Quality Metrics in Requirements

- Quality in a requirements specification is based on how well the specification expresses client wants and needs
- Collect a number of values to determine quality
  - Called metrics
  - Most valuable when target values (indicating when adequate quality is reached) are determined in advance
- QA organization is often included in requirements analysis

Attributes that Promote Quality

- Requirements are **accessible**
- Specification is **comprehensive**
- Each requirement is **understandable**
- Each requirement is **unambiguous**
- Specification is **consistent**
- Requirements are **effectively prioritized**
- Requirements are **secure**
- Specification is **self-complete**
- Each requirement is **testable**
- Each requirement is **traceable**

Accessibility

- Need numbering scheme to identify requirements
  - Allow determining if requirement has been implemented
  - Allow tracing to code
- Considerations
  - Know where high-level and detailed requirements are listed
  - Detailed requirements are grouped by relevant high-level requirement
  - Can requirements be searched by relevant factors
    - Keyword, use case, interface type, user type,…

Accessibility (cont.)

- Good metric is average time to access a detailed requirement
  - Choose random sample of >= 150 requirements
  - Time people trying to find sample requirements
  - Score based on performance compared to organization norms
    - 0 score for “extremely long” access times
    - 10 score for “extremely fast”

Comprehensiveness

- % of total requirements specified
- Could have client evaluate this but
  - Time commitment is a problem
  - Contradictory stakeholder desires a problem
- If system is implemented iteratively, detailed requirements for later iterations may not yet be done
  - Can summarize requirements that haven't been detailed yet
- IEEE defines completeness using formula with 18 different quantities and 10 weights

Understandability

- Measure by taking random sample of relevant people (clients and designers) and asking opinions
- Considerations to improve understandability
  - Written to level of typical reader understanding?
  - Use vocabulary of the client problem domain?
  - Describe only external behavior?
  - Don’t say how to solve problem or how to design system

Unambiguity

- Sample set of readers, have each score each requirement
  - 2 if exactly one clear meaning
  - 0 if multiple possible interpretations
  - 1 otherwise
  - Sum ambiguity measure over all detailed requirements, divide by 2*number of requirements

Consistency

- One approach
  - Sample >= 150 detailed requirements
  - Evaluate each against ALL other requirements to see if another requirement contradicts it
  - Score 0, 1, 2
- Measures only pairwise inconsistency
  - Also possible to have inconsistency among higher order groups

Prioritization

- Reprioritize after first release of system, since clients often change their mind
- Rank requirements as high, medium, or low priority
  - Good prioritization would have each group about the same size
  - Calculate variability measure

If T is the total number of detailed requirements

$V = 100 * (T - |T/3 - high| - |T/3 - medium| - |T/3 - low|) / T$

Prioritization (cont.)

- For example, suppose there are 900 requirements
  - If high, medium and low categories each have 300, V = 100*(900 – 0 – 0 – 0)/900 = 100%
  - If high = 700, medium = 100, and low = 100 then V= 100 * (900 – 400 – 200 – 200)/900 = 11.1%

Security

- Some security requirements are different from all other requirements – they say what the system **should not** do (**mis**use cases)
  - An automated user of the application enters a known user ID and more than 10 passwords per second
  - A user accesses more than 30 customer records in a single session and transmits those to another address within 10 seconds of access
- As opposed to
  - All passwords shall be unavailable except to the system administrator

Security (cont.)

- Some things to check
  - Evaluate the system for places where intrusion might be possible, and verify that concrete security requirements are identified
  - Where appropriate, have data confidentiality requirements been specified
  - Has user identity and password security been specified
  - Has ownership of files and data access been specified
  - Has encryption been specified where appropriate
  - Has protection from known security exploits been specified (e.g., SQL injection)

Self Completeness

- Requirements specification contains every requirement necessitated by any other requirement
  - For calendar app, “app will retain all information entered for each appointment”
    - Implies requirement describing how information is entered
  - Need to follow implication chains to verify
- Possible metric
  - # of missing necessary requirements/# of requirements

Testability

- Can’t determine whether untestable requirements have been met
- Include tests for requirements and count percentage of requirements accompanied by tests
- Checklist to improve testability includes
  - Is requirement clear enough to identify test data and actions? Do it!
  - Would two different client representatives agree that proposed tests really do test the requirement?

