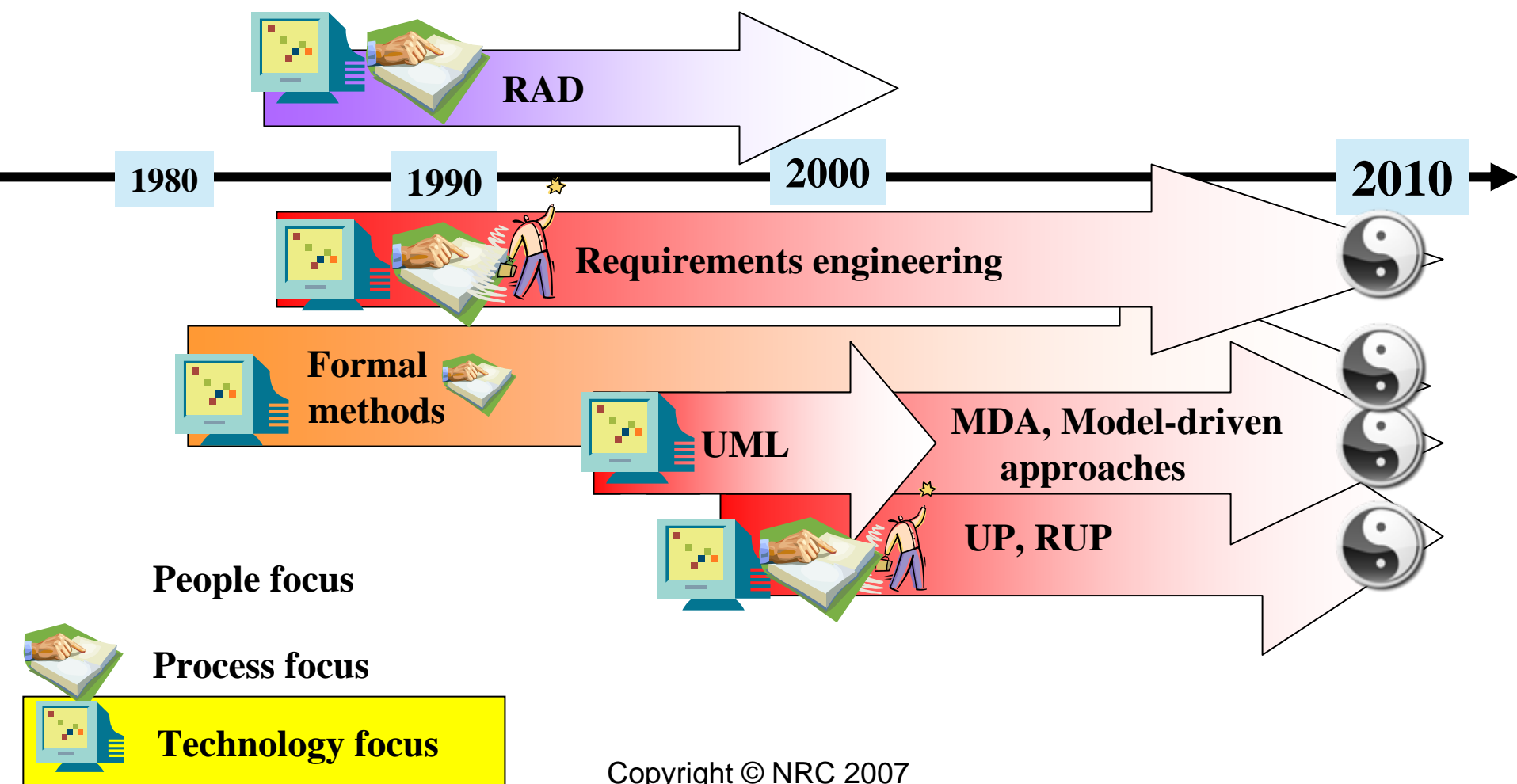
Scope of this talk

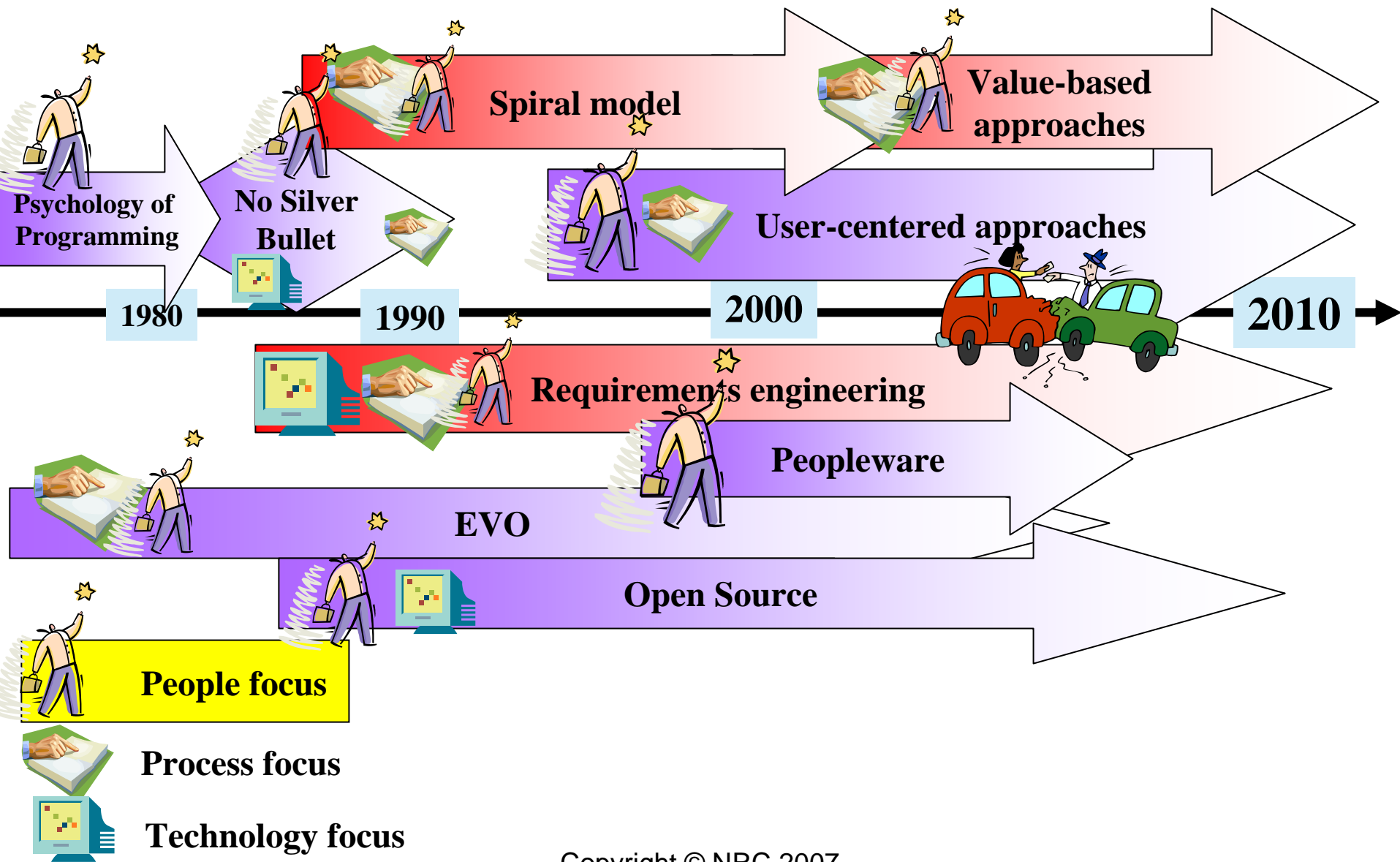
- **Process-related trends**
- ***Not* technological trends and programming paradigms**
- ***Not* innovation, market success, business models**

