Софтуерен процес

Софтуерен процес. Модели на софтуерен процес. Примери.

Съдържание

- Софтуерен процес
- Модели на софтуерен процес
- Примери

Процес

- *Последователност от стъпки включващи дейности, ограничения и ресурси, които осигуряват постигането на някакъв вид резултат.*

- Софтуерен процес
  - *Структуриран набор от дейности, необходими за разработване на софтуерна система в срок и с високо качество.*

Процес

- Основни цели на процеса
  - *Ефикасност (Effectiveness)*
  - *Възможност за поддръжка (Maintainability)*
  - *Предсказуемост (Predictability)*
  - *Повторяемост (Repeatability)*
  - *Качество (Quality)*
  - *Усъвършенстване (Improvement)*
  - *Проследяване (Tracking)*

Процес

- Основни дейности
  - *Комуникация (Communication)*
  - *Планиране (Planning)*
  - *Моделиране (Modeling)*
  - *Конструиране (Construction)*
  - *Внедряване (Deployment)*

Процес

- Допълнителни дейности
  - *Следене и управление на софтуерния продукт (Software product tracking and control)*
  - *Управление на риска (Risk management)*
  - *Осигуряване на качеството (Software quality assurance)*
  - *Измерване (Measurement)*
  - *Управление на софтуерната конфигурация (Software configuration management)*
  - *Управление на повторното използване (Reusability management)*
  - *Подготовка и генериране на работни продукти (Work product preparation and production)*

Модел на софтуерен процес

- *A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.*

- *Опростено описание на софтуерен процес, което е представено от определена перспектива*

Модел на софтуерен процес

- Цели
  - *Формиране на общо разбиране за участниците в разработването на софтуер за дейностите, ресурсите и ограниченията*
  - *Намиране на несъответствия, излишества и пропуски в процеса от разработващия екип*
  - *Намиране и оценяване на подходящи дейности за постигане на целите на процеса*
  - *Адаптиране на общ процес към отделна ситуация, в която ще се приложи*

