Lecture 6
Software Engineering

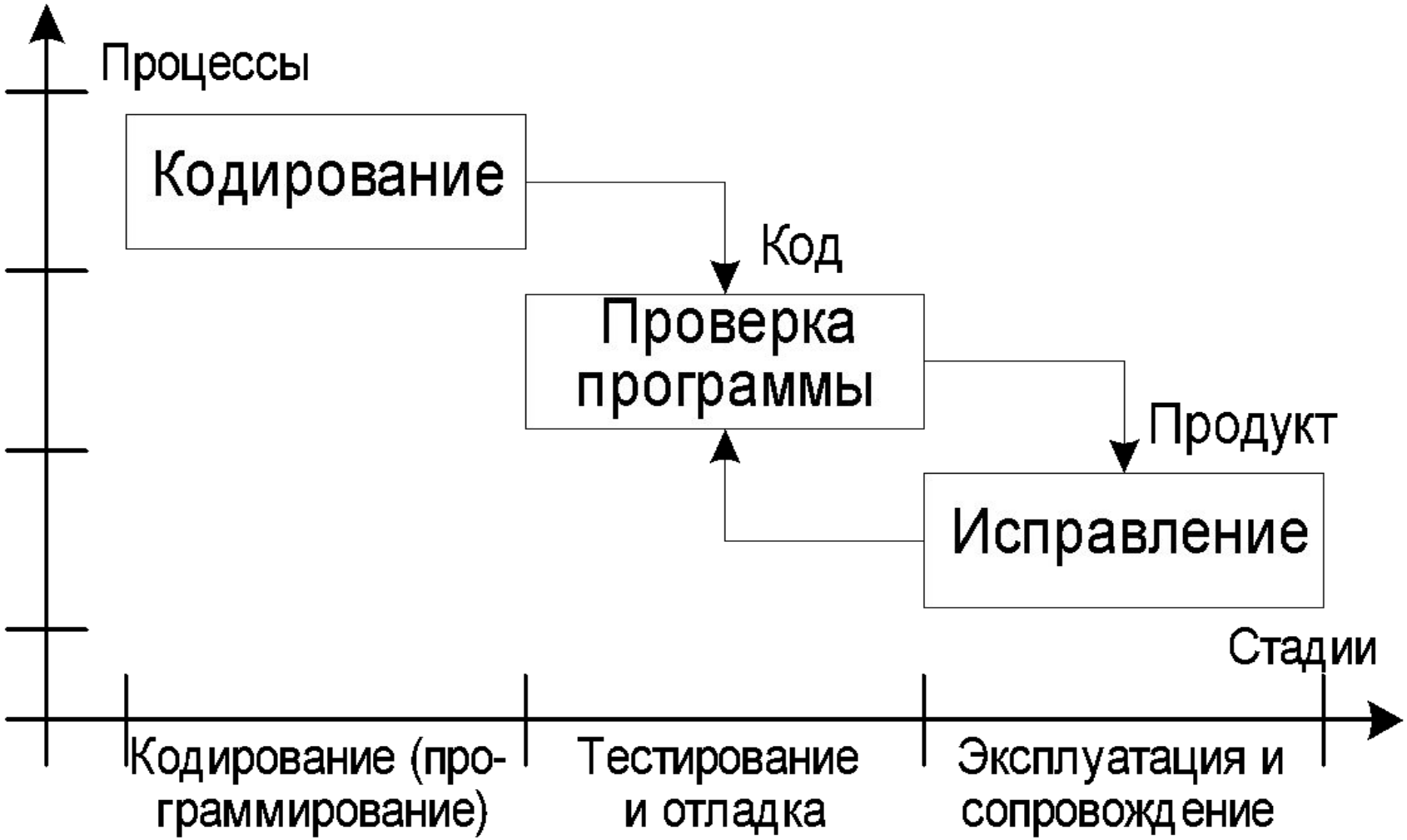
Model of the software life cycle

- Major software life cycle models are:
  The cascade model,
  Iterative incremental model
  The evolutionary model
  Spiral Model.

  Unplanned model or "coding - fix" is the most simple model of software life cycle.

  The principle of the model (Figure 6.1) is to write the code of the program without any serious pre-requirements analysis and design, launch programs to test its performance and the subsequent correction of errors and / or adding functionality to obtain version of the program that satisfies the user.

