Il processo software

- **Processo software**
  - serie di attività necessarie alla realizzazione del prodotto software nei tempi, con i costi e con le desiderate caratteristiche di qualità.
- Nel suo contesto:
  - si applicano metodi, tecniche e strumenti
  - si creano prodotti (sia intermedi che finali)
  - si stabilisce il controllo gestionale del progetto
  - si garantisce la qualità
  - si governano le modifiche

Fasi del processo

- Come visto, il processo software segue un **ciclo di vita** che si articola in 3 stadi (sviluppo, manutenzione, dismissione). Nel primo stadio si possono riconoscere due tipi di fasi:
  - fasi di tipo **definizione**
  - fasi di tipo **produzione**
- Le fasi di **definizione** si occupano di "*cosa*" il software deve fornire. Si definiscono i requisiti, si producono le specifiche
- Le fasi di **produzione** definiscono "*come*" realizzare quanto ottenuto con le fasi di definizione. Si progetta il software, si codifica, si integra e si rilascia al cliente
- Lo stadio di *manutenzione* è a supporto del software realizzato e prevede fasi di definizione e/o produzione al suo interno
- Durante ogni fase si procede ad effettuare il **testing** di quanto prodotto, mediante opportune tecniche di *verifica e validazione* (*V&V*) applicate sia ai prodotti intermedi che al prodotto finale

Tipi di manutenzione

- Manutenzione **correttiva**, che ha lo scopo di eliminare i difetti (*fault*) che producono guasti (*failure*) del software
- Manutenzione **adattativa**, che ha lo scopo di adattare il software ad eventuali cambiamenti a cui è sottoposto l'ambiente operativo per cui è stato sviluppato
- Manutenzione **perfettiva**, che ha lo scopo di estendere il software per accomodare funzionalità aggiuntive
- Manutenzione **preventiva** (o *software reengineering*), che consiste nell'effettuare modifiche che rendano più semplici le correzioni, gli adattamenti e le migliorie

