CS 428 Inside-Out: An SQA-Oriented SDL

Fall 2022 – Bruce F. Webster

The Problem

- Software quality assurance (SQA) is the ‘red-headed stepchild’ of IT management: underfunded, low prestige, treated as an afterthought
- ‘SQA’ is often (falsely) equated with just ‘testing’
- SQA is often seen as filling that brief gap between development and production and thus introduced late in the lifecycle
- SQA is often the first thing to get squeezed or cut back due to schedule and/or budget

The Results

- IT projects end up taking longer and costing more than if proper SQA had been applied
  - Brooks: 50% spent on testing [SQA] whether you plan for it or not
  - Glass: defects & missing requirements cost more to fix the later in the cycle you are
- Systems in production are less reliable and cost more to support

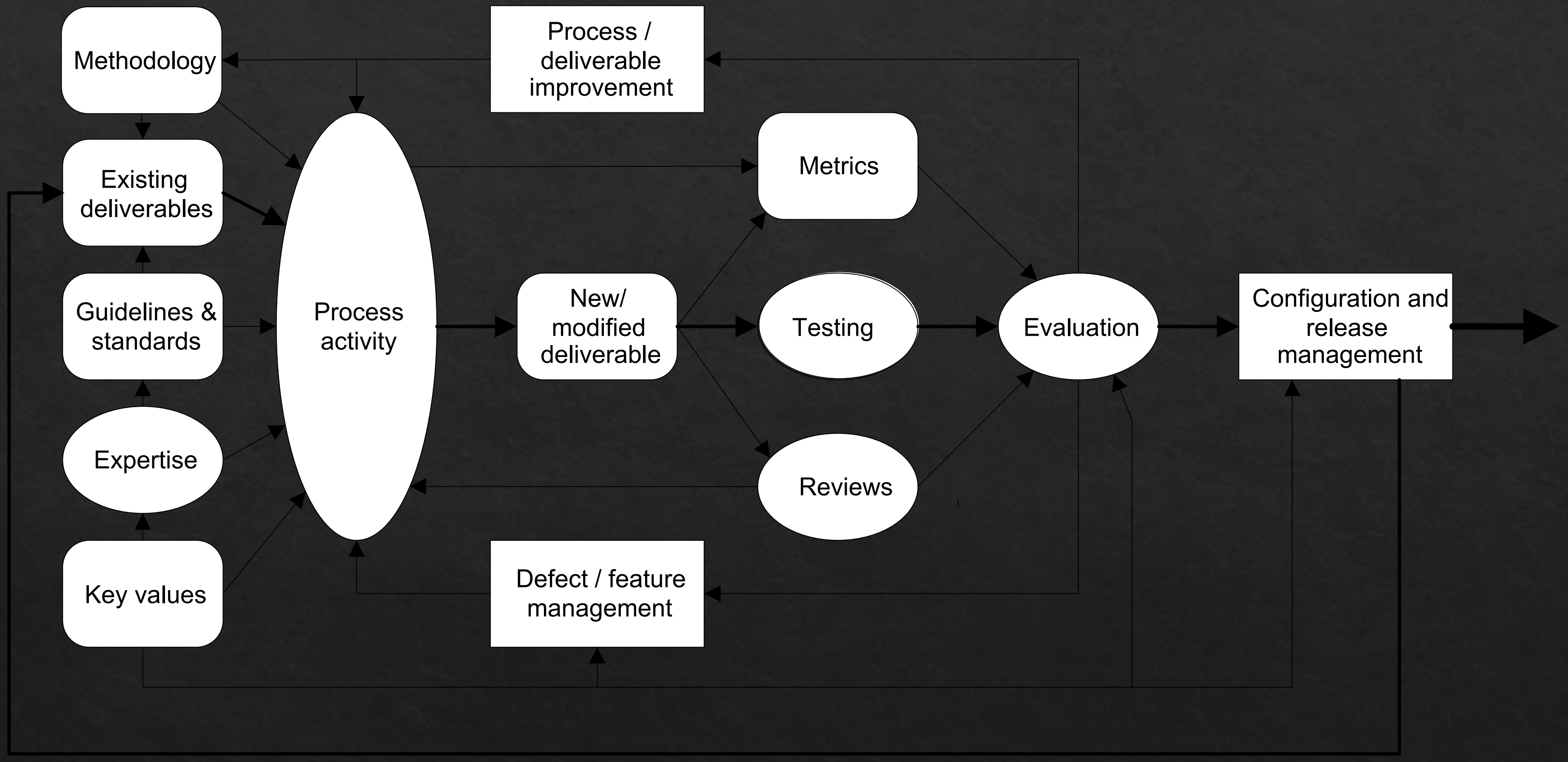
Typical Software Lifecycle Views

- Predictive: waterfall and derivatives
- Adaptive: iterative/incremental/agile
- Methodologies tend to fall into one of these two camps
- In either case, "testing" (not SQA) is usually seen to be just a phase in the lifecycle
- There tends to be less focus (if any) on other SQA activities