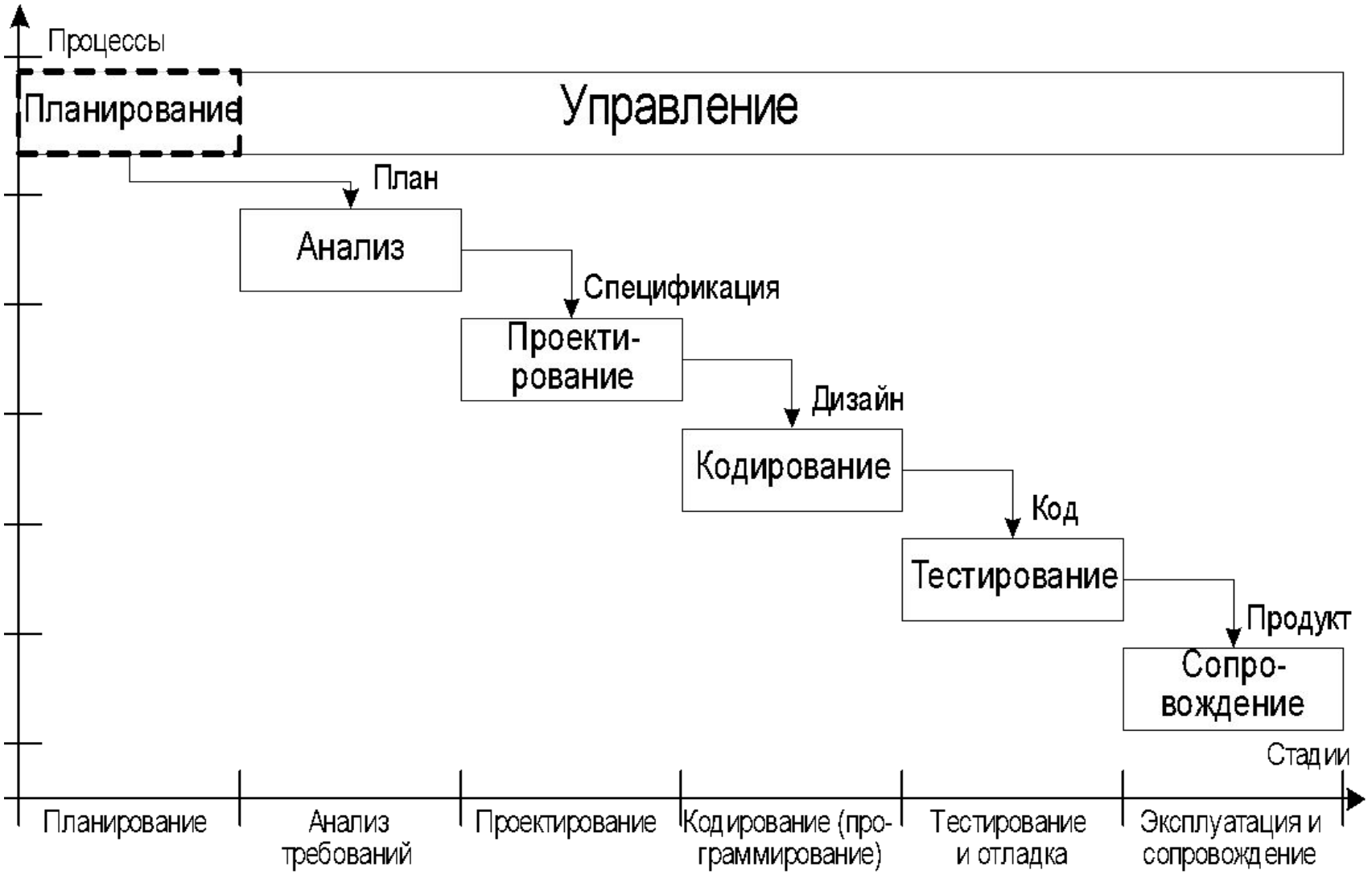
Unplanned model or "coding - fix"



The classic waterfall model or waterfall model

- The classic waterfall model or waterfall model created by analogy with the methods of the other engineering fields, where the practice of product creation phase of preparing specifications prior to delivery to the customer.
  The principle of the model (Figure 6.2) is a single execution of processes in the form of pre-ordered and clearly limited in time steps, as it were implemented in their natural boundaries.
  All the processes, activities, tasks are performed consistently well, is not allowed any overlap, duplication and return it. This hard limit does such a model is not practicable, as the already completed steps are not allowed any errors that require the utmost quality artifacts.