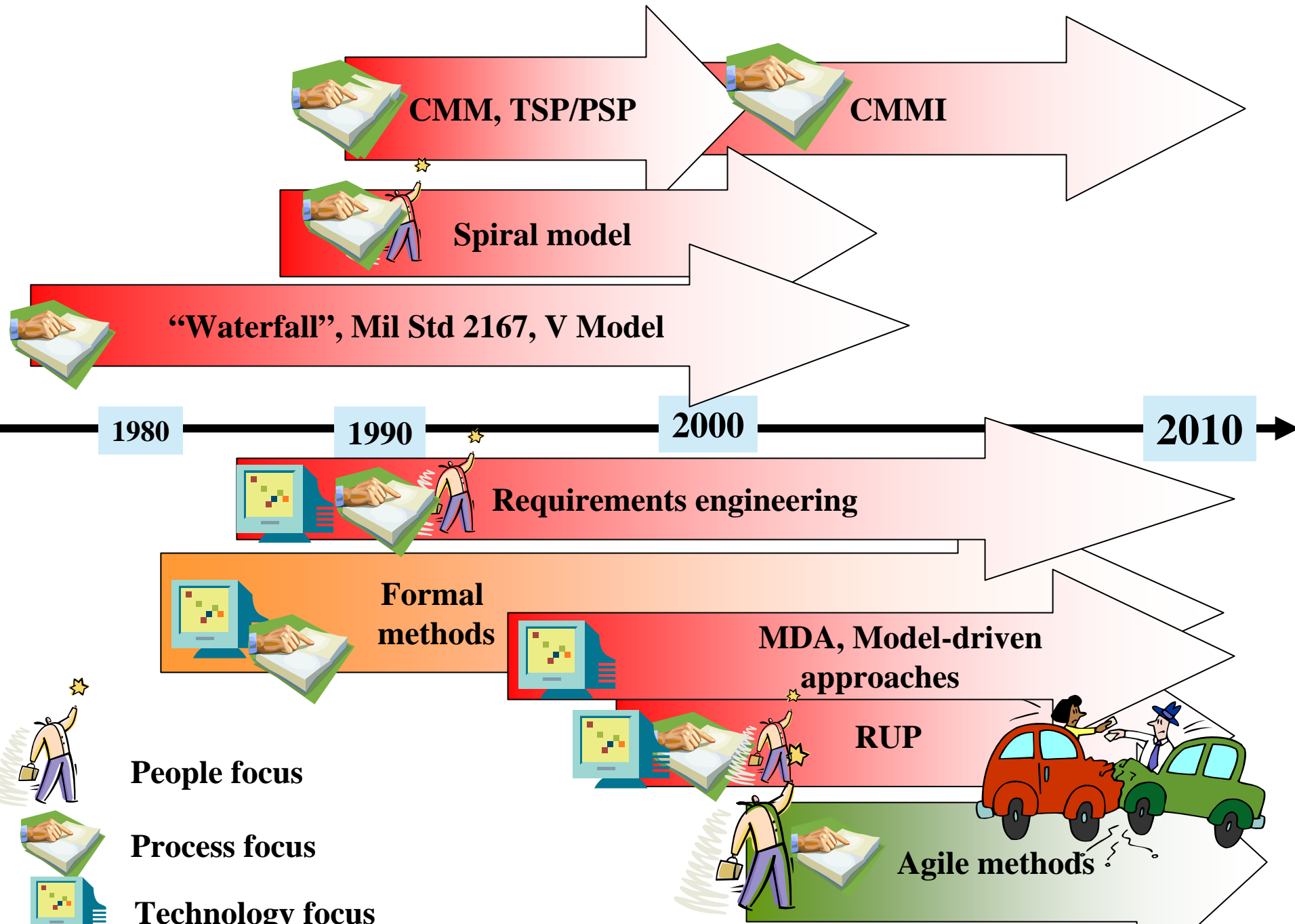
A brief history of process-related trends

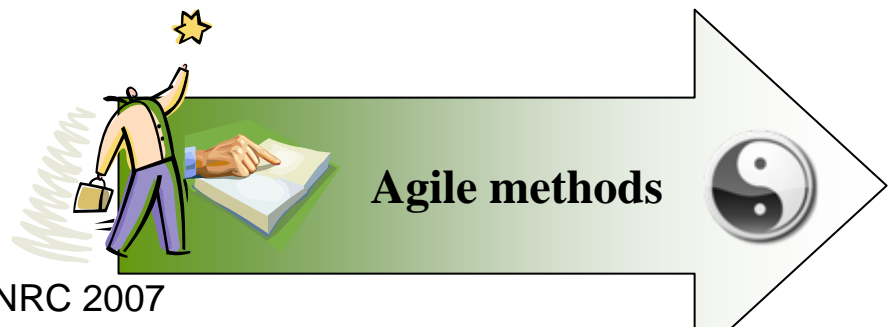
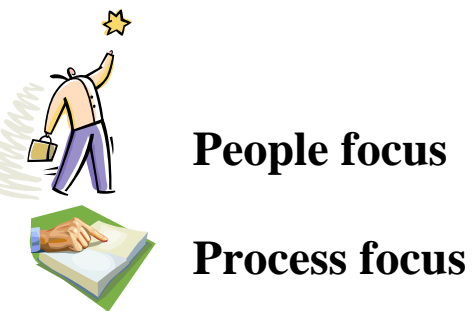
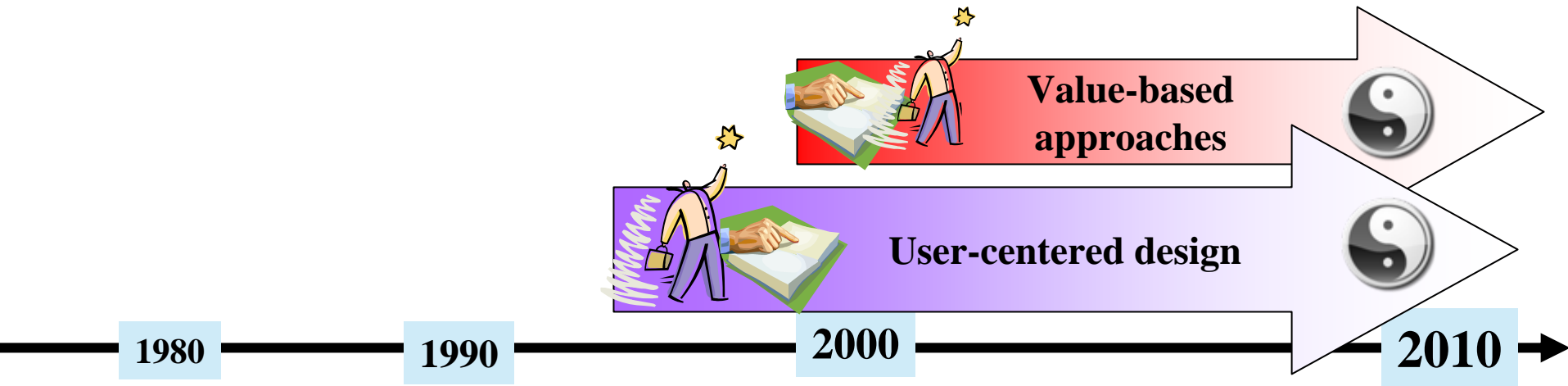












None complete, none definitive!