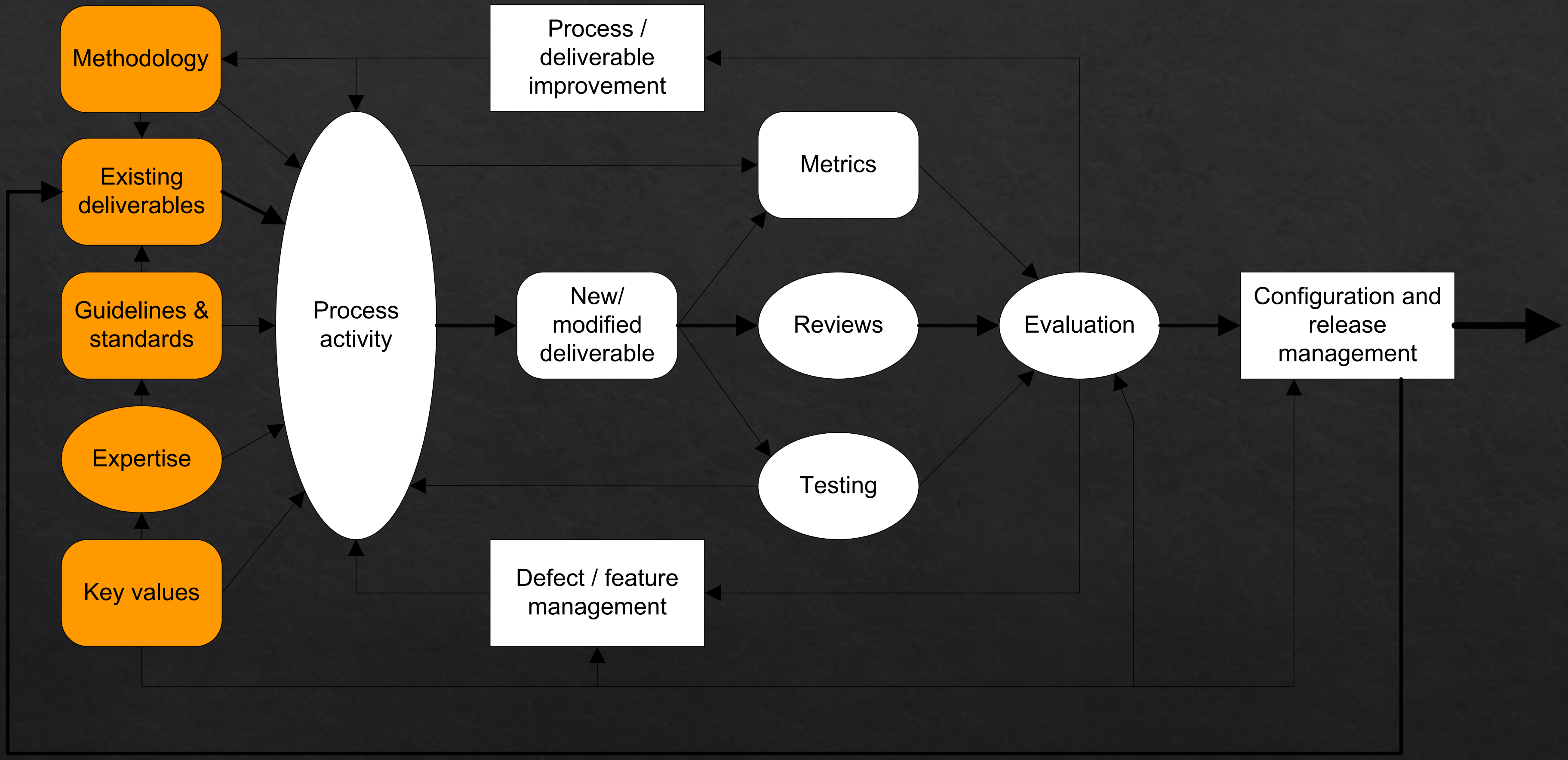
Turning the SDL inside-Out

- Don’t focus on changing the SDL or methodology itself
- Instead, consciously surround each ‘process activity’ (deliverable creation) in your chosen SDL/methodology with the supporting SQA activities, artifacts, and processes
- Goal: carry out quality efforts each step along the way

Inside-Out view of SQA


Methodology
Process / deliverable improvement
Existing deliverables
Metrics
Guidelines & standards
Process activity
New/ modified deliverable
Testing
Evaluation
Configuration and release management
Expertise
Reviews
Key values
Defect / feature management

Inputs


Methodology
Existing deliverables
Guidelines & standards
Expertise
Key values
Process activity
Process / deliverable improvement
Metrics
New/ modified deliverable
Reviews
Testing
Evaluation
Configuration and release management
Defect / feature management