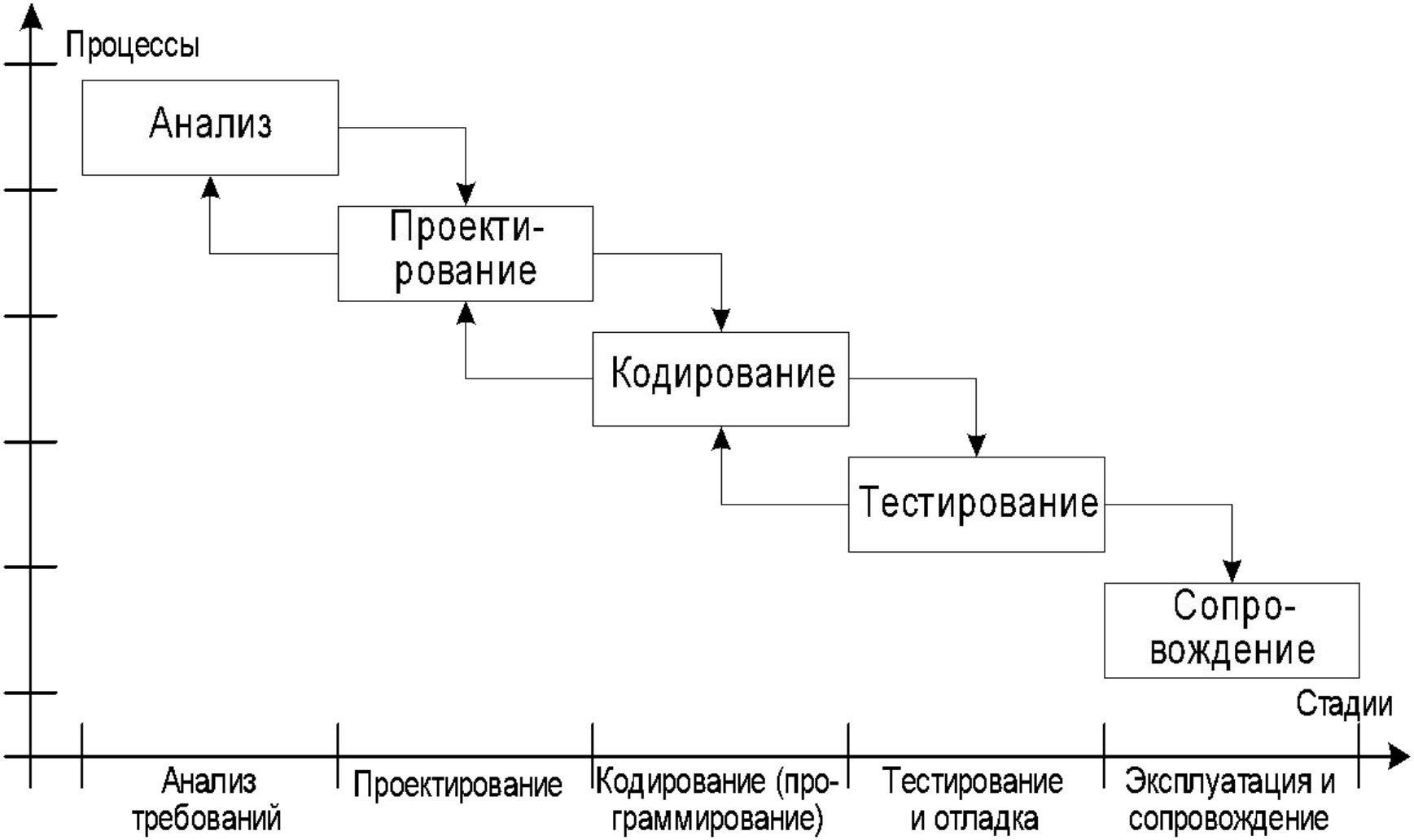
The classic waterfall model or waterfall model



Modified cascade model or whirlpool

- In practice, the cascading model without these constraints. Modified cascade model or whirlpool is a simple case of such models. The principle of the model (Figure 6.3) is the ability to return to a previous stage in the case of finding errors in this stage and the revision or clarification of previous decisions.

Modified cascade model or whirlpool



Prototipiruemaya model or prototype

- Prototipiruemaya model or prototype is designed to solve the problems in the design under uncertainty of initial requirements through the development of prototypes of the desired product. Model requires a set of quick construction of prototypes, and therefore the implementation of this model is only possible with appropriate instrumentation automation.
  The classical model uses prototyping prototyping to gradually identify the requirements. The principle of the model (Figure 6.4) is in the early development of a highly simplified prototype, which in accordance with the wishes of the user cycle improves and becomes more complicated as long as the requirements for the software does not become apparent. After that, the formalization of the identified requirements (itemize) and maintains its own product development for any model. Prototyping and product treated as a separate iteration of software life cycle.