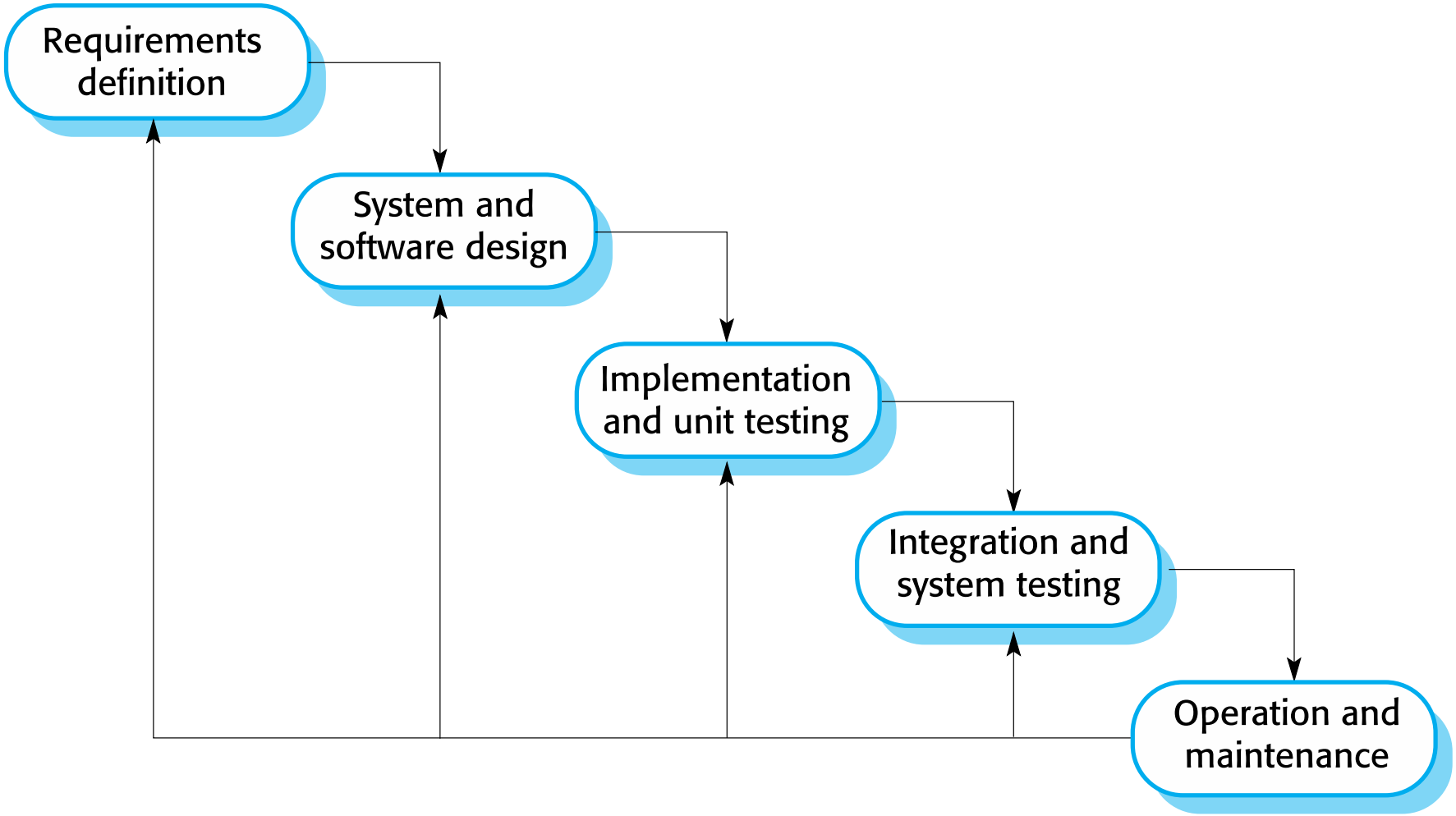
Plan-driven and agile processes

- Plan-driven processes are processes where all of the process activities are planned in advance and progress is measured against this plan.
- In agile processes, planning is incremental and it is easier to change the process to reflect changing customer requirements.
- In practice, most practical processes include elements of both plan-driven and agile approaches.
- ***There are no right or wrong software processes!***

Модел на софтуерен процес

- Каскаден (водопад) - *The Waterfall model*
- Модел на бързата разработка - *The Rapid Application Development (RAD) model*
