SOFTWARE PROTOTYPING

Software Prototyping

- Rapid software development to validate requirements

System prototyping

- Prototyping is the rapid development of a system
- In the past, the developed system was normally thought of as inferior in some way to the required system so further development was required
- Now, the boundary between prototyping and normal system development is blurred and many systems are developed using an evolutionary approach

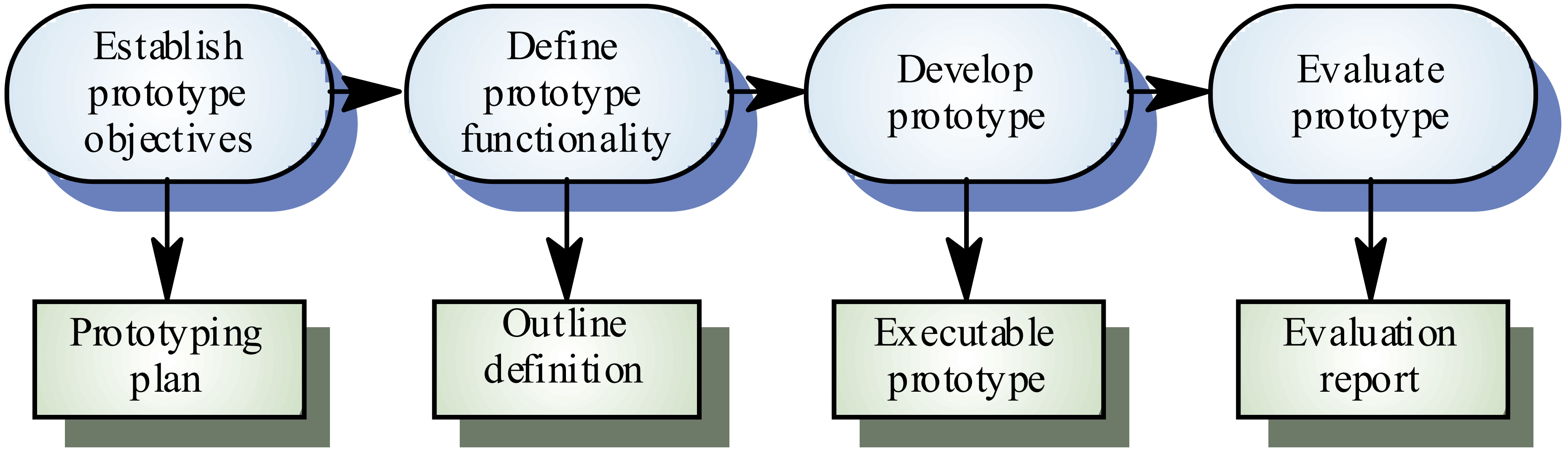
Uses of system prototypes

- The principal use is to help customers and developers understand the requirements for the system
  - Requirements elicitation. Users can experiment with a prototype to see how the system supports their work
  - Requirements validation. The prototype can reveal errors and omissions in the requirements
- Prototyping can be considered as a risk reduction activity which reduces requirements risks

Prototyping benefits

- Misunderstandings between software users and developers are exposed
- Missing services may be detected and confusing services may be identified
- A working system is available early in the process
- The prototype may serve as a basis for deriving a system specification
- The system can support user training and system testing

Prototyping process



Prototyping benefits

- Improved system usability
- Closer match to the system needed
- Improved design quality
- Improved maintainability
- Reduced overall development effort

