# Important Points

* There are 5 major phases of the software requirements process [113]. The 5 phases are elicitation, analysis, specification, management, and validation [113]. These phases are not completely distinct [113].
* A requirement is defined as a property of a system [114]. Within the scope of software requirements there are three kinds of requirements: functional requirements that specify what a system must do [114], nonfunctional requirements that specify a quality of the system [114, 115], and process requirements that constrain the development process [115].
* Elicitation is the process of eliciting requirements from stakeholders. [115]. To elicit requirements well you need to understand the problem the software is meant to solve [115], understand the domain and subdomains of the problem [115, 116], and “make the business case” [116]. Cleland-Huang notes that “it is imperative to find the right set of stakeholders.” [115] Failing to include each group of stakeholders in elicitation may cause excluded stakeholders to withhold support from the project [115].
* There are many elicitation techniques [117]. Cleland-Huang lists collaborative sessions, interviewing, questionnaires, and ethnography as common elicitation techniques [117]. She also introduces prototyping, documentation, modeling, roleplaying, and reviewing checklists of nonfunctional requirements as common elicitation techniques [117].
* Elicitors are responsible for recognizing conflicts in the requirements due to differing visions of the system’s functionality [118]; left unresolved, these kinds of conflicts can result in the cancellation of the project [118].
* During requirement analysis the requirements accumulated thus far are analyzed via concept modeling and requirements classification [118]. High-level architectural design is considered during the requirement analysis phase [118].
* The requirements specification phase results in the creation of three documents: the systems definition document [119], the systems requirements document [119], and the software requirements document [119].
* Requirements in the software requirements document must be concise, correct, unambiguous, feasible, and verifiable [120]. The author states that “all requirements describing a mandatory system property use the words ‘shall’ or ‘must’” [120].
* The set of all requirements together should be realistic, concise, complete, and consistent [120].
* The validation and verification phase is when the requirements are checked for correctness [120], and when we make sure we have the correct requirements [121]. Stakeholder reviews, prototyping, model validation, and acceptance tests may be used in this phase.
* Requirements management is the management of changes to the product during its life [121]. Techniques such as requirements traceability, change requests, and requirements attributes are used to manage change [121, 122].
* Despite the recent trends towards agile development, requirements engineering is useful when developing large and complicated systems regardless of the development philosophy in play [122].

# Disagreements

“Roleplaying or use of surrogate techniques can be used to explore stakeholders [sic] needs when those stakeholders are unavailable.” [117]

I disagree with this method of eliciting requirements. While it may be useful for brainstorming ideas, if a requirement doesn’t come from a stakeholder you risk becoming disconnected from your stakeholders. The risk could lead to a situation like the one we discussed in class regarding Research In Motion. Requirements need to come from stakeholders.

# Questions and Clarifications

I don’t understand requirements traceability [121].

In requirements analysis the author suggests modeling using UML [118]. I know we briefly touched on using UML in software requirements during class yesterday, but I don’t understand how we could use UML without crossing into the realm of design.

I don’t understand figure 1 [114].