- Фазови (еволюционни) модели - *Evolutionary process models*
  - Постъпков (инкрементален) модел
  - Итеративен модел
- Прототипен модел - *The Prototyping model*
- Спираловиден модел - *The spiral model*

Модел на водопада – *The Waterfall model*



Модел на водопада – *The Waterfall model*

- Характеристики
  - *ясно разграничен процес, който е лесен за разбиране*
  - *всяка дейност трябва да бъде напълно завършена, преди да се премине към следваща*
  - *ясно са дефинирани входовете и изходите на дейностите, както и интерфейсите между отделните стъпки*
  - *ясно са дефинирани ролите на разработчиците на софтуер*

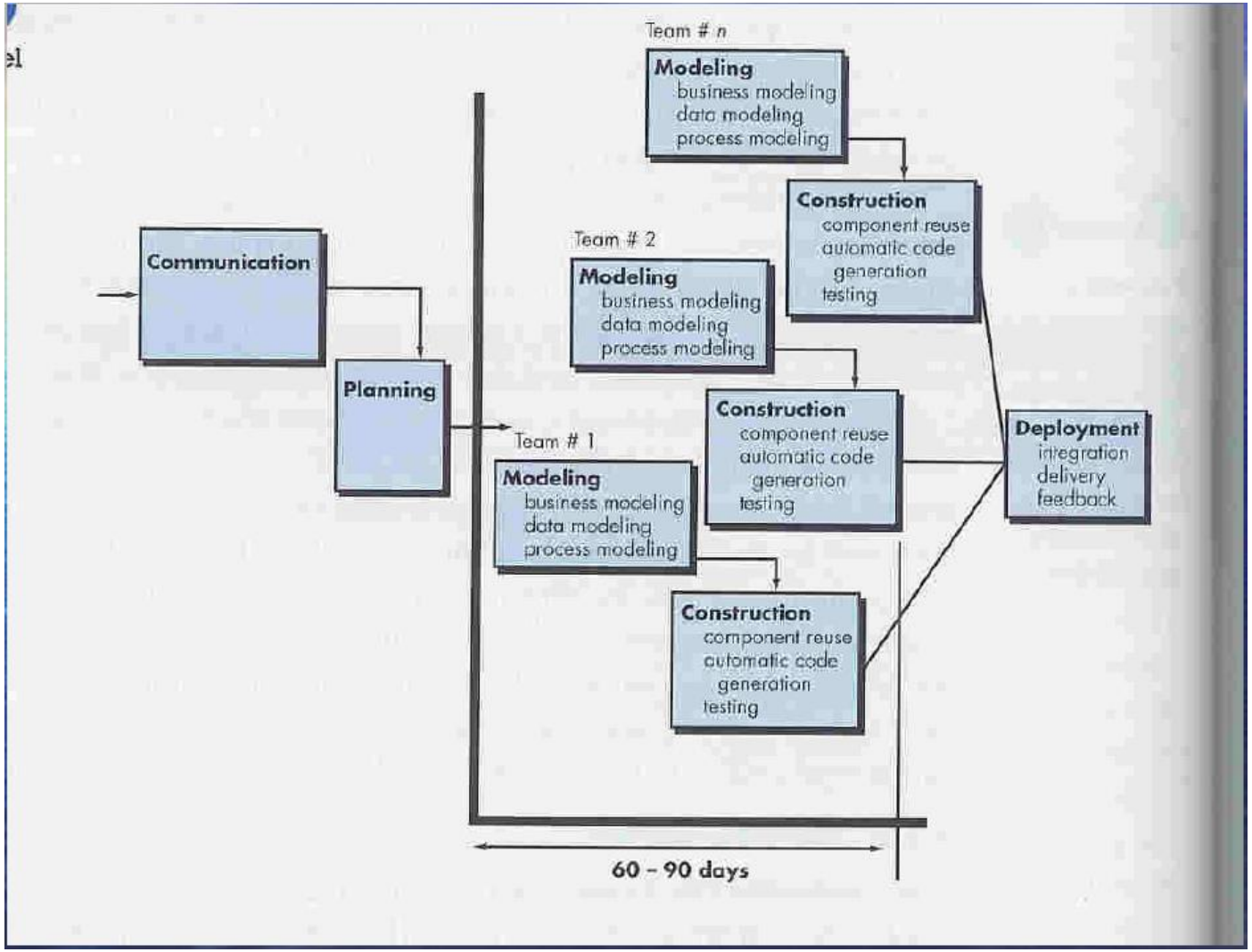
Модел на водопада – *The Waterfall model*