Traceability

- Forward traceability from requirement to
  - Design element that accommodates it
  - Code that implements it
  - Test that verifies it is satisfied
- Metric might be to rate traceability of each requirement on a 0 – 2 scale
  - Divide sum of ratings of all requirements by 2*total number of detailed requirements

SE Ethics - Public

- SEs shall act consistently with the public interest
  - Accept full responsibility for their own work
  - Moderate the interests of the SE, employer, client, and users with the public good
  - Approve SW only if they have a well-founded belief that it is safe, meets specifications, passes appropriate tests, and does not diminish quality of life, diminish privacy, or harm the environment. The ultimate effect of all work should be to the public good.

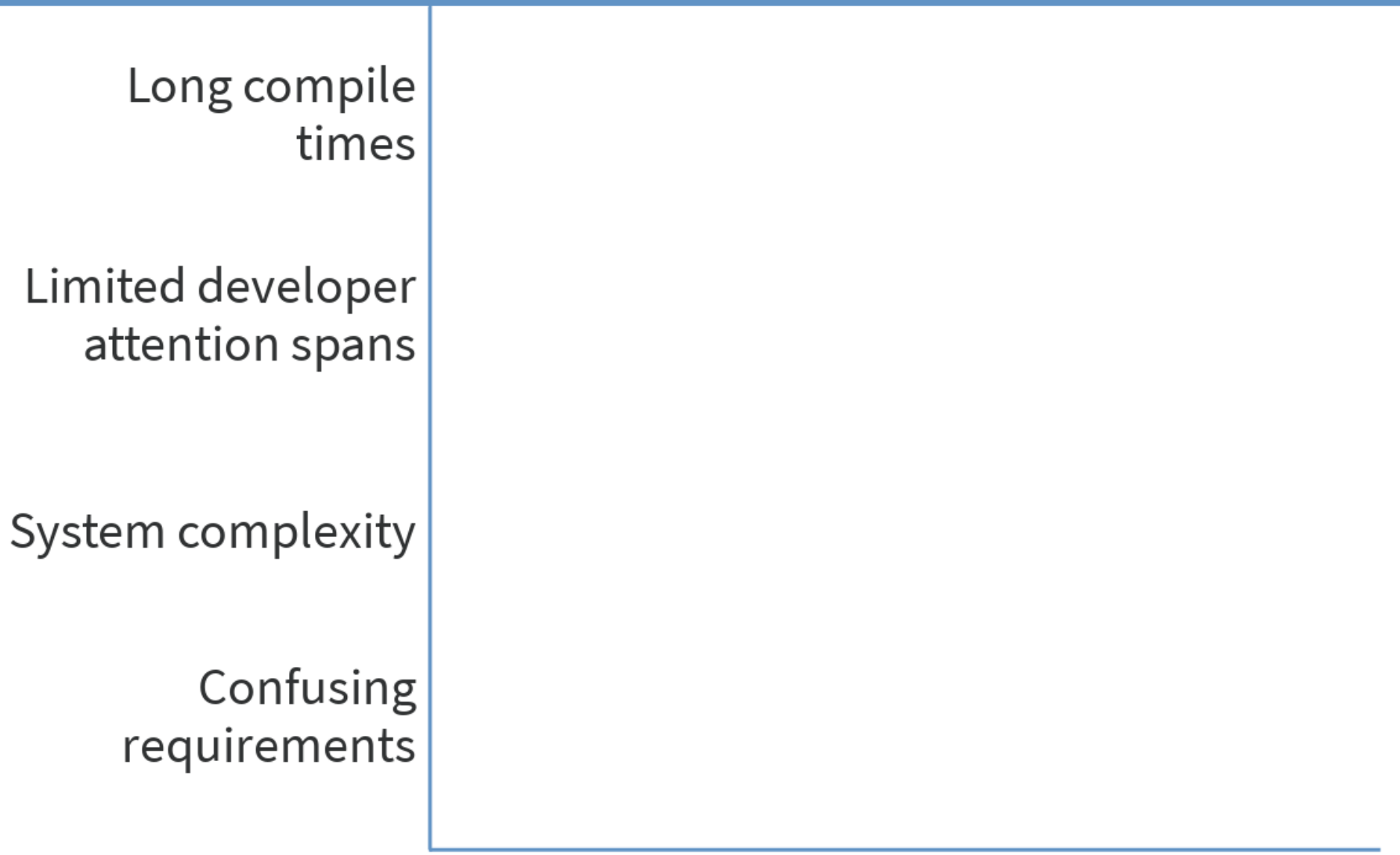
SE Ethics – Public (cont.)