**Next: an era of
complex interactions**

**With moderation of trends
and interactions,
initial clashes
slowly transforming
into complementarities and
synergies!**



- **Reconciliation**
- **Integration**




People, people, people

[HireAttitudeSample.mp4](#)

(courtesy of Janice Singer)



Human-centricity



**Software development
as a team activity:**

**Motivated, creative, skilled individuals
who take pride in their work and work with others –
good engineers, craftsmen, managers leaders**



- **Alignment with how people do their job and use the software**
- **Soft skills along with hard skills**
- **Deep understanding of**
 - how software professionals do their work and interact with stakeholders
 - what customers want and how end users use the software
- **Collaboration**
 - working with minimal friction
 - trust, social interaction

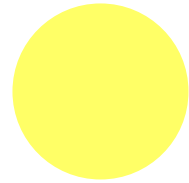
Human-centricity

- **Agile Manifesto**
 - *Individuals and interactions* over processes and tools
 - *Customer collaboration* over contract negotiation

“Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.”

“Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.”

“Business people and developers must work together daily throughout the project.”
- **CMMI Level 3 Process Area**
 - Organizational Training



But does *peopleware* singly guarantee success?

Technical orientation

The end product is software: software technology and competence with it matters

- **Use of appropriate technology (languages, programming paradigms, design methodologies, ...)**
- **Valuing technical solutions, and technical skills that support their effective implementation**
- **Development supported by appropriate infrastructure (tools, environments, frameworks)**
- **Respect for lowest abstraction level as the central, definitive artifact – *if you touch it, you must know it and care for it***
- ***High leverage of (if not complete reliance on) automation***



Technical orientation

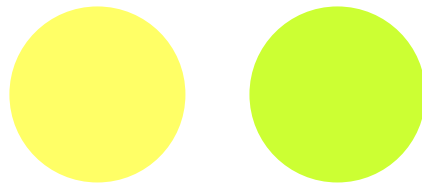
- **Agile Manifesto**

- **Working software** over *comprehensive documentation*

- “Continuous attention to technical excellence and good design enhances agility.”*

- **CMMI Level 3 Process Area**

- Technical Solution? - “... to design, develop, and implement solutions to requirements. Solutions, designs, and implementations encompass products, product components, and product-related lifecycle processes either singly or in combination as appropriate.”



***Technical orientation* adds meat to *peopleware*, but
doesn't any sustainable, serious, scalable
business needs rigor, both technically and
managerially?**

Discipline

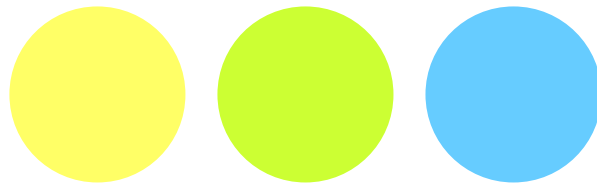
**For scalability, repeatability process is essential...
but
discipline \neq formal process**

- **Structuring flow and division of work:**
 - individually
 - within a team/project
 - within organization
- **Technical practices and activities**
 - Coding standards
 - Requirements techniques
 - Test-driven development
 - Architecting, design, modeling
 - Inspections
- **Management practices and activities**
 - Managing people, project, product artifacts
 - Iterative and incremental development, Scrum
 - Selection of CMMI process areas, including measurement and quantitative management, *as needed and appropriate* (more later...)

synergistic practices
vs.
*maturity levels,
compliance,
formality*

Discipline

- **Both process frameworks, such as CMMI and RUP, and flexible processes such as Extreme Programming and Scrum has a strong discipline focus**

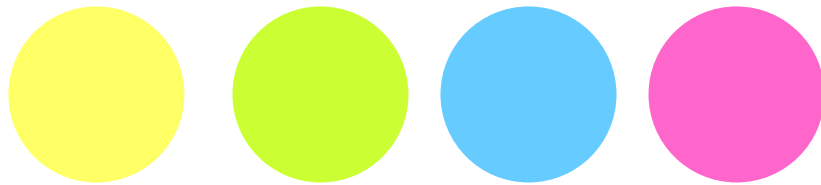


But *discipline* needs to be moderated by...