- Прилагане
  - *Когато изискванията са осъзнати и ясно формулирани в началото*
  - *Когато проектите са ясно организирани*
  - *При повторяеми проекти и/или големи проекти, за които времето и бюджетът не са критични*

Модел на водопада – *The Waterfall model*

- Проблеми
  - *Трудно е за потребителя да формулира всичките си изисквания в началото*
  - *Разделянето на проекта на отделни етапи не е гъвкаво - трудно е да се реагира на променящите се изисквания на клиента*
  - *Моделът е произлязъл от областта на хардуера и не отчита същността на софтуера като творчески процес на решаване на проблем (с итерации и връщане назад)*

Модел на бързата разработка



Модел на бързата разработка

- **Комуникация / Планиране**
- **Моделиране**
  - *бизнес моделиране (Business modeling)*
  - *моделиране на данните (Data modeling)*
  - *моделиране на процес (Process modeling)*
- **Конструиране**
- **Внедряване**

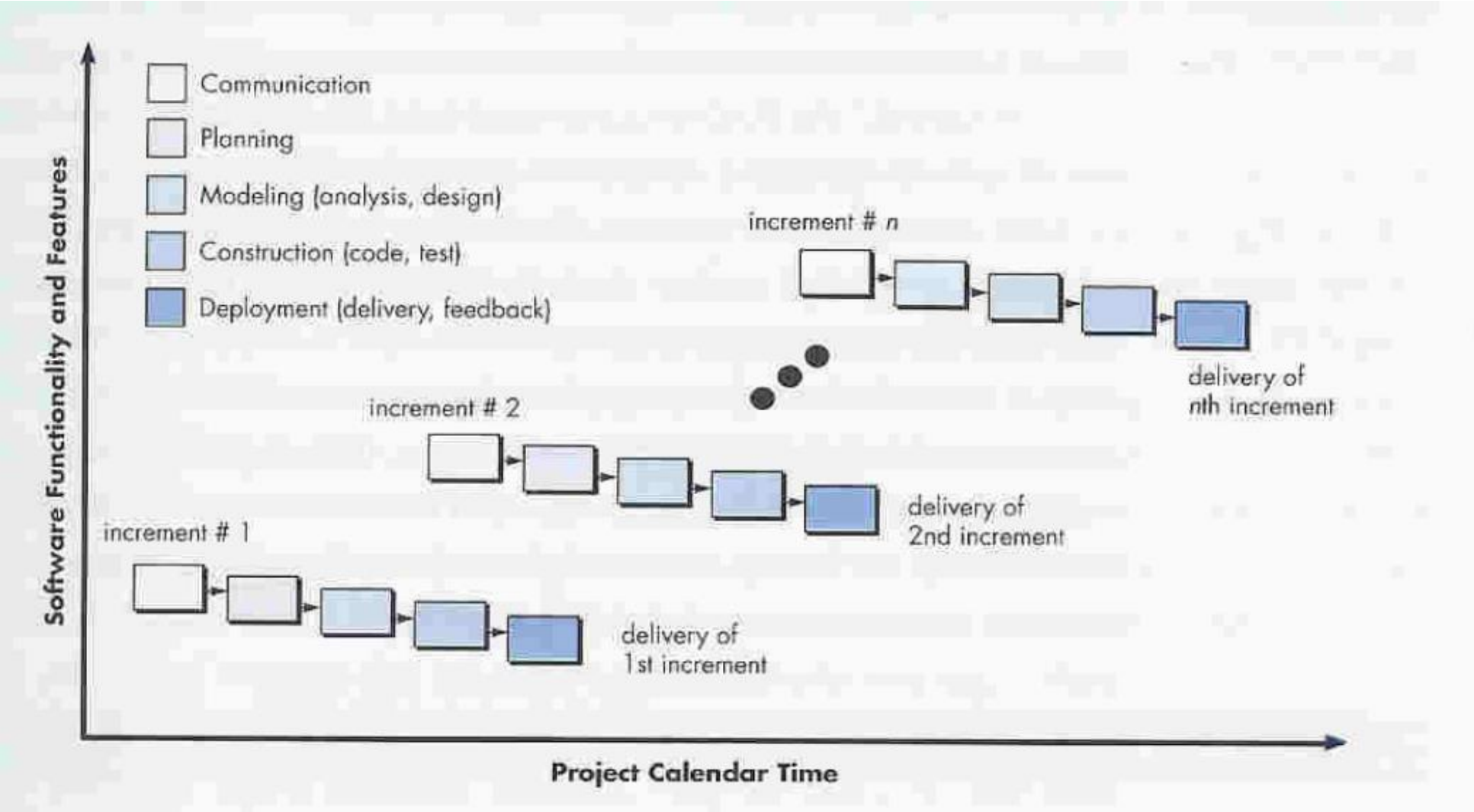
Модел на бързата разработка

- Предимства
  - *Подобрена гъвкавост, тъй като разработчиците могат да се адаптират към необходимите промени по време на изграждане*
  - *Може да се създават бързи итерации, които намаляват времевите рамки*
  - *Може да се извършват интеграции в началните етапи и да се използва повторно кода във всеки момент (по-кратко време за тестване)*
  - *По-добро качество (от гледна точка на потребителите - бизнес функционалността) тъй като взаимодействат с развиващи се прототипи*

Модел на бързата разработка

- Недостатъци
  - *За големи приложения, подлежащи на разделяне на модули (значителни човешки ресурси)*
  - *Липса на акцент върху качествените атрибути на софтуера*
  - *Фокусът върху прототипите може да доведе до непрекъснато незначителни промени в отделни компоненти и игнориране проблемите на системната архитектура.*
  - *RAD изисква участие на клиента на много етапи, което прави процеса по-сложен от други методологии.*

Постъпков (инкрементален) модел

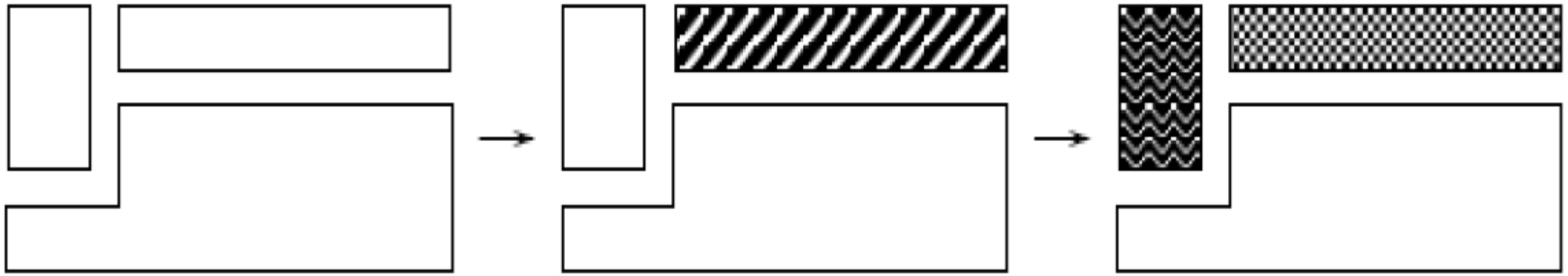


Постъпков (инкрементален) модел

- Системата не се доставя като едно цяло, а вместо това процесът на разработка и доставянето са разделени на стъпки, като всяка стъпка доставя само част от цялата функционалност
- На идентифицираните потребителски изисквания се присвояват приоритети и тези с по-висок се реализират в първите стъпки
- След като започне разработката на една стъпка, изискванията не се променят