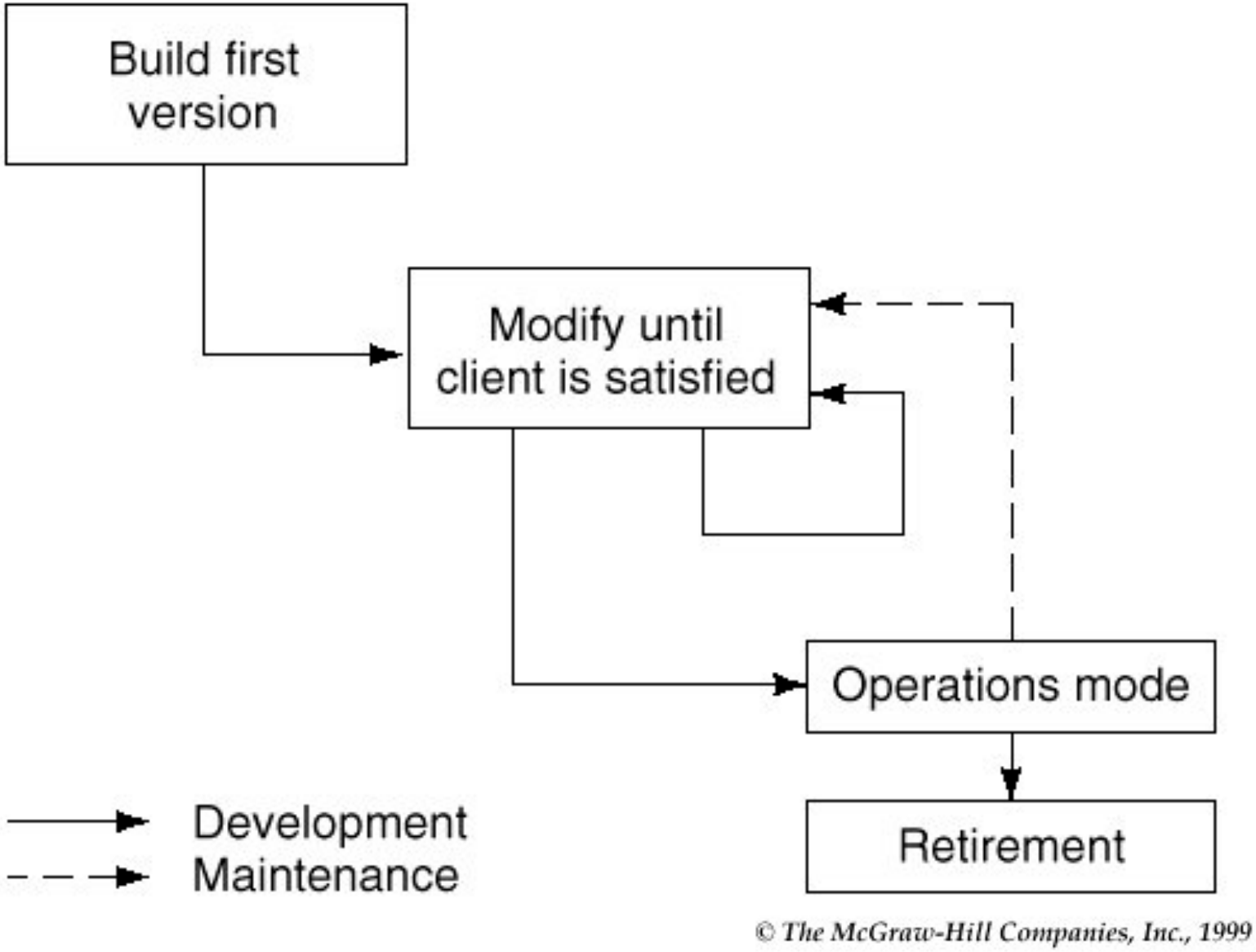
Definizione di ciclo di vita

- Def. IEEE Std 610-12 (*Software Eng. Terminology*)
  - intervallo di tempo che intercorre tra l'istante in cui nasce l'esigenza di costruire un prodotto software e l'istante in cui il prodotto viene dismesso
  - include le fasi di definizione dei requisiti, specifica, pianificazione, progetto preliminare, progetto dettagliato, codifica, integrazione, testing, uso, manutenzione e dismissione
  - *Nota*: tali fasi possono sovrapporsi o essere eseguite in modo iterativo

