Requirement Specification

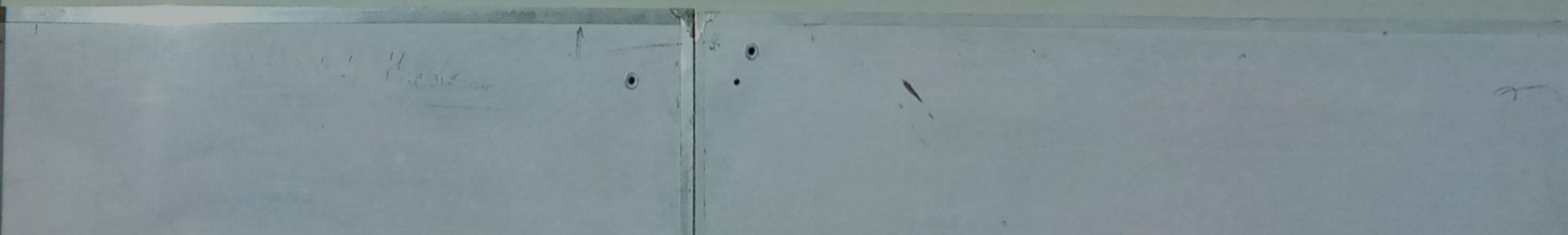
In writing a requirements document, two aspects have to be carefully balanced:

1. The need to make the requirements document readable
2. The need to make the set of requirements processable

- It is sometimes called a business requirements document (BRD), functional specification, product specification, system specification, or simply requirements document.

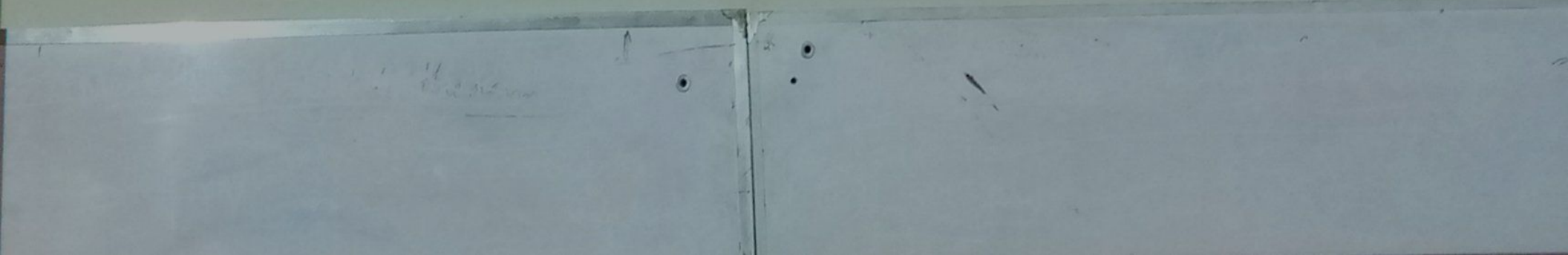
Attribute of Requirement

- The ambulance control system shall be able to handle up to 100 simultaneous emergency calls.
  - Source: R. Thomas
  - Priority: Mandatory
  - Release: 1
  - Review status: Accepted
  - Verifiable: Yes
  - Verification: By simulation, then by system test.



Language of Requirements

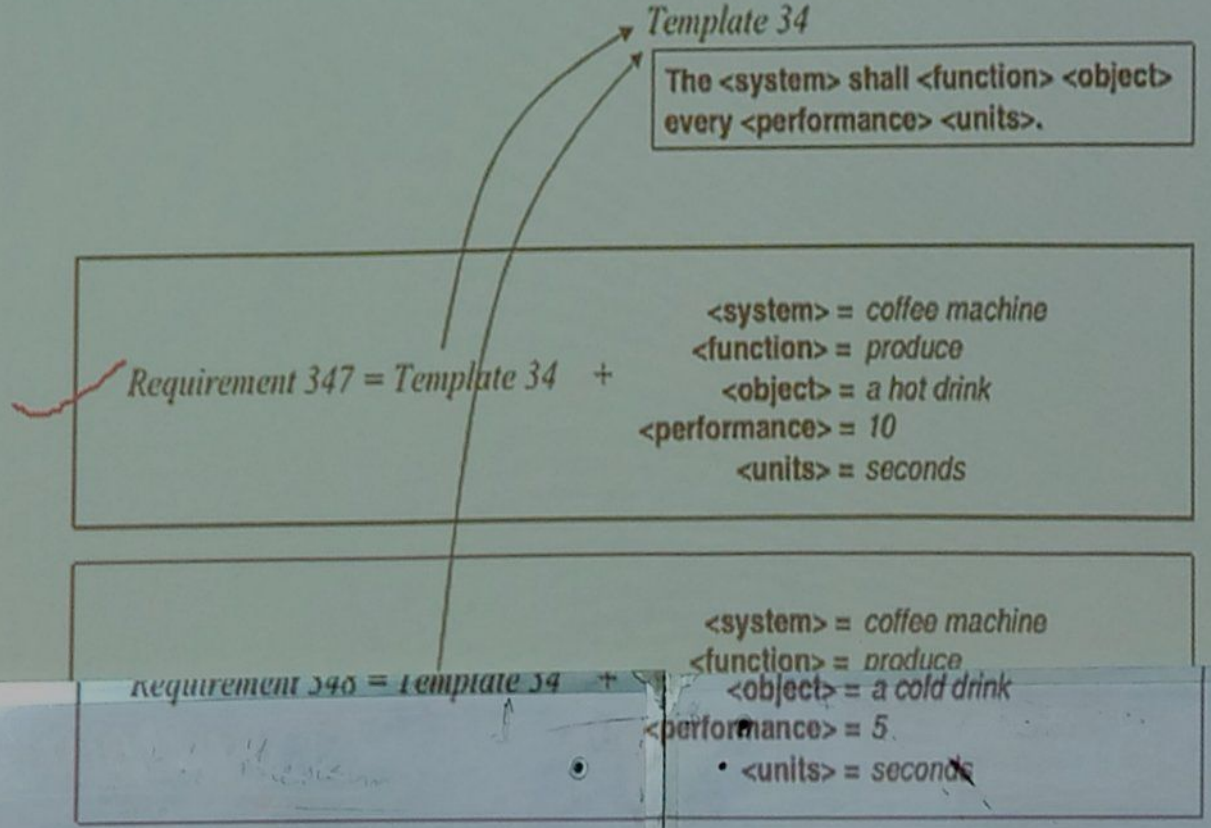
- The < stakeholder type > shall be able to < capability>