Итеративен модел

- В самото начало доставя цялостната софтуерна система, макар и част от функционалността да е в примитивна форма
- При всяка следваща итерация не се добавя нова функционалност, а само се усъвършенства съществуващата



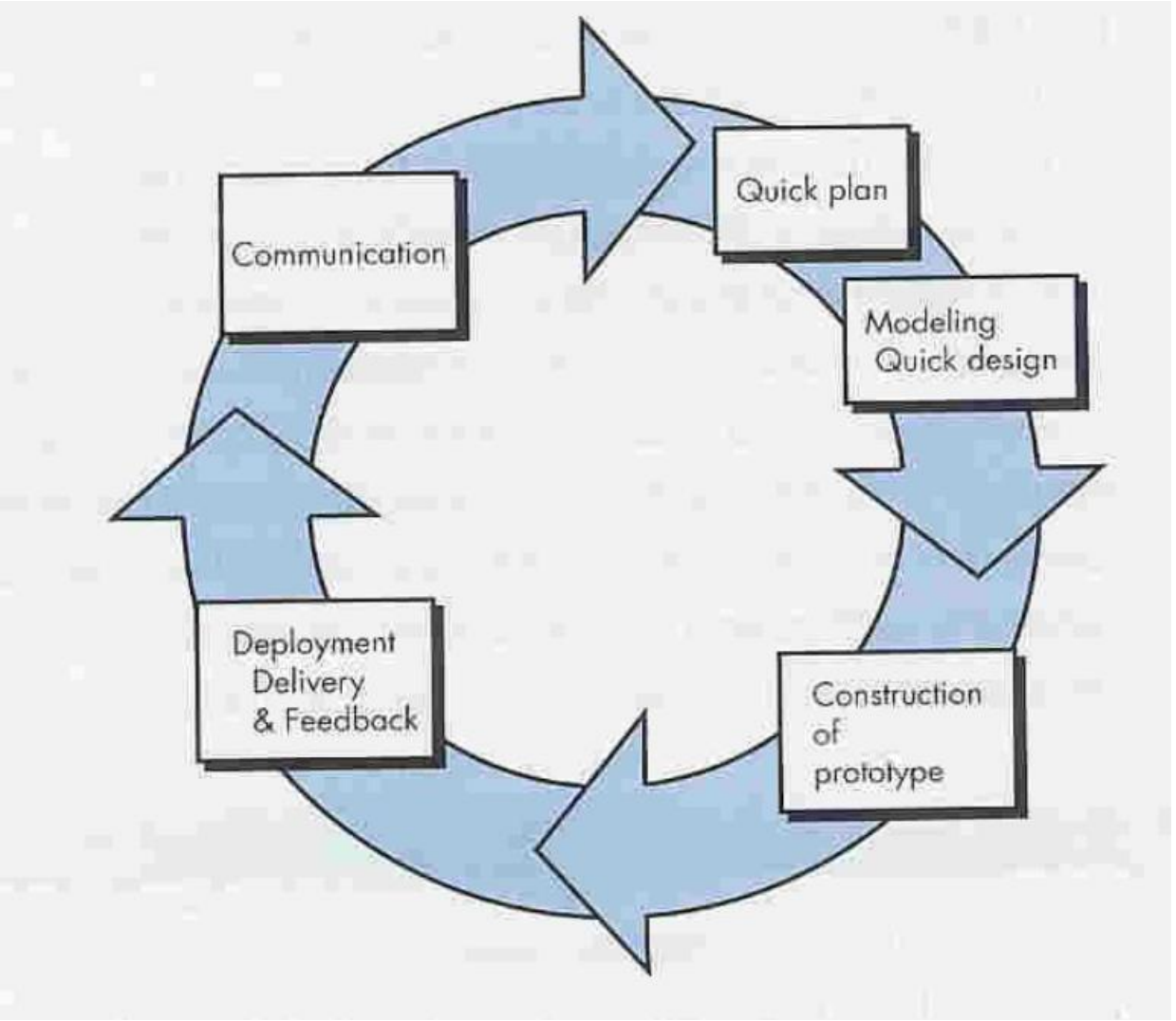
Фазови (еволюционни) модели

- Предимства
  - *Клиентът може да използва системата, преди да е готов целият продукт*
  - *Функционалностите от цялата система, които са с най-висок приоритет, са тествани най-много*
  - *В разработването на по-ранните версии участват по-малко хора и в зависимост от обратната връзка, могат да се присъединят още разработчици на следващите итерации*
  - *Итерация, която изисква използването на нова технология или продукт може да се планира по-късно*