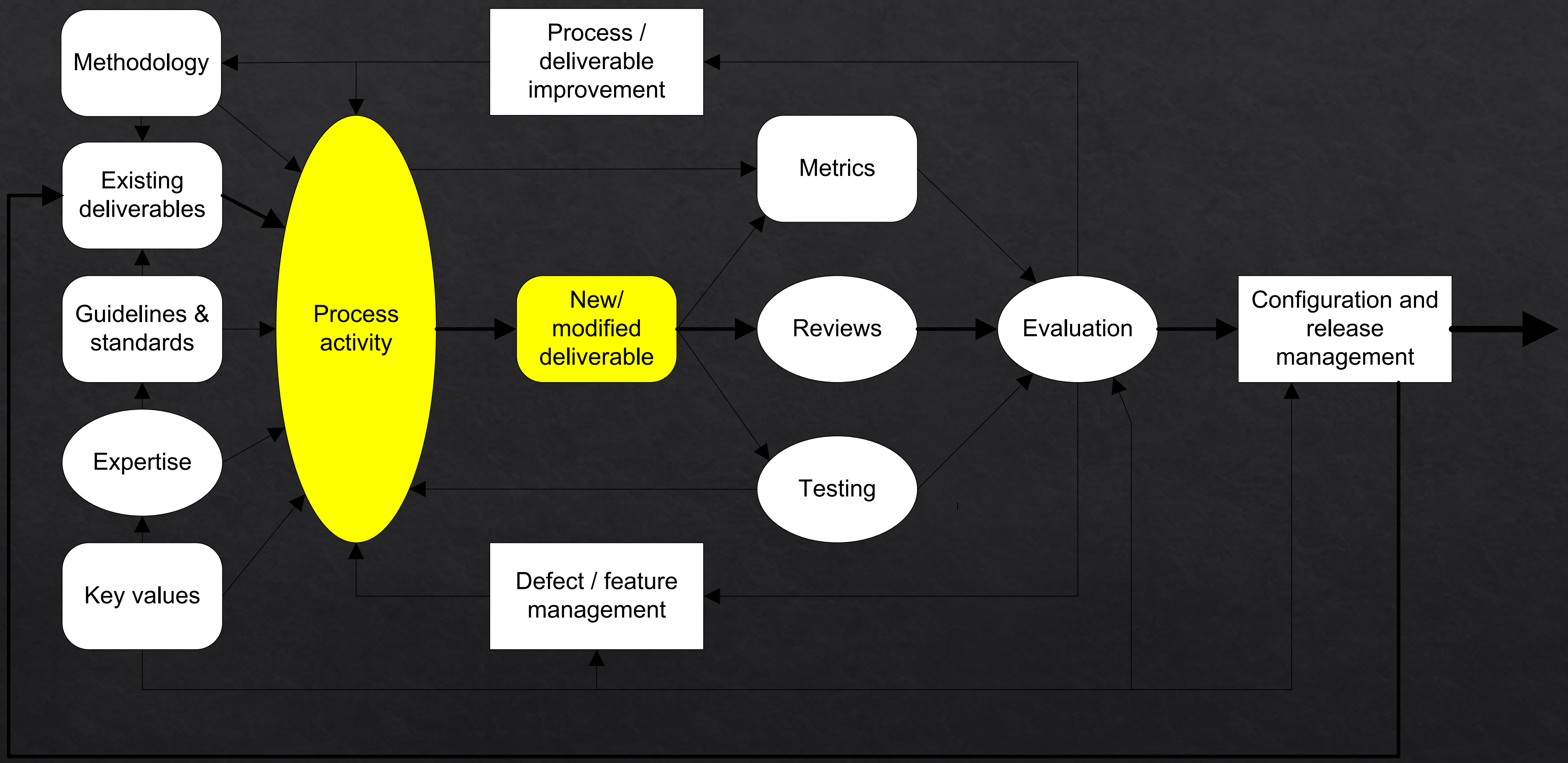
Inputs

- **Key values**: business drivers, enterprise architecture, market forces, key performance indicators (KPIs), service level agreements (SLAs)
- **Expertise**: subject matter, technical, methodology, language
- **Standards and guidelines**: appropriate to deliverables under development
- **Existing deliverables**:
  - Use standardized templates for brand-new deliverables
  - Improve existing deliverables (functionality, reliability, performance)
  - Use existing deliverables to create or improve other deliverables