- SEs shall act consistently with the public interest
  - Disclose to appropriate persons or authorities any actual or potential danger to the user, the public, or the environment, that they reasonably believe to be associated with software or related documents.
  - Cooperate in efforts to address matters of grave public concern caused by software, its installation, maintenance, support or documentation
  - Be fair and avoid deception in all statements, particularly public ones, concerning software or related documents, methods and tools.

SE Ethics – Public (cont.)

- SEs shall act consistently with the public interest
  - Consider issues of physical disabilities, allocation of resources, economic disadvantage and other factors that can diminish access to the benefits of software.
  - Be encouraged to volunteer professional skills to good causes and contribute to public education concerning the discipline

Modularization is an approach to deal with



How do high-level requirements relate to low-level requirements?

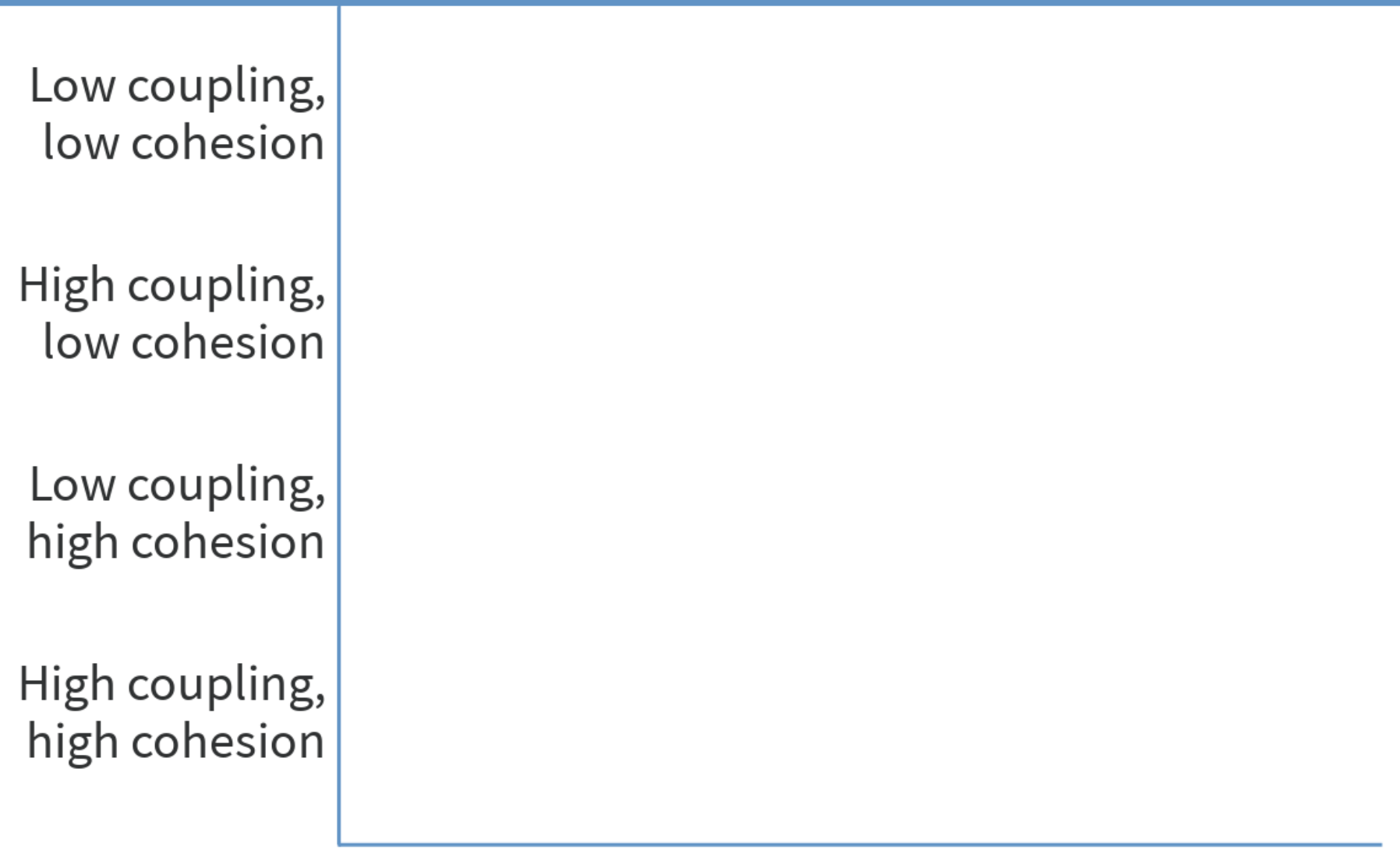
They cover the same things but in different levels of details for different audiences