Фазови (еволюционни) модели

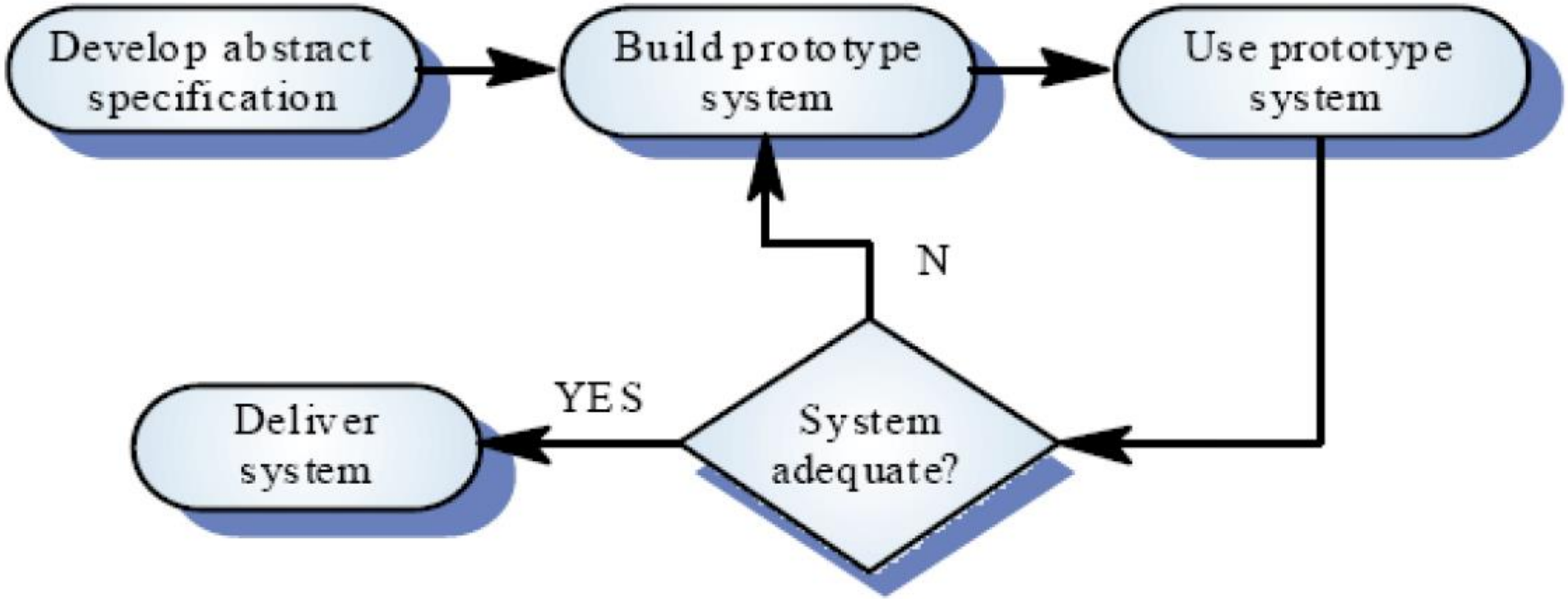
- Проблеми
  - *Необходимостта от активно участие на клиентите по време на изпълнение на проекта може да доведе до закъснения*
  - *Уменията за комуникация и координация са от особено голямо значение при разработката*
  - *Неформалните заявки за подобрения след завършването на всяка стъпка могат да доведат до объркване*
  - *Може да доведе до т. нар. “scope-creep” – бавно и постепенно разширяване на обхвата на приложението*

