Designing Principles

The set of principles which has been established to aid the software engineer in navigating the design process are:

1. **The design process should not suffer from tunnel vision** – A good designer should consider alternative approaches. Judging each based on the requirements of the problem, the resources available to do the job and any other constraints.
2. **The design should be traceable to the analysis model –** because a single element of the design model often traces to multiple requirements, it is necessary to have a means of tracking how the requirements have been satisfied by the model.
3. **The design should not reinvent the wheel** – Systems are constructed using a set of design patterns, many of which may have likely been encountered before. These patterns should always be chosen as an alternative to reinvention. Time is short and resources are limited! Design time should be invested in representing truly new ideas and integrating those patterns that already exist.
4. **The design should minimise intellectual distance between the software and the problem as it exists in the real world** – That is, the structure of the software design should (whenever possible) mimic the structure of the problem domain.
5. **The design should exhibit uniformity and integration** – a design is uniform if it appears that one person developed the whole thing. Rules of style and format should be defined for a design team before design work begins. A design is integrated if care is taken in defining interfaces between design components.
6. **The design should be structured to degrade gently, even with bad data, events, or operating conditions are encountered** – Well-designed software should never "bomb". It should be designed to accommodate unusual circumstances, and if it must terminate processing, do so in a graceful manner.
7. **The design should be reviewed to minimize conceptual (semantic) errors** – there is sometimes the tendency to focus on minute details when the design is reviewed, missing the forest for the trees. The designer team should ensure that major conceptual elements of the design have been addressed before worrying about the syntax if the design model.
8. **Design is not coding, coding is not design** – Even when detailed designs are created for program components, the level of abstraction of the design model is higher than source code. The only design decisions made of the coding level address the small implementation details that enable the procedural design to be coded.
9. **The design should be structured to accommodate change**.
10. **The design should be assessed for quality as it is being created**.

When these design principles are properly applied, the design exhibits both external and internal quality factors. External quality factors are those factors that can readily be observed by the user, (e.g. speed, reliability, correctness, usability). Internal quality factors relate to the technical quality (which is important to the software engineer) more so the quality of the design itself. To achieve internal quality factors the designer must understand basic design concepts.