They're completely unrelated

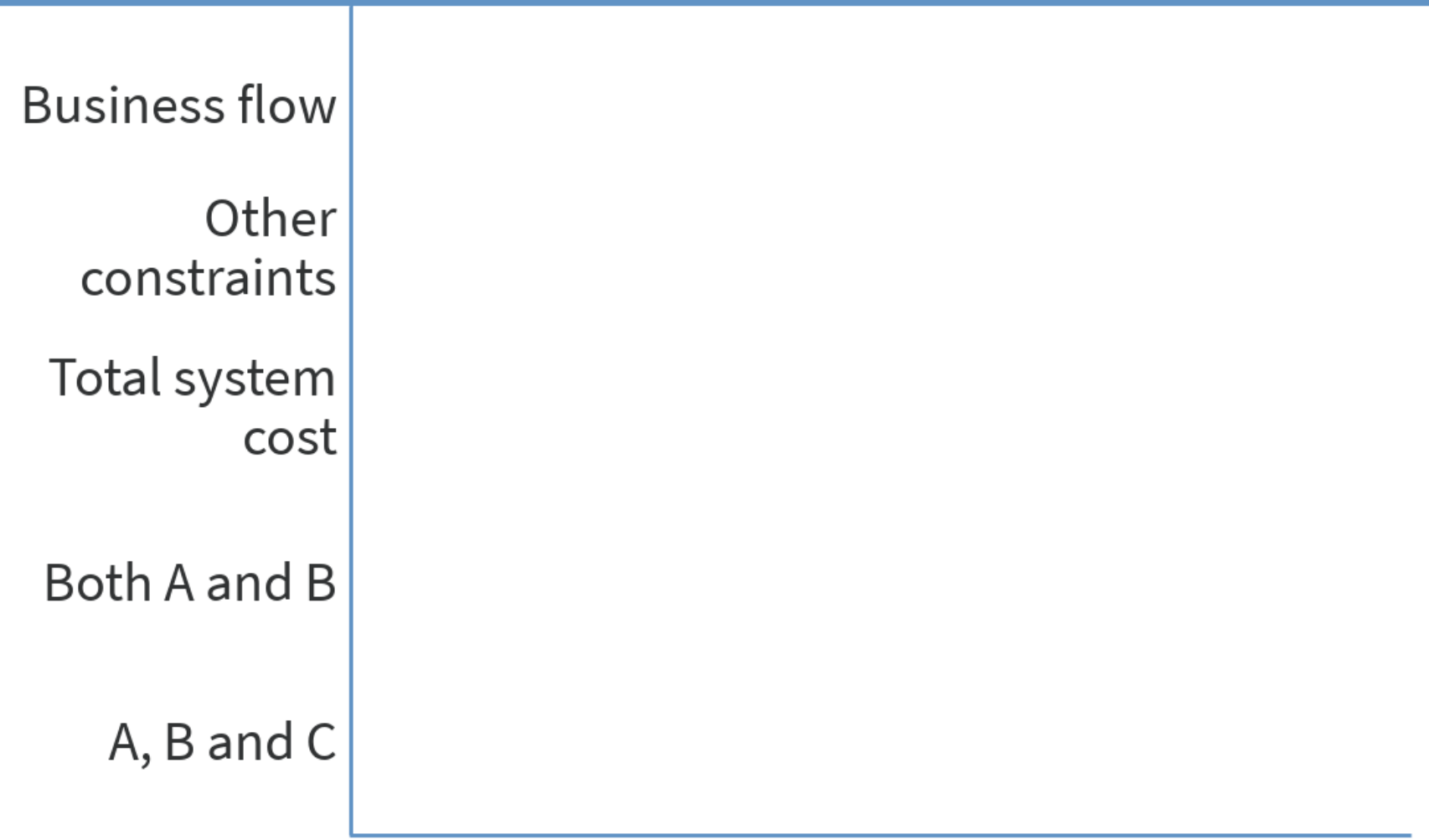
High-level requirements provide an overview and low-level requirements provide all the details

Only low-level requirements are important for system design and implementation

What is the optimal relationship of coupling and cohesion?