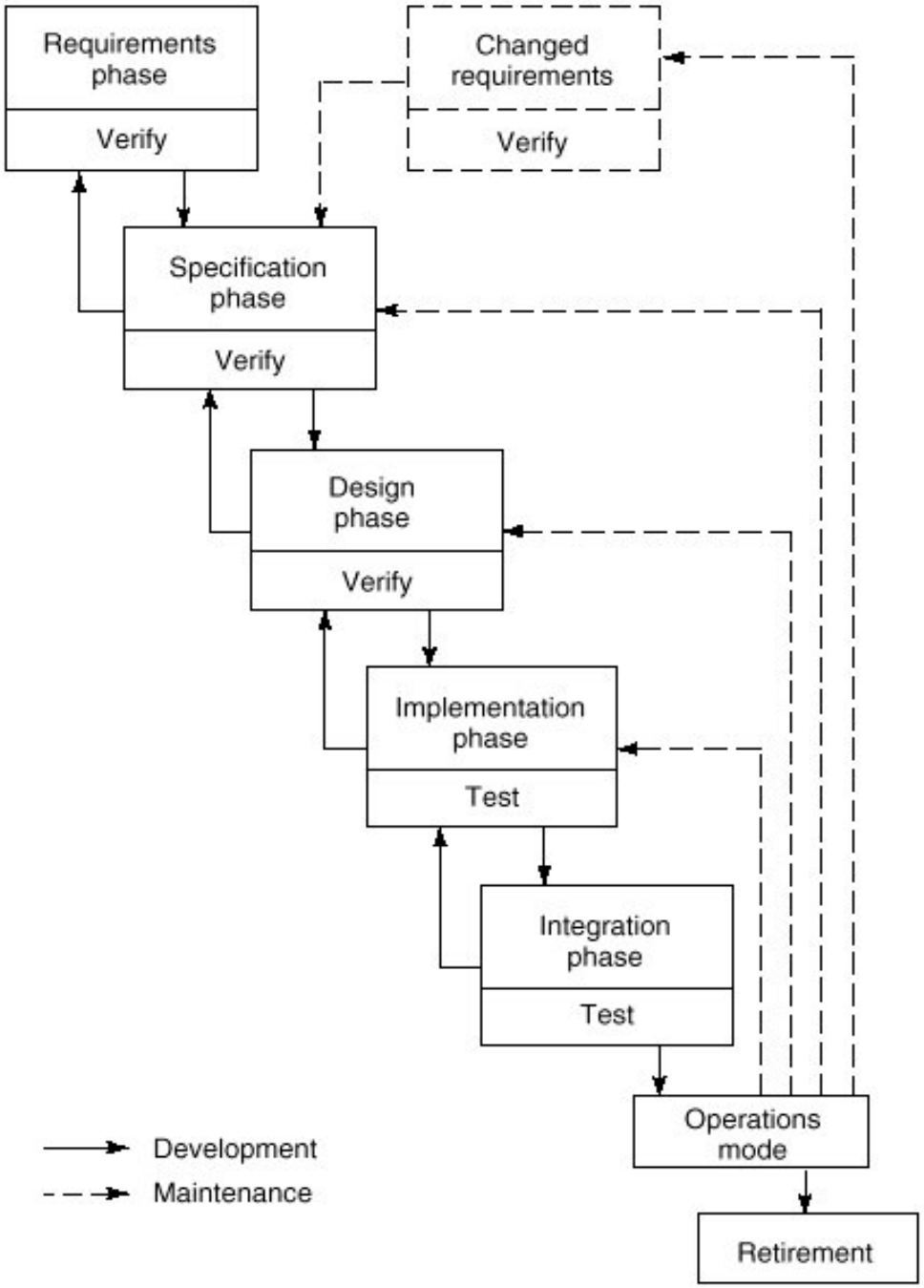
Modelli di ciclo di vita

- Il **modello del ciclo di vita** del software specifica la serie di fasi attraverso cui il prodotto software progredisce e l'ordine con cui vanno eseguite, dalla definizione dei requisiti alla dismissione
- La scelta del modello dipende dalla natura dell'applicazione, dalla maturità dell'organizzazione, da metodi e tecnologie usate e da eventuali vincoli dettati dal cliente
- L'assenza di un modello del ciclo di vita corrisponde ad una modalità di sviluppo detta "*build & fix*" (o "*fix-it-later*"), in cui il prodotto software viene sviluppato e successivamente rilavorato fino a soddisfare le necessità del cliente