- **Methodology**: your choice, based on needs, personnel, experience

Key Quality Attributes

- Weinberg: “Quality is value to some person(s).”
- Key quality attributes that you must choose among, prioritize, and scale to an acceptable level:
  - Reliability
  - Performance
  - Functionality
  - Compatibility
  - Security
  - Lifespan
  - Deployment
  - Support
  - Cost
- The key issue is “acceptable” – acceptable to the person(s) who have to use, support, and market the system under development

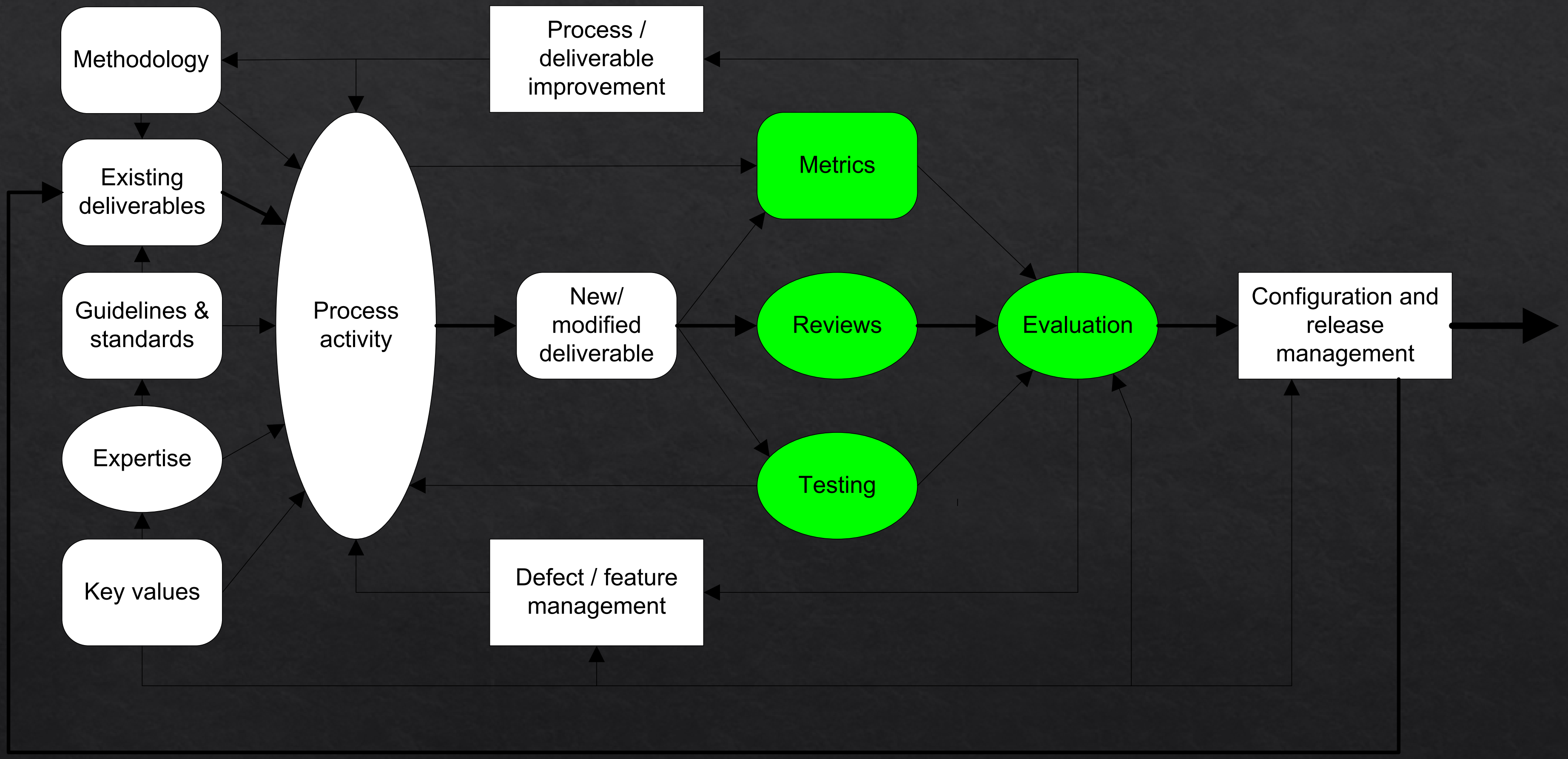
Process Activity (Lifecycle/Methodology)