Pragmatism

A process activity should

- serve a clear purpose
 - support other essential process characteristics
 - satisfy a regulatory requirement
- be feasible in your context
- be adapted to your context
- be scalable in your context
- be compatible with your organization's culture and risk-taking preferences



**You can be very *pragmatic* with process choices,
but how do you know your pragmatism is paying
off as you expected?**

How can you take calculated risks?

Empiricism & experimentation

- **Supporting decisions with empirical knowledge: observations, data, measures**
- **Ability to connect outcomes and empirical knowledge**
- **Know what you are doing:**
 - when you are failing/succeeding
 - when you are improving /deteriorating
- **Support *learning* and *feedback***
- **Capture critical knowledge**
- **Collect and review evidence for future decision making**
 - from your own context
 - from external sources – *ask for help from researchers*
- **Support future planning**
- **Nurture an environment where alternative solutions and hypotheses can be tested safely and at low cost**



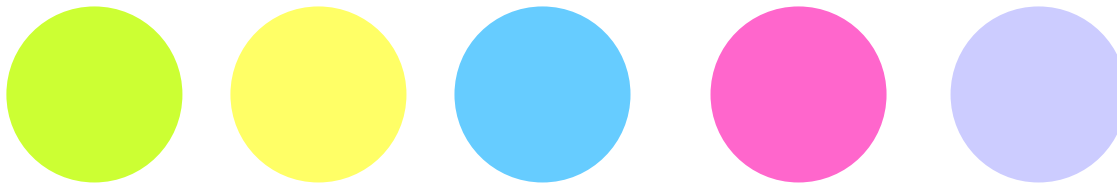
Empiricism & experimentation

- **Activities that support E&E**
 - Measurement (quality, effort, tools usage, ...)
 - Project tracking
 - Prototyping, *spiking* (experimentation in context)
 - Documentation, especially of design decisions/rationale
 - Literature review
- **Activities that benefit from E&E**
 - Estimation
 - Project planning and management
 - Risk management
 - Contract negotiation and pricing
 - Decision-making in general

Empiricism & experimentation

- **CMMI Level 2 Process Area**
 - Process and Product Quality Assurance
 - Measurement and Analysis
 - Project Monitoring and Control

- **Agile Manifesto**
 - “Working software is the primary measure of progress”?
 - Not explicit on measurement
 - Implied in certain practices/implementations of popular methods
 - Stressed in practice



**And everything needs a purpose, justification,
an end goal...**



Value orientation

- **Postulating, understanding, and articulating:**
 - bottom-line implications of process activities: how do they support the bottom line, how do they support each other to improve bottom line?
 - what really matters (value propositions), to whom, and why?
- **Reconciling and balancing value propositions of different stakeholders**
- **Defining and measuring success in terms of value generation**
- **Aligning incentives and value generation**

Value orientation

Value creation through software process...

- **Improve cost effectiveness**
 - increase **development productivity**
 - reduce **cost of poor quality** (increase quality?)
 - ⇒ Increase **real productivity**

ProductionOutput

TotalCostOfDevelopment = CostOfPoorQuality + ProductionEffort



Value orientation

Value creation through software process...

- **Increase present value of receiver benefits**
 - Maximize business value to end-user, customers through
 - understanding what end-users, customers care about
 - methods that enable early and frequent delivery
 - effective prioritization: emphasis on high-value features
- **Maximize flexibility and learning where uncertainty is high**
 - *Flexibility is valuable only under uncertainty!*
- **Monitor and mitigate foreseeable risks through sufficient assessment and planning**



Value orientation

- **Agile Manifesto**

- **Working software** over comprehensive documentation

- “Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.”

- *Business people and developers must work together daily throughout the project.*

- **Responding to change** over following a plan

- **CMMI Level 3 Process Area**

- Risk Management

- *But **not** Level 2 Process and Product Quality Assurance*

Essential characteristics of a successful software development environment

H

T

D

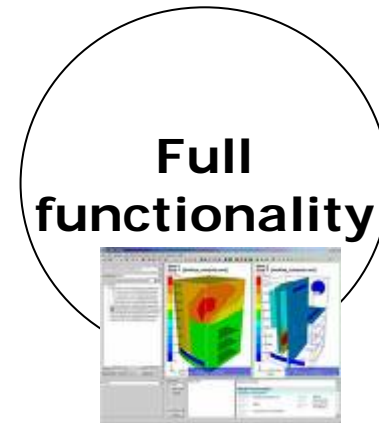
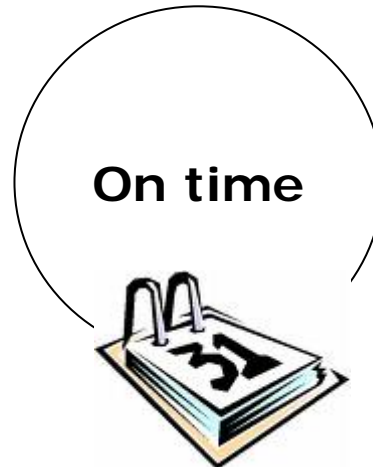
P

