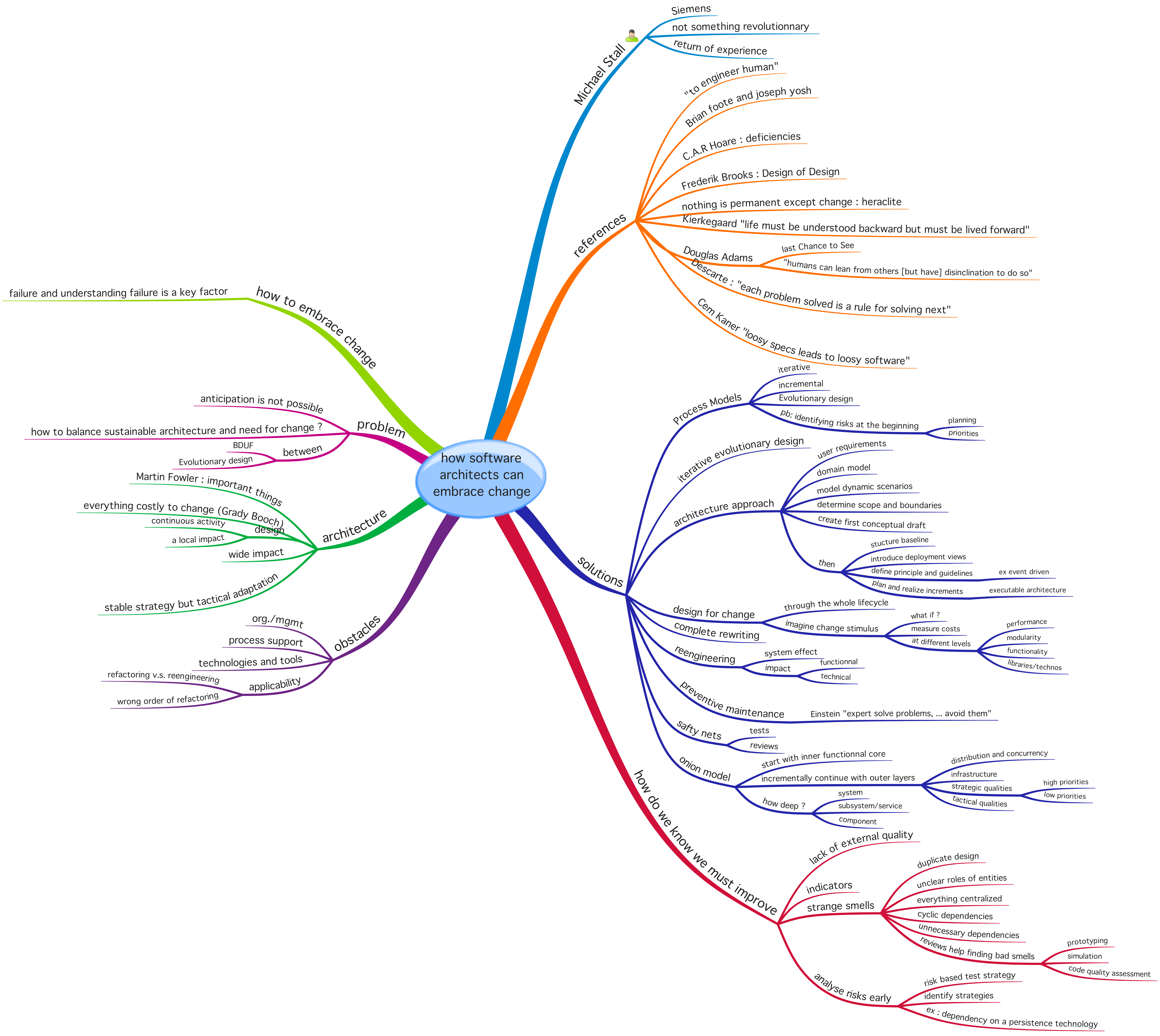

how software architects can embrace change

- Michael Stall
  - Siemens
  - not something revolutionnary
  - return of experience
- references
  - "to engineer human"
  - Brian foote and joseph yosh
  - C.A.R Hoare : deficiencies
  - Frederik Brooks : Design of Design
  - nothing is permanent except change : heraclite
  - Kierkegaard "life must be understood backward but must be lived forward"
  - Douglas Adams
    - last Chance to See
    - "humans can lean from others [but have] disinclination to do so"
  - Descarte : "each problem solved is a rule for solving next"
  - Cem Kaner "loosy specs leads to loosy software"
- solutions
  - Process Models
    - iterative
    - incremental
    - Evolutionary design
    - pb: identifying risks at the beginning
      - planning
      - priorities
  - iterative evolutionary design
  - architecture approach
    - user requirements
    - domain model
    - model dynamic scenarios
    - determine scope and boundaries
    - create first conceptual draft
    - then
      - stucture baseline
      - introduce deployment views
      - define principle and guidelines
        - ex event driven
      - plan and realize increments
        - executable architecture
  - design for change
    - through the whole lifecycle
    - imagine change stimulus
      - what if ?
      - measure costs
      - at different levels
        - performance
        - modularity
        - functionality
        - libraries/technos
  - complete rewriting
  - reengineering
    - system effect
    - impact
      - functionnal
      - technical
  - preventive maintenance
    - Einstein "expert solve problems, ... avoid them"
  - safty nets
    - tests
    - reviews
  - onion model
    - start with inner functionnal core
    - incrementally continue with outer layers
      - distribution and concurrency
      - infrastructure
      - strategic qualities
        - high priorities
        - low priorities
      - tactical qualities
    - how deep ?
      - system
      - subsystem/service
      - component
- how do we know we must improve
  - lack of external quality
  - indicators
  - strange smells
    - duplicate design
    - unclear roles of entities
    - everything centralized
    - cyclic dependencies
    - unnecessary dependencies
    - reviews help finding bad smells
      - prototyping
      - simulation
      - code quality assessment
  - analyse risks early
    - risk based test strategy
    - identify strategies
    - ex : dependency on a persistence technology
- how to embrace change
  - failure and understanding failure is a key factor
- problem
  - anticipation is not possible
  - how to balance sustainable architecture and need for change ?
  - between
    - BDUF
    - Evolutionary design
- architecture
  - Martin Fowler : important things
  - everything costly to change (Grady Booch)
  - design
    - continuous activity
    - a local impact
  - wide impact
  - stable strategy but tactical adaptation
- obstacles
  - org./mgmt
  - process support
  - technologies and tools
  - applicability
    - refactoring v.s. reengineering
    - wrong order of refactoring