Methodology
Process / deliverable improvement
Existing deliverables
Metrics
Guidelines & standards
Process activity
New/ modified deliverable
Reviews
Evaluation
Configuration and release management
Expertise
Testing
Key values
Defect / feature management

Process Activity

- "Process activity" represents non-SQA software development activities as dictated by your methodology or lifecycle choices:
  - Analysis
  - Specification/Requirements
  - Architecture & design
  - Development (including graphics, database, etc.)
  - Deployment
  - Production
- The nature of the inputs and assessment depend upon the activity
- As does the result: new or modified deliverables

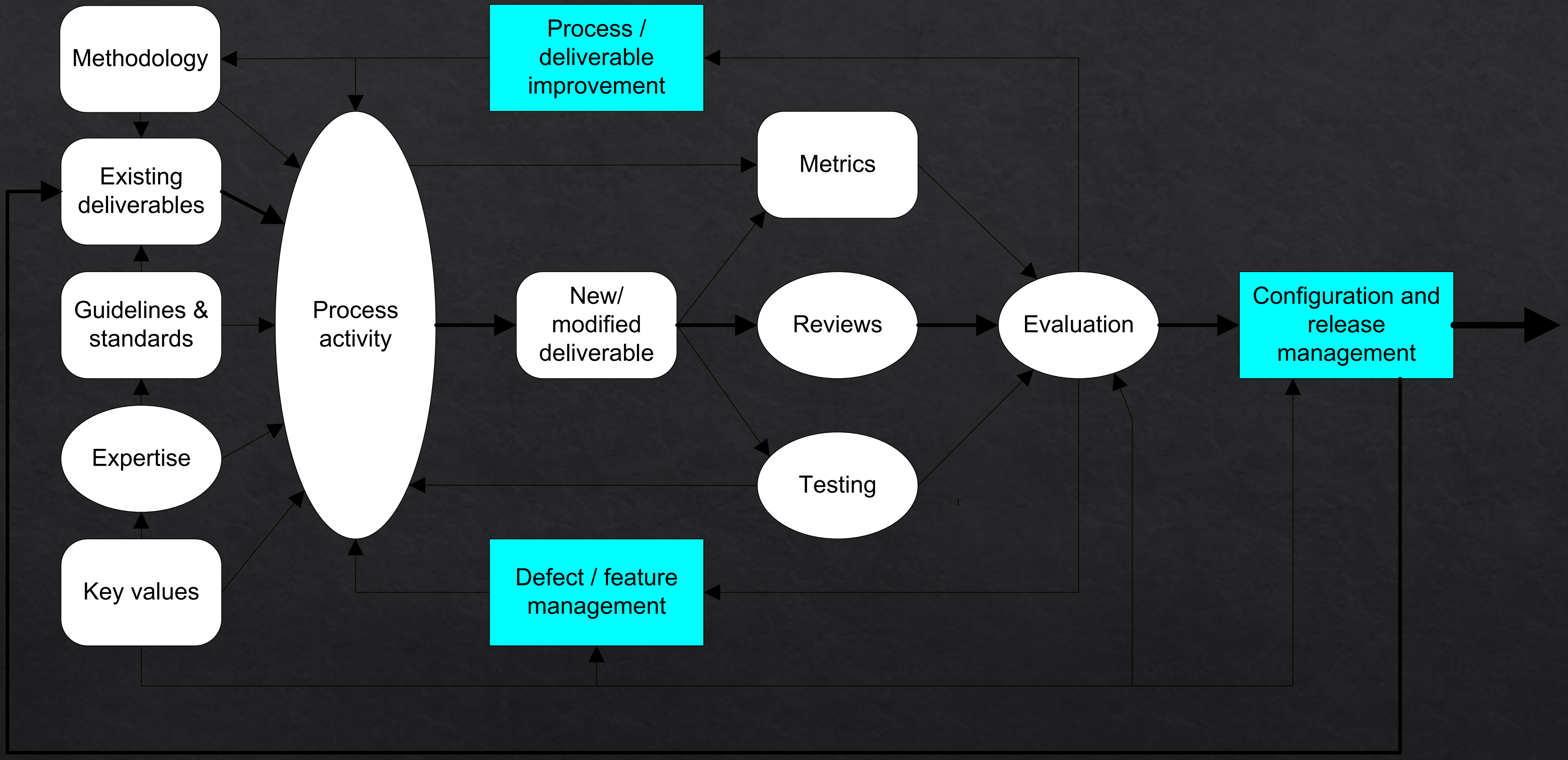
Assessment


Methodology
Process / deliverable improvement
Existing deliverables
Metrics
Guidelines & standards
Process activity
New/ modified deliverable
Reviews
Evaluation
Configuration and release management
Expertise
Testing
Key values
Defect / feature management

Assessment

- Any or all of three types, as appropriate
  - Metrics (from process activities and resulting deliverables)
    - Where appropriate and useful
    - Remember: objective, repeatable, automated, predictive/informative
  - Reviews, walkthroughs, and other forms of examination
  - Testing – again, where appropriate and useful
- Evaluation: human judgment as to the meaning of the results
  - Project/team/organization key values help determine that meaning