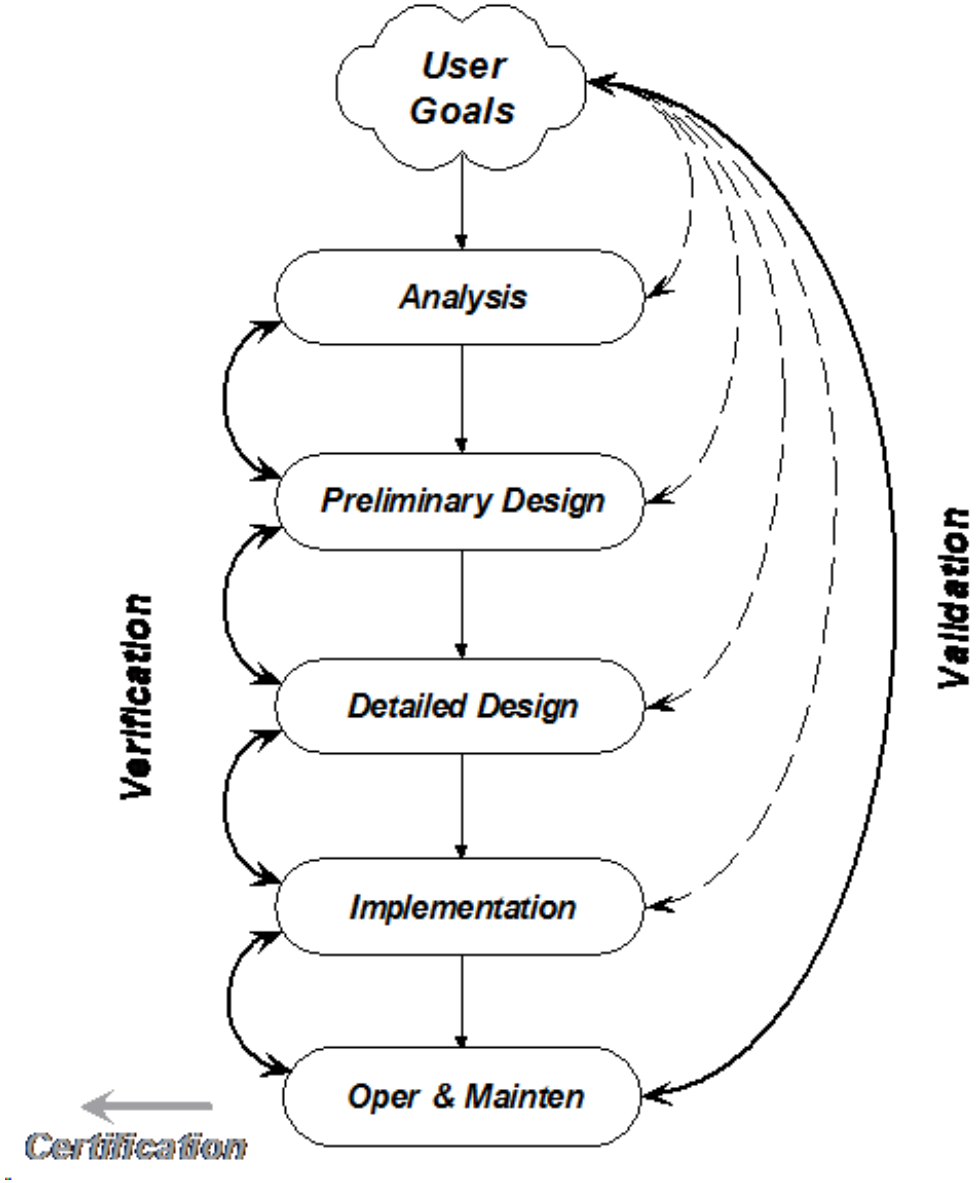
Build&Fix



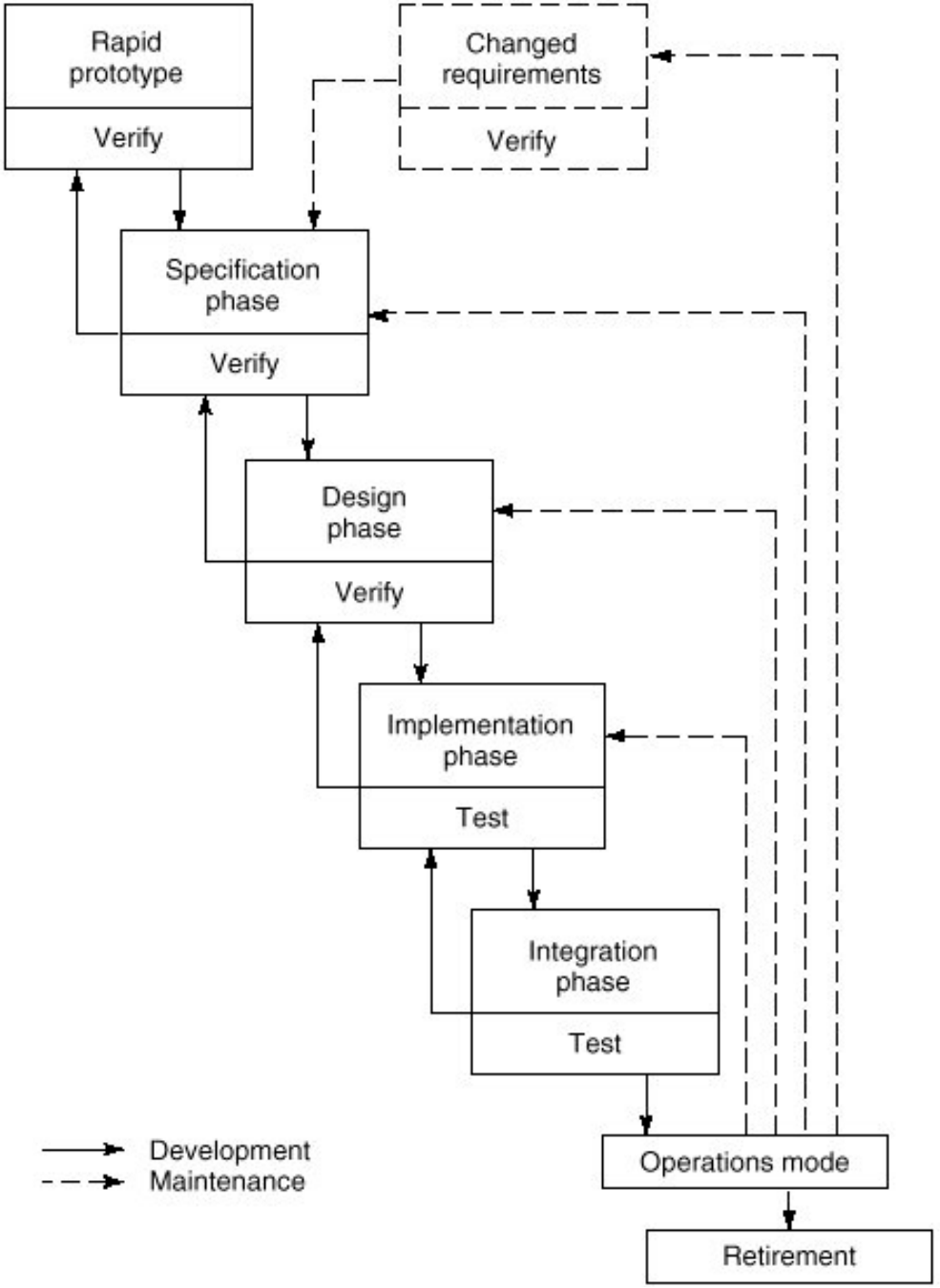
Modello Waterfall



Verification & Validation (V&V) nel Waterfall



Rapid Prototyping Model



Software Prototyping

Rapid software development to

elicit or *validate* requirements