E

V



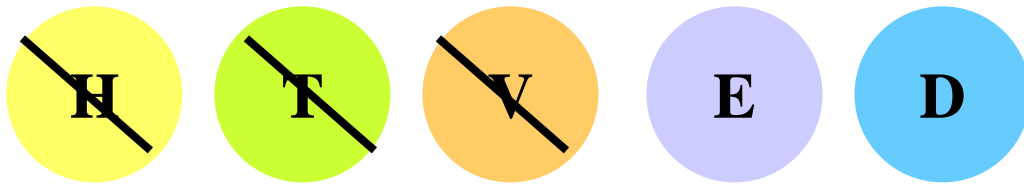
Traditional success criteria



- Reported success rates are low: $< 60\%$ \Rightarrow software crisis
- Goal of software process: *repeatability* and *predictability*

Management need
or
real success?

What about a project that was:
over-budget,
delayed,
with 50% of original functionality,
but *successfully deployed to 10,000
corporate users?*



PSP and TSP

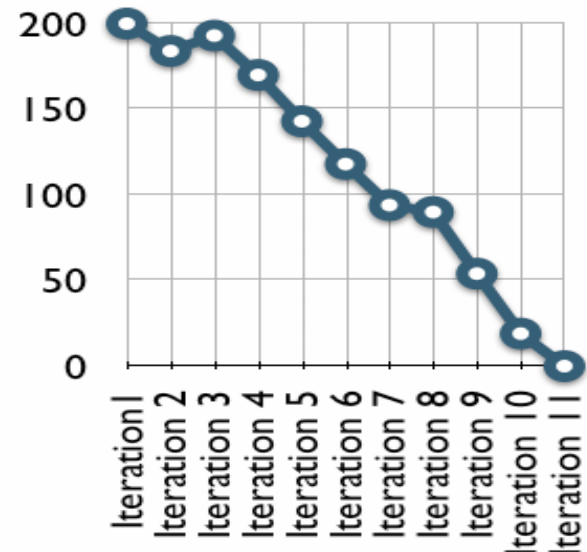
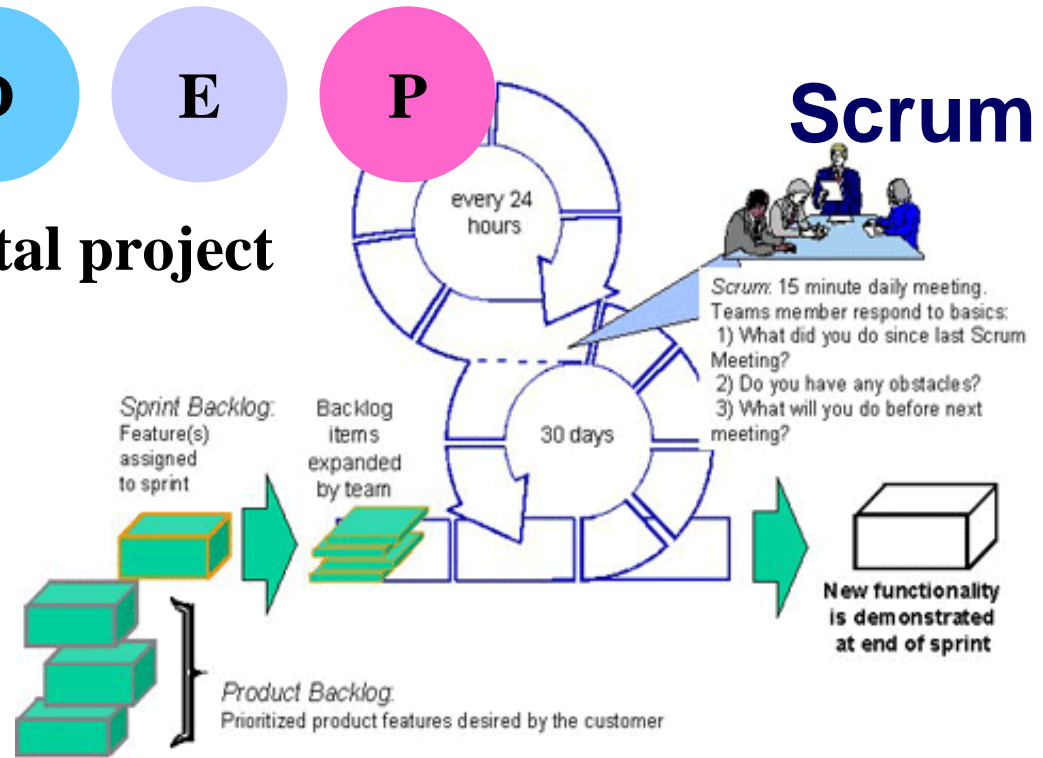
CMM Level 5 compliance for individuals and small teams