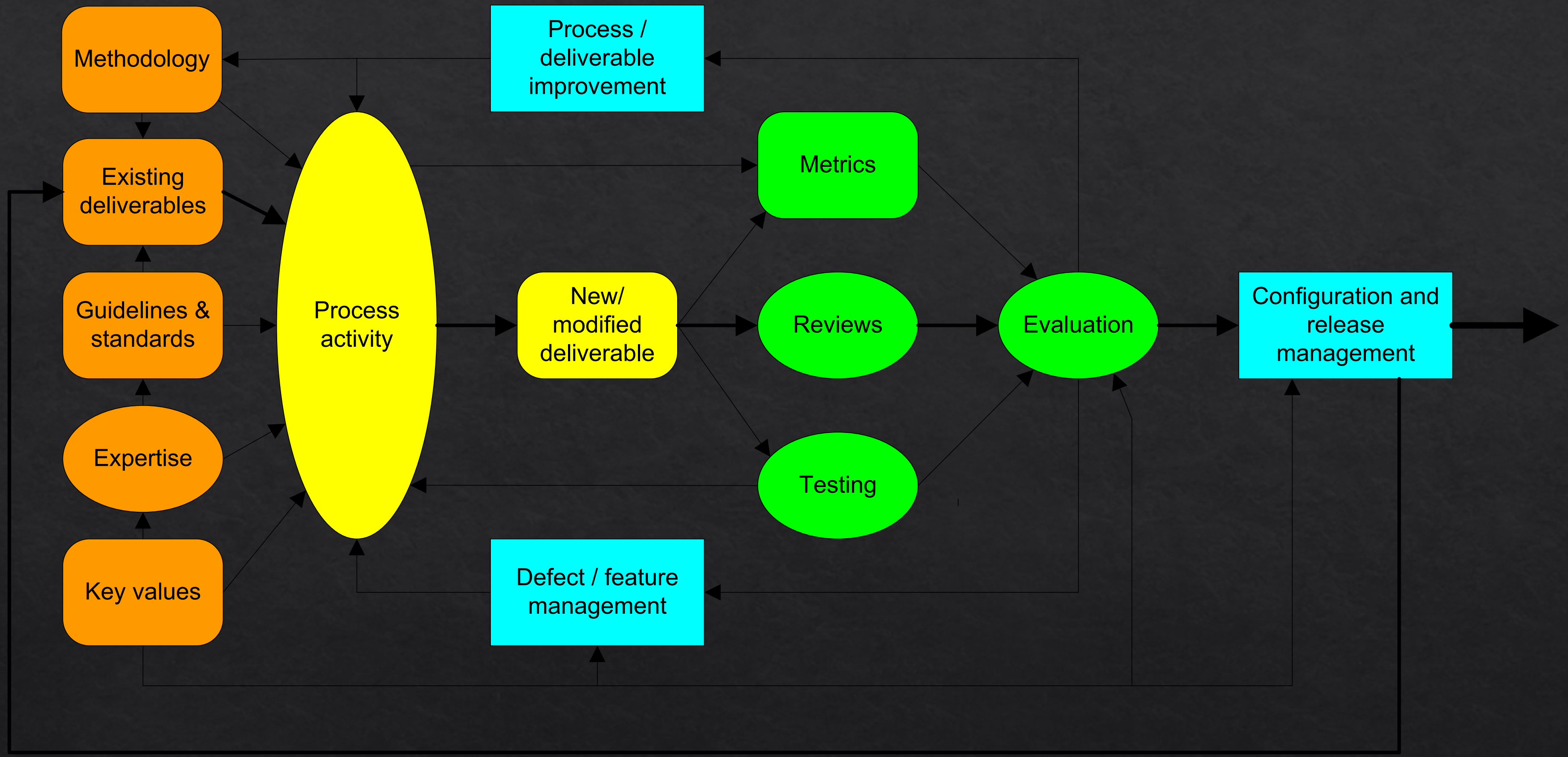
Feedback and Control


Methodology
Process / deliverable improvement
Existing deliverables
Metrics
Guidelines & standards
Process activity
New/ modified deliverable
Reviews
Evaluation
Configuration and release management
Expertise
Testing
Key values
Defect / feature management

Feedback and Control

- Defect/feature management
  - Prioritization and assignment
  - May involve a change control board (formal or informal)
- Configuration/release management
  - Digital management of all deliverables and artifacts
  - Gateway to shipping/production
- Process/deliverable improvement
  - Seeking to increase process quality and efficiency

Inside-Out (Again)


Methodology
Existing deliverables
Guidelines & standards
Expertise
Key values
Process / deliverable improvement
Process activity
New/ modified deliverable
Metrics
Reviews
Testing
Evaluation
Configuration and release management
Defect / feature management

Why Inside-Out?

- To encourage (or force) a more comprehensive and more integrated view of SQA
- To shorten the overall development time/costs and to reduce production/post-shipping costs
- To do the right things as early as possible in the software development lifecycle, thus reducing risks

- Goal: straightforward document for internal communication and alignment
- Should tie back to **requirements and design**
- Should check for **reliability**, **performance**, **functionality**
- Should indicate what tests are being done and when they are done (or repeated)
- Should indicate what constitutes success for each test
- Should include some form of user-acceptance testing
- Get feedback, input from entire team
- First draft due by midnight Saturday (10/15), but should be revised through the rest of the semester

Building your test plan

- By midnight Saturday (10/15)
  - Test plan up on team wiki
  - Status report up on team wiki
- By class next week (10/24)
  - Read *Facts & Fallacies of Software Engineering*, chapter 1
  - ***Start*** Webster #6 (you have 4 weeks to read these)
- NEXT WEEK (10/24): PROTOTYPE DEMOS IN CLASS
- Remember: **work-in-progress demo in four weeks** (11/14)
- Remember: **midterm in five weeks** (11/21)

FOR THIS COMING WEEK