Categories of information that detailed requirements should address include



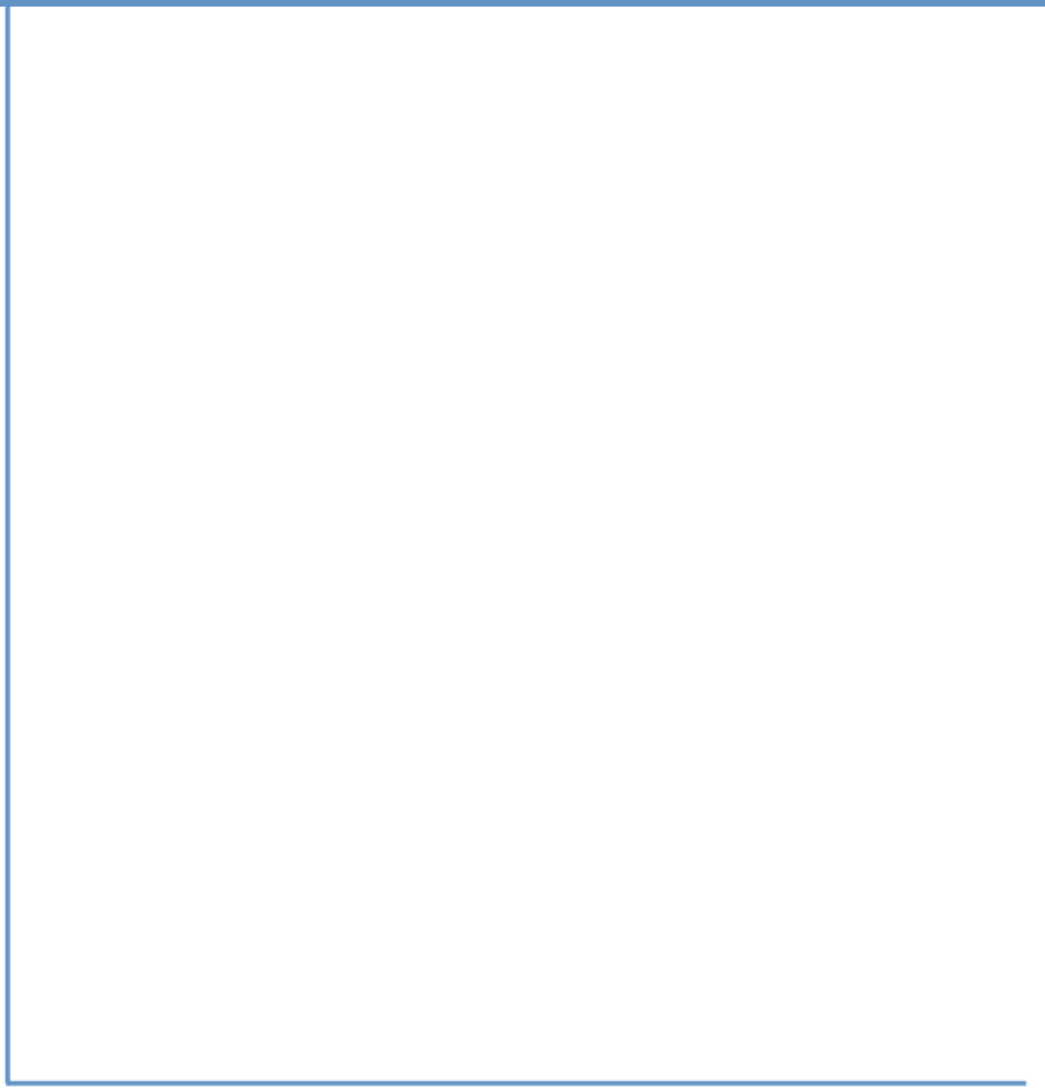
Prioritizing requirements

Is a precise activity

Is affected by the resources available to the development organization

Only utilizes immediate client needs and wants

Isn't necessary for "small" projects



A consistent specification