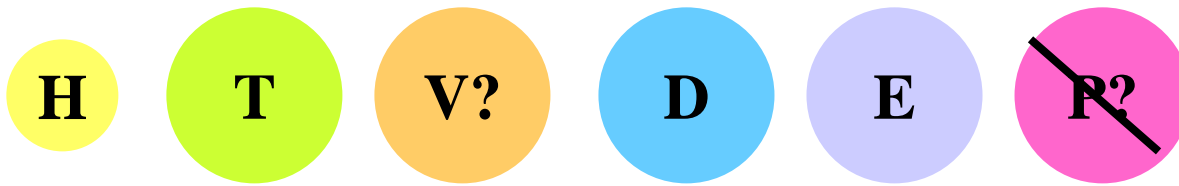
- **For small organizations, small teams, small projects, aspirations of big ceremonial process is overkill!**
- **An example of where empiricism and discipline stamp everything else.**
- **Probably explaining failure of adoption despite heavy marketing.**

H ~~**T**~~ **V** **D** **E** **P** **Scrum**

An iterative & incremental project management wrapper...

- Robust for both uncertain and stable environments
- Planning, estimation, measurement:
 - product backlog; roles & responsibilities; velocity, burn-down charts;
 - earned-value PM compatible
- Scalable versions emerging, adoption exploding

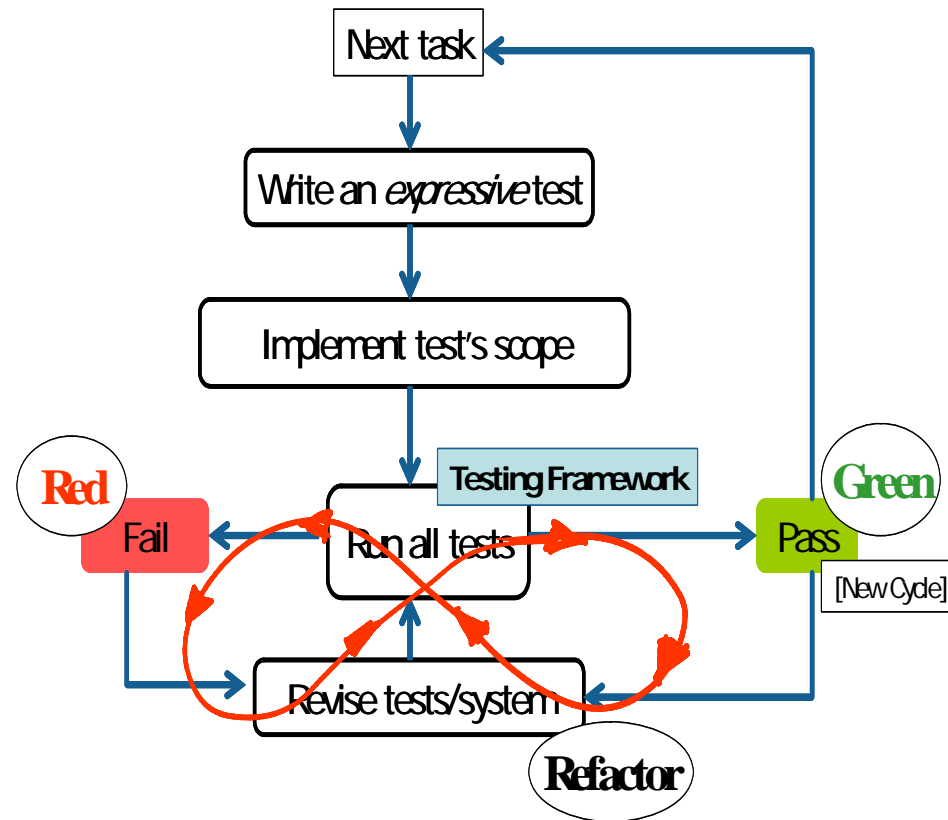


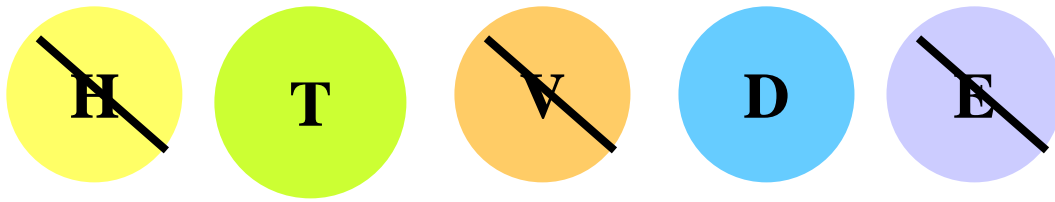


Test-driven development

An incremental technical development practice guided by “tests”

- Visible, measurable progress
- Enable change
- Instant feedback
- Task focus
- Quality benefits
- Automated regression testing
- Experimentation support
- Applicable at abstraction multiple-levels
- Possible productivity penalty?
- Issues in legacy contexts?
- Difficult to apply for some?