Прототипен модел



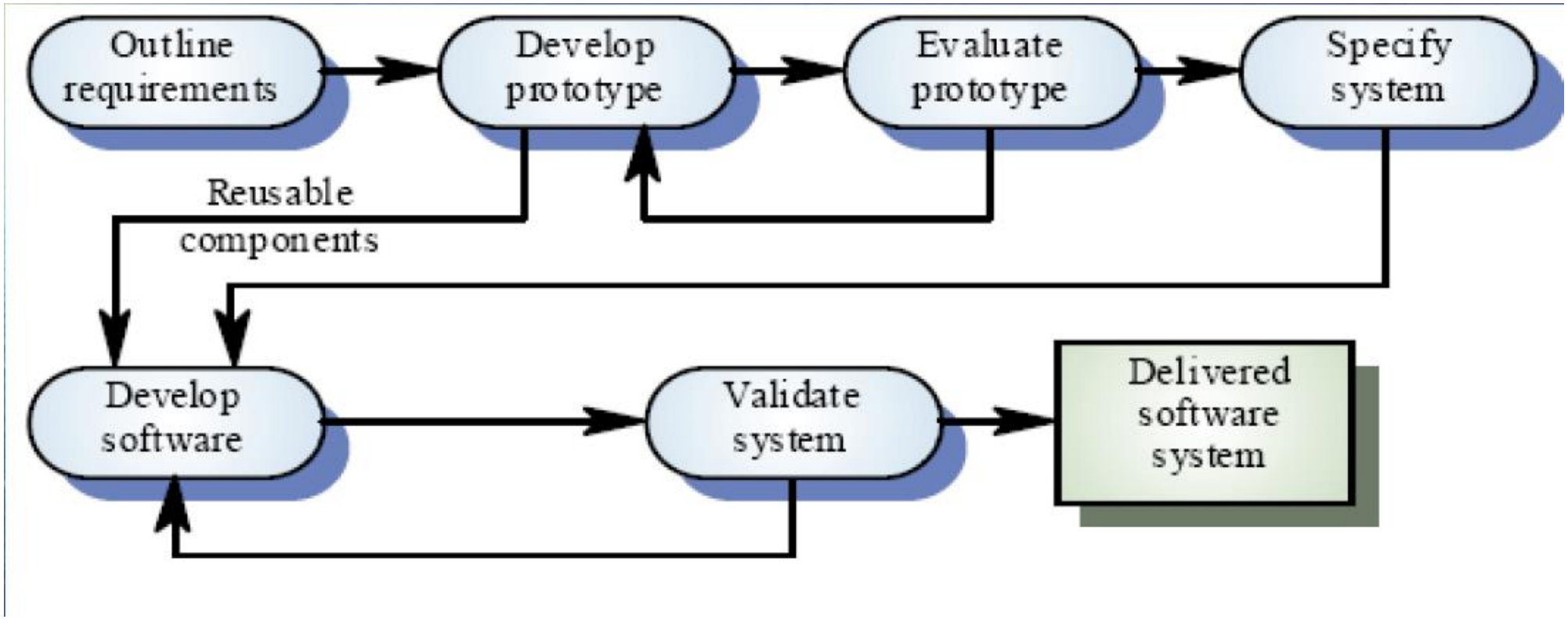
Прототипен модел

- Еволюционен прототип
  - *Цел: да достави работеща система на крайния потребител*



Прототипен модел

- Изхвърлен (throw-away) прототип
  - *Цел: да подпомогне специфицирането на изискванията към софтуера*



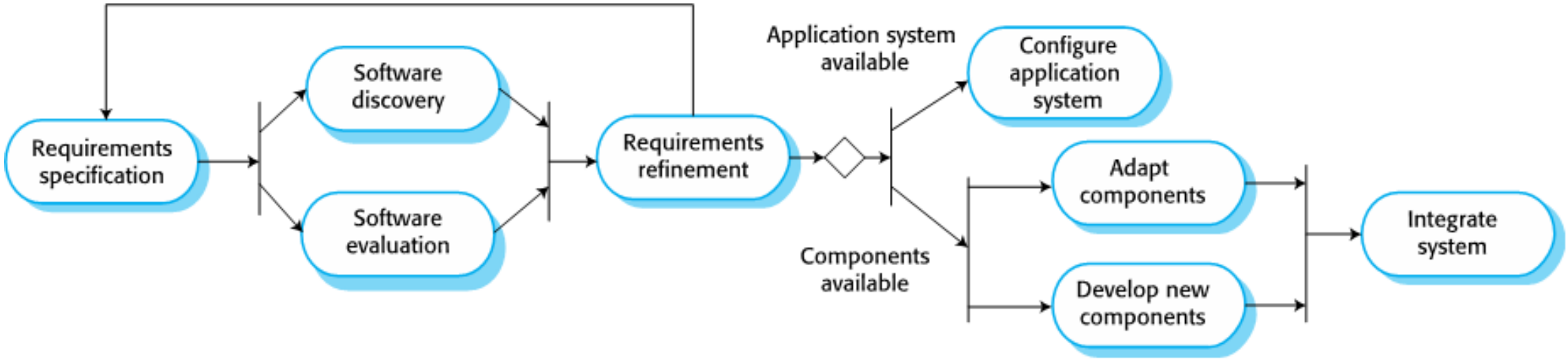
Прототипен модел

- Създаване на основните потребителски интерфейси, без да има някакво значително кодиране
- Разработване на съкратена версия на системата, която изпълнява ограничено подмножество от функции
- Използване на съществуваща система или компоненти от система, за да се демонстрират някои функции, които ще бъдат включени
- Прилага се в проекти, където не са достатъчно ясни изискванията на потребителите и дизайнът на софтуерната система
  - *Както самостоятелно, така и в комбинация с други модели на процеси*