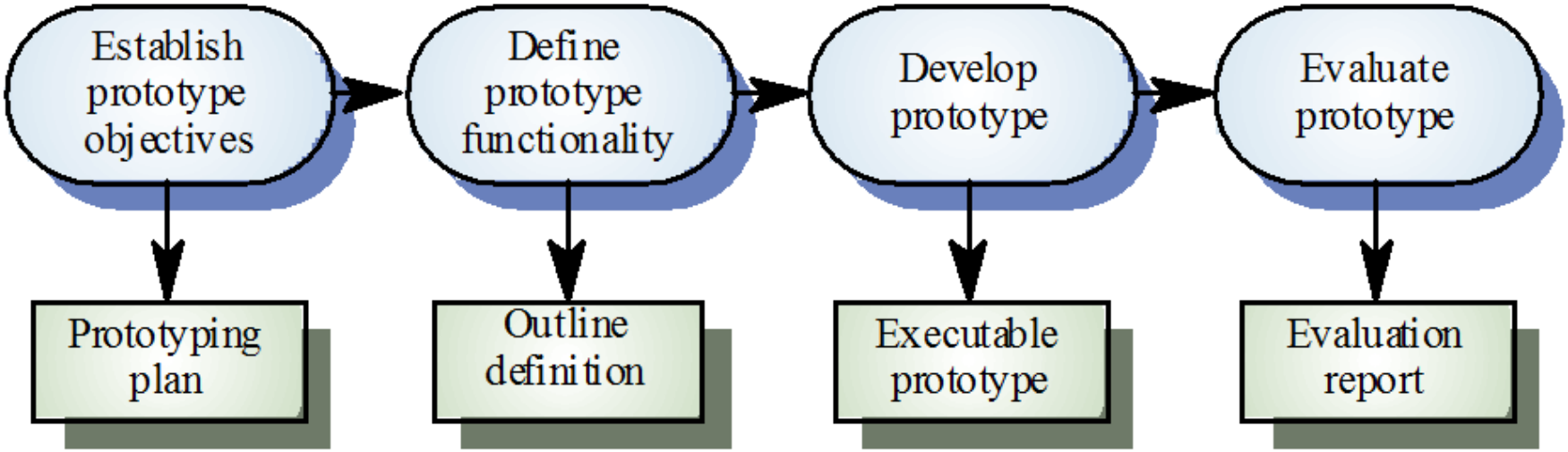
Uses of system prototypes

- The principal use is to help customers and developers understand the software requirements
  - 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 software specification
- The prototype can support user training and product testing