Prototyping in the software process

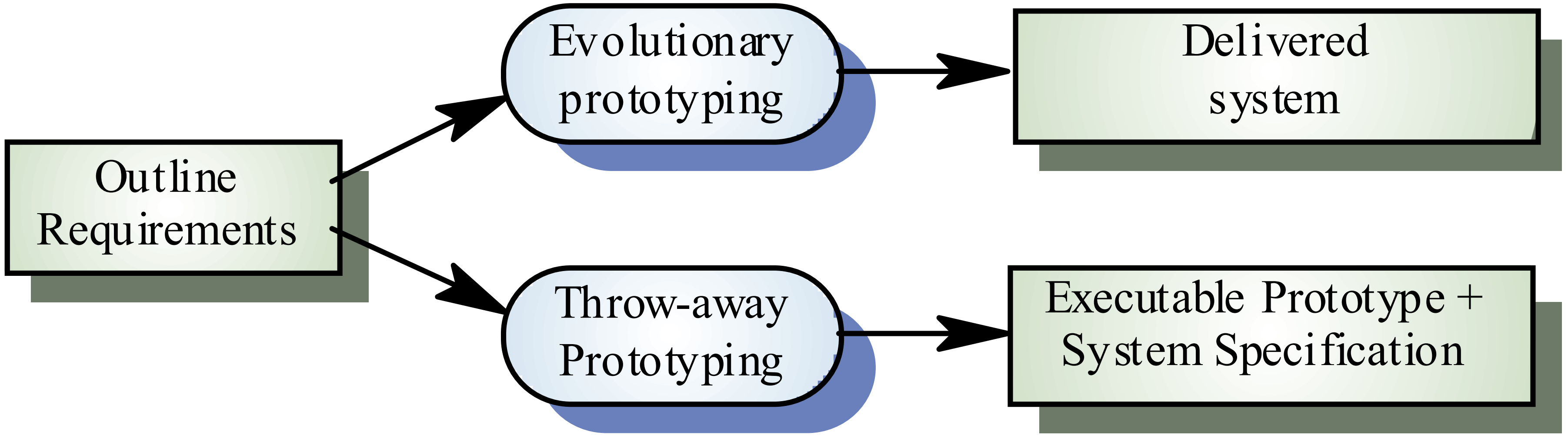
- Evolutionary prototyping
  - An approach to system development where an initial prototype is produced and refined through a number of stages to the final system
- Throw-away prototyping
  - A prototype which is usually a practical implementation of the system is produced to help discover requirements problems and then discarded.
  - The system is then developed using some other development process

Prototyping objectives

- The objective of *evolutionary prototyping* is to deliver a working system to end-users. The development starts with those requirements which are best understood.

- The objective of *throw-away prototyping* is to validate or derive the system requirements. The prototyping process starts with those requirements which are poorly understood