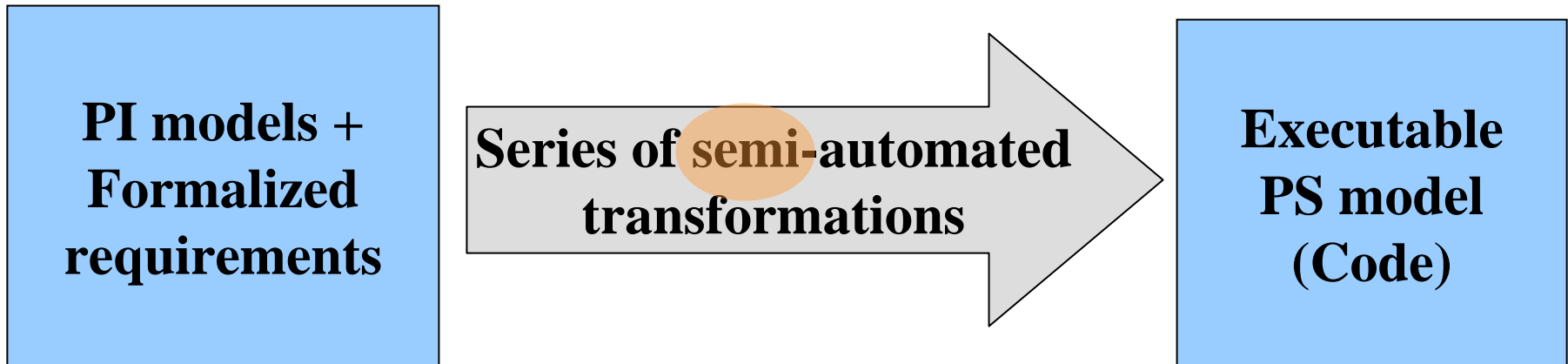




MDA

From models to executable code...

- **MDA is not a process *per se*, but marketed almost as one**
- **As much as a heavy-weight technical solution can influence dictate process**
- **Favors a big-bang approach; confusing step-wise refinement with iterative & incremental**
- **No attention to flexibility: if technology matures, potentially high value in low uncertainty environments**
- **Little attention to lower abstraction levels**





Fixed-price, single-stage, fixed-scope contract

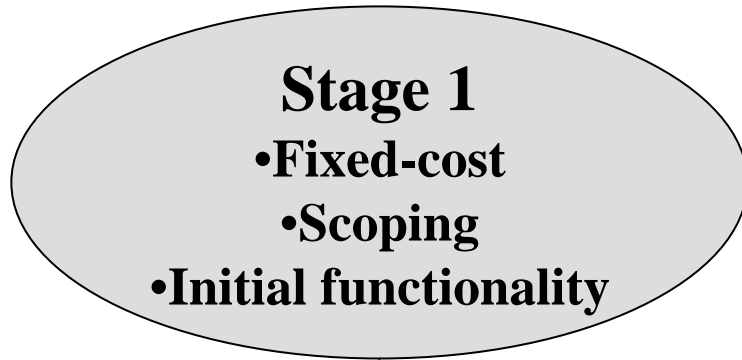
Under uncertainty

(*developer*: technology; *customer*: requirements)

- Early scope definition – *developer's risk premium*
- Excess scope – *customer's risk premium*
- Price includes developer's risk premium who bears budget risk
- No flexibility, no feedback, no learning
- Up-front effort focuses on contract negotiation, scope
- Both parties in a straight jacket when things change



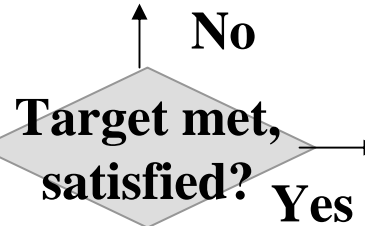
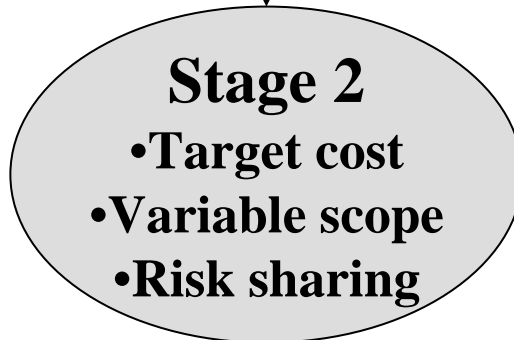
Target cost, multi-stage variable-scope contract



- Continued collaboration to limit opportunism
- Flexibility
- Feedback & learning
- Aligned incentives

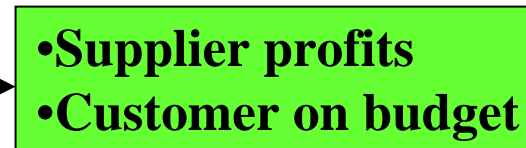


No



No

Yes



**Essentials are both
mutually reinforcing
and moderating**