Prototyping process



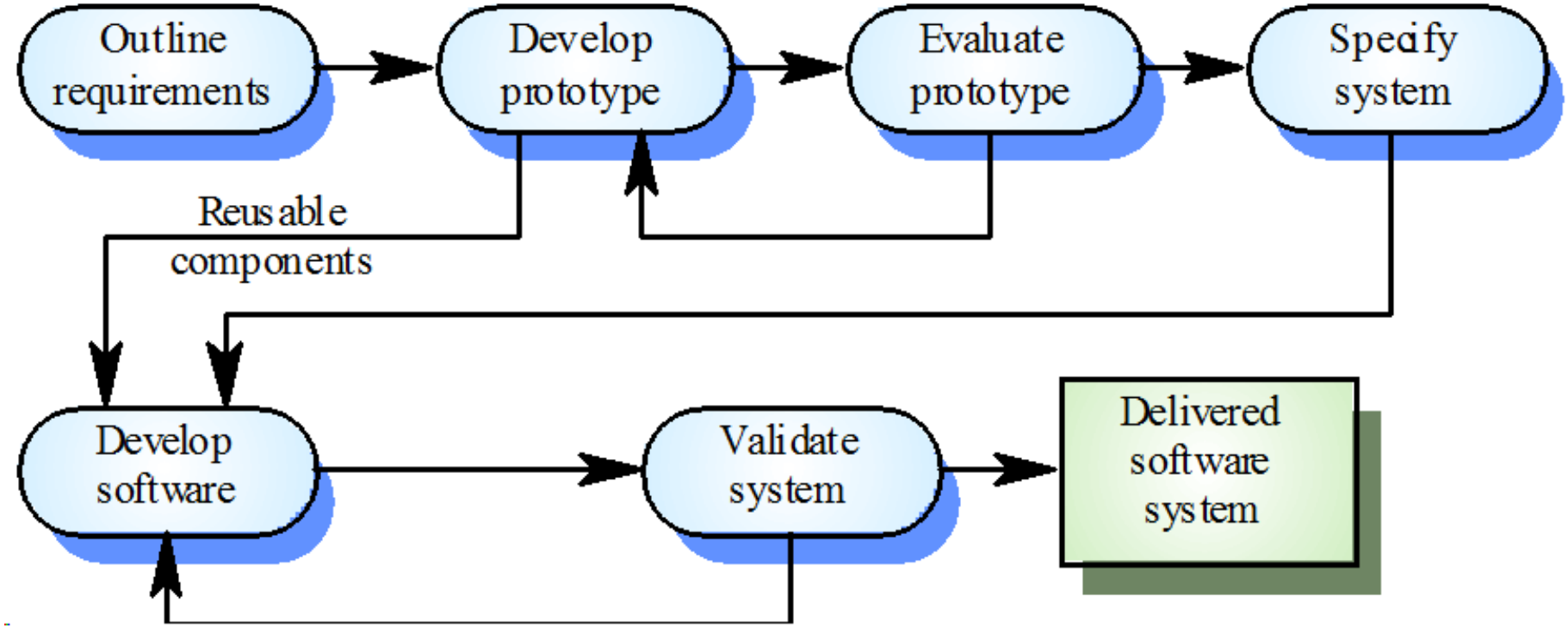
Prototypes as specifications

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

Throw-away prototyping

- A prototype which is usually a practical implementation of the product is produced to help discover requirements problems and then discarded. The product is then developed using some other development process
- Used to reduce requirements risk
- The prototype is developed from an initial requirement, delivered for experiment then discarded
- The throw-away prototype should NOT be considered as a final product
  - Some characteristics may have been left out
  - There is no specification for long-term maintenance
  - The product will be poorly structured and difficult to maintain