Approaches to prototyping



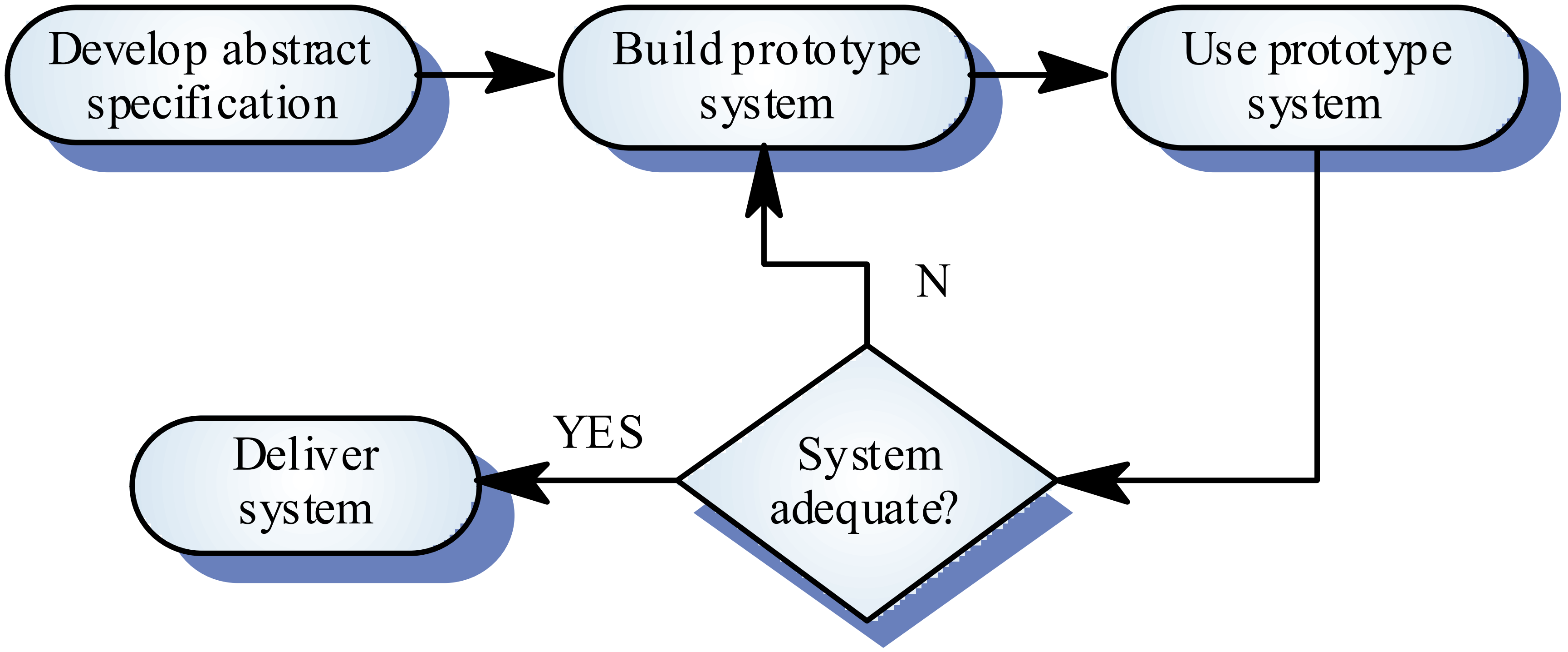
Evolutionary prototyping

- Must be used for systems where the specification cannot be developed in advance e.g. AI systems and user interface systems

- Based on techniques which allow rapid system iterations

- Verification is impossible as there is no specification. Validation means demonstrating the adequacy of the system

Evolutionary prototyping



Evolutionary prototyping advantages

- Accelerated delivery of the system
  - Rapid delivery and deployment are sometimes more important than functionality or long-term software maintainability
- User engagement with the system
  - Not only is the system more likely to meet user requirements, they are more likely to commit to the use of the system

Evolutionary prototyping problems

- Management problems
  - Existing management processes assume a waterfall model of development
  - Specialist skills are required which may not be available in all development teams
- Maintenance problems
  - Continual change tends to corrupt system structure so long-term maintenance is expensive
- Contractual problems

