Iteration

A process of repeating a set of operations until a specific result is achieved.

Ordered complexity does not occur without iteration. In nature, iteration allows complex structures to form by progressively building on simpler structures. In design, iteration allows complex structures to be created by progressively exploring, testing, and tuning the design. The emergence of ordered complexity results from an accumulation of knowledge and experience that is then applied to the design. For example, a quality software user interface is developed through a series of design iterations. Each version is reviewed and tested, and the design is then iterated based on the feedback. The interface typically progresses from low fidelity to high fidelity as more is learned about the interface and how it will be used. Iteration occurs in all development cycles in two basic forms: design iteration and development iteration.[1]

Design iteration is the expected iteration that occurs when exploring, testing, and refining design concepts. Each cycle in the design process narrows the wide range of possibilities until the design conforms to the design requirements. Prototypes of increasing fidelity are used throughout the process to test concepts and identify unknown variables. Members of the target audience should be actively involved in various stages of iterations to support testing and verify design requirements. Whether tests are deemed a success or failure is irrelevant in design iteration, since both success and failure provide important information about what does and does not work. In fact, there is often more value in failure, as valuable lessons are learned about the failure points of a design. The outcome of design iteration is a detailed and well-tested specification that can be developed into a final product.[2]

Development iteration is the unexpected iteration that occurs when building a product. Unlike design iteration, development iteration is rework—i.e., unnecessary waste in the development cycle. Development iteration is costly and undesirable, and generally the result of either inadequate or incorrect design specifications, or poor planning and management in the development process. The unknowns associated with a design should ideally be eliminated during the design stage.

Plan for and employ *design* iteration. Establish clear criteria defining the degree to which design requirements must be satisfied for the design to be considered complete. One of the most effective methods of reducing development iteration is to ensure that all development members have a clear, high-level vision of the final product. This is often accomplished through well-written specifications accompanied by high-fidelity models and prototypes.

See also Development Cycle, Fibonacci Sequence, Most Advanced Yet Acceptable, Prototyping, and Self-Similarity.

[1] A seminal contemporary work on iteration in design is *The Evolution of Useful Things* by Henry Petroski, Vintage Books, 1994. See also *Product Design and Development* by Karl T. Ulrich and Steven D. Eppinger, McGraw-Hill Higher Education, 2nd ed., 1999. See also "Positive vs. Negative Iteration in Design" by Glenn Ballard, *Proceedings of the Eighth Annual Conference of the International Group for Lean Construction*, 2000.

[2] A common problem with design iteration is the absence of a defined endpoint—i.e., each iteration refines the design, but also reveals additional opportunities for refinement, resulting in a design process that never ends. To avoid this, establish clear criteria defining the degree to which design requirements must be satisfied for the design to be considered complete.

Quality design does not occur without iteration. For example, the design of the SnoWalkers snowshoes is the result of numerous design iterations over a two-year period. The design process made ample use of prototypes, which allowed designers to improve their understanding of design requirements and product performance, and continually refine the design with each iteration.