Requirement Boilerplate

- A requirement boilerplate is essentially a Natural Language (NL) pattern that restricts the syntax of requirements sentences to pre-defined linguistic structures.

Type of Constraint	Boiler-Plate
Performance/capability	The <system> shall be able to <function> <object> **not less than <performance> times per <units>.**
Performance/capability	The <system> shall be able to <function> <object> **of type <qualification> within <performance> <units>.**
Performance/capacity	The <system> shall be able to <function> **Not less than <quantity> <object>**
Performance/timeliness	The <system> shall be able to <function> <object> **within <performance> <units> from <event>.**
Performance/periodicity	The <system> shall be able to <function> **not less than** **<quantity> <object> within <performance> <units>.**
Interoperability/capacity	The <system> shall be able to <function> <object> **composed of not less than <performance> <units>** **with <external entity>**
Sustainability/periodicity	The <system> shall be able to <function> <object> **for <performance> <units> every <performance> <units>.**
Environmental/operability	The <system> shall be able to <function> <object> **while <operational condition>.**


Template 34
The <system> shall <function> <object> every <performance> <units>.
Requirement 347 = Template 34 +
<system> = coffee machine
<function> = produce
<object> = a hot drink
<performance> = 10
<units> = seconds
<system> = coffee machine
<function> = produce
<object> = a cold drink
<performance> = 5
<units> = seconds

Criteria for Writing Requirements

- Atomic: each statement carries a single traceable element.
- Unique: each statement can be uniquely identified.
- Feasible: technically possible within cost and schedule.
- Legal: legally possible.
- Clear: each statement is clearly understandable.
- Precise: each statement is precise and concise.
- Verifiable: each statement is verifiable, and it is known how.
- Abstract: does not impose a solution of design specific to the layer below.

Problems in Requirement

- The system shall perform at the maximum rating at all times except that in emergencies it shall be capable of providing up to 125% rating unless the emergency condition continues for more than 15 min in which case the rating shall be reduced to 105% but in the event that only 95% can be achieved then the system shall activate a reduced rating exception and shall maintain the rating within 10% of the stated values for a minimum of 30 min

- Avoid rambling: conciseness is a virtue; it doesn't have to read like a novel
- Avoid let-out clauses: such as "if that should be necessary"; they render the requirements useless.
- Avoid putting more than more requirement in a paragraph: often indicated by the presence of the word "and".
- Avoid speculation.
- Avoid vague words: usually, generally, often, normally, typically.
- Avoid vague terms: user friendly, versatile, flexible.
- Avoid wishful thinking: 100% reliable, please all users, safe, run on all platforms, never fail, handle all unexpected failures, upgradeable to all future situations.

1. Introduction
   1.1 Purpose
   1.2 Document conventions
   1.3 Project scope
   1.4 References
2. Overall description
   2.1 Product perspective
   2.2 User classes and characteristics
   2.3 Operating environment
   2.4 Design and implementation constraints
   2.5 Assumptions and dependencies
3. System features
   3.x System feature X
      3.x.1 Description
      3.x.2 Functional requirements
4. Data requirements
   4.1 Logical data model
   4.2 Data dictionary
   4.3 Reports
   4.4 Data acquisition, integrity, retention, and disposal
5. External interface requirements
   5.1 User interfaces
   5.2 Software interfaces
   5.3 Hardware interfaces
   5.4 Communications interfaces
6. Quality attributes
   6.1 Usability
   6.3 Security
   6.4 Safety
   6.x [others]
7. Internationalization and localization requirements
8. Other requirements

Appendix A: Glossary

Appendix B: Analysis models