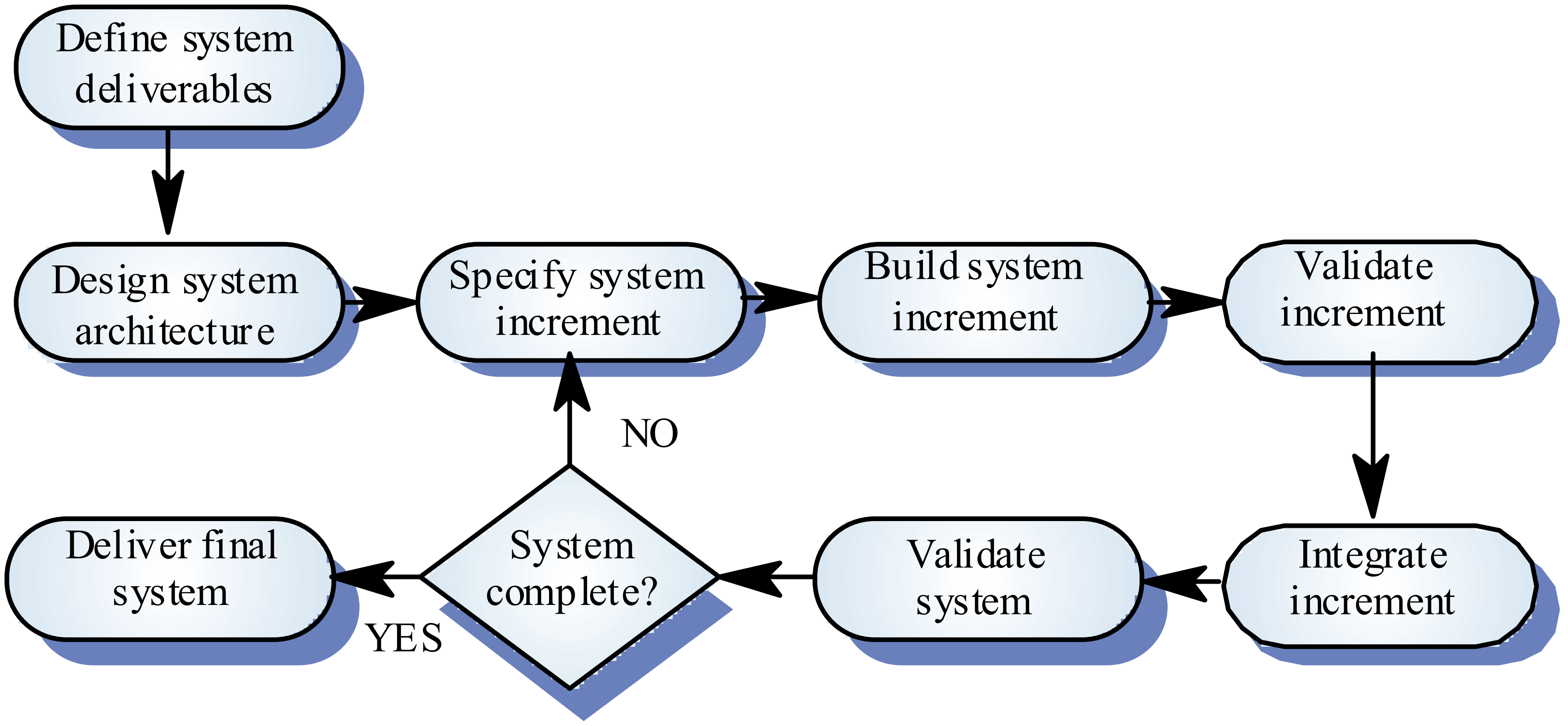
Prototypes as specifications

- Some parts of the requirements (e.g. safety-critical functions) may be impossible to prototype and so don’t appear in the specification
- An implementation has no legal standing as a contract
- Non-functional requirements cannot be adequately tested in a system prototype

Incremental development

- System is developed and delivered in increments after establishing an overall architecture
- Requirements and specifications for each increment may be developed
- Users may experiment with delivered increments while others are being developed. therefore, these serve as a form of prototype system
- Intended to combine some of the advantages of prototyping but with a more manageable process and better system structure

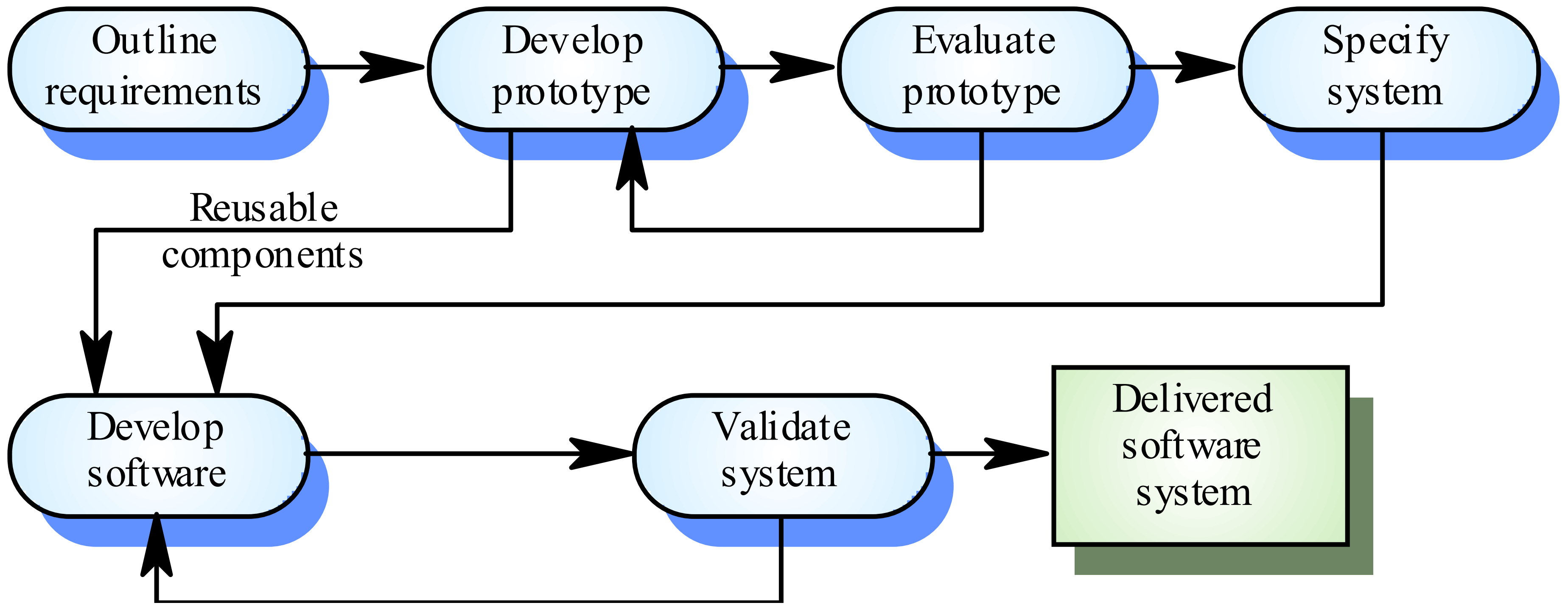
Incremental development process



Throw-away prototyping

- Used to reduce requirements risk
- The prototype is developed from an initial specification, delivered for experiment then discarded
- The throw-away prototype should NOT be considered as a final system
  - Some system characteristics may have been left out
  - There is no specification for long-term maintenance
  - The system will be poorly structured and difficult to maintain