Throw-away prototyping process



Throw-away prototype delivery

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

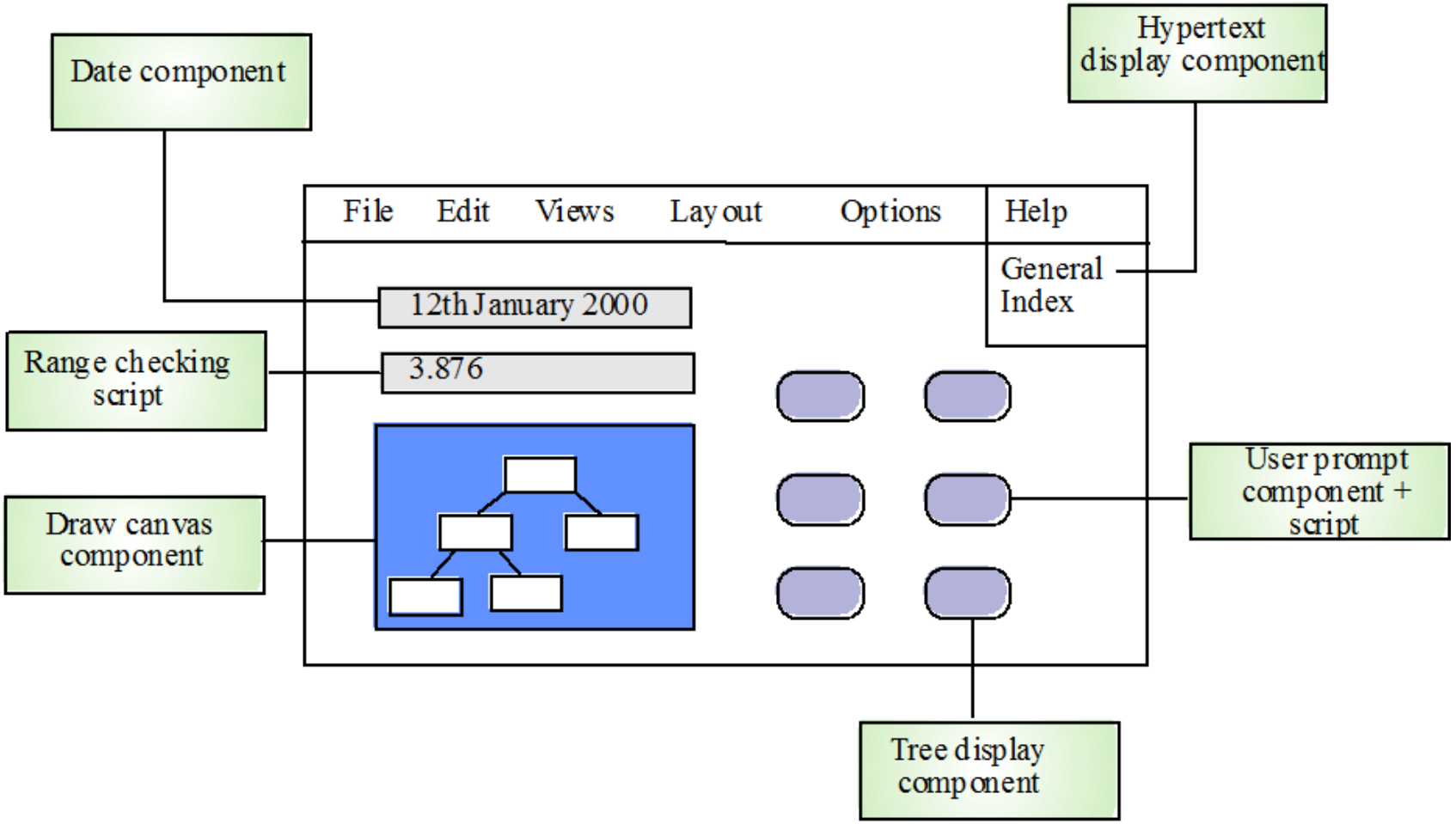
Prototyping key points

- A prototype can be used to give end-users a concrete impression of the product's capabilities
- Prototyping is becoming increasingly used for product development where rapid development is essential
- Throw-away prototyping is used to understand the product requirements
- Rapid development of prototypes is essential. This may require leaving out functionality or relaxing non-functional constraints
- Visual programming is an inherent part of most prototype development methods

Visual programming

- Scripting languages such as Visual Basic support visual programming where the prototype is developed by creating a user interface from standard items and associating components with these items
- A large library of components exists to support this type of development
- These may be tailored to suit the specific application requirements

Visual programming (2)



Problems with visual development

- Difficult to coordinate team-based development
- No explicit software architecture
- Complex dependencies between parts of the program can cause maintainability problems