Throw-away prototyping



Prototype delivery

- Developers may be pressurised to deliver a throw-away prototype as a final system
- This is not recommended
  - It may be impossible to tune the prototype to meet non-functional requirements
  - The prototype is inevitably undocumented
  - The system structure will be degraded through changes made during development
  - Normal organisational quality standards may not have been applied

Rapid prototyping techniques

- Various techniques may be used for rapid development
  - Dynamic high-level language development
  - Database programming
  - Component and application assembly
- These are not exclusive techniques - they are often used together
- Visual programming is an inherent part of most prototype development systems

Dynamic high-level languages

- Languages which include powerful data management facilities
- Need a large run-time support system. Not normally used for large system development
- Some languages offer excellent UI development facilities
- Some languages have an integrated support environment whose facilities may be used in the prototype

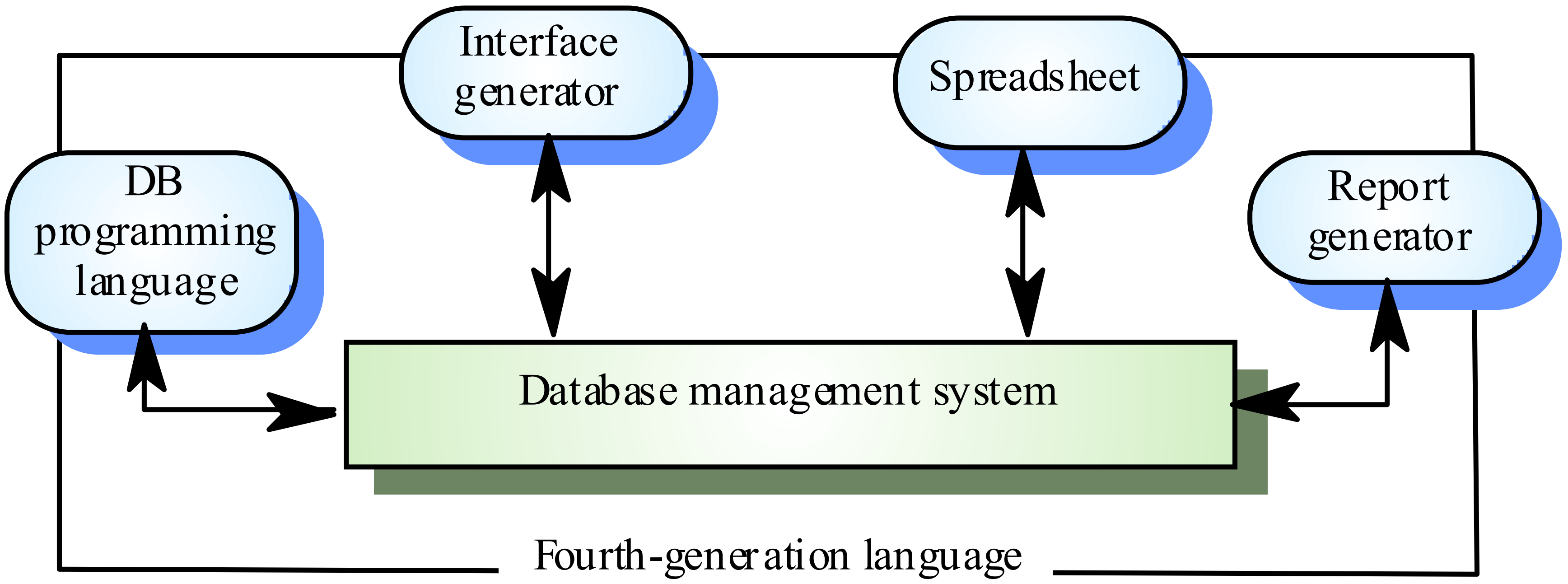
Choice of prototyping language

- What is the application domain of the problem?
- What user interaction is required?
- What support environment comes with the language?
- Different parts of the system may be programmed in different languages. However, there may be problems with language communications

Database programming languages

- Domain specific languages for business systems based around a database management system
- Normally include a database query language, a screen generator, a report generator and a spreadsheet.
- May be integrated with a CASE toolset
- The language + environment is sometimes known as a fourth-generation language (4GL)
- Cost-effective for small to medium sized business systems

Database programming



Prototyping with reuse

- Application level development
  - Entire application systems are integrated with the prototype so that their functionality can be shared
  - For example, if text preparation is required, a standard word processor can be used
- Component level development
  - Individual components are integrated within a standard framework to implement the system
  - Frame work can be a scripting language or an integration framework such as CORBA