Прототипен модел

- Проблеми
  - *Прототипният модел може да използва значителни ресурси, а като резултат прототипът да не успее да удовлетвори очакванията*
  - *Прототипът може да доведе до лошо проектирана система, ако самият той стане част от крайния продукт*
  - *Прототипният модел не е подходящ за използване при разработване на софтуерни системи, където проблемът е добре разбран и интерфейсът е ясен и прост*

Reuse-oriented software engineering



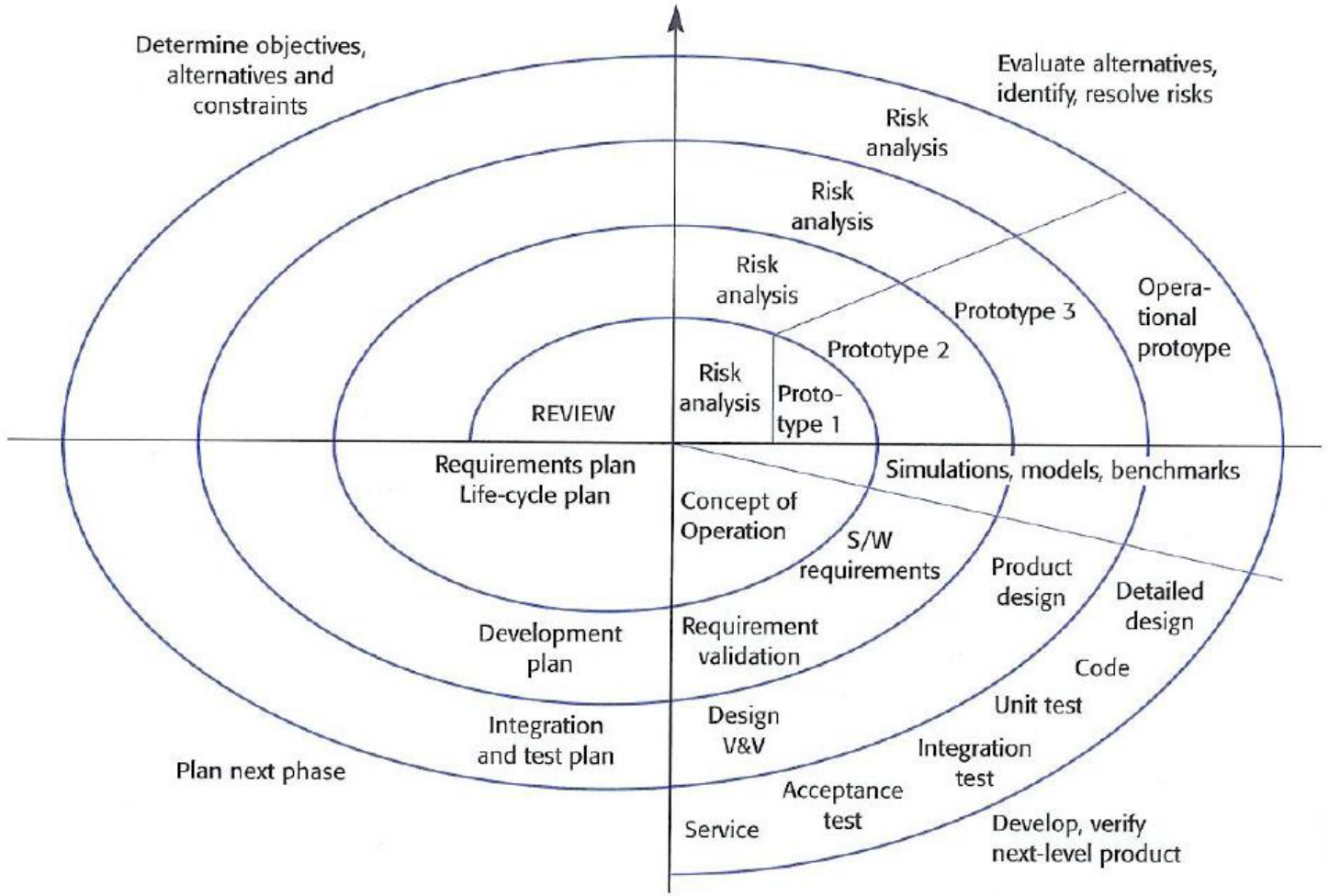
Integration and configuration

- Based on software reuse where systems are integrated from existing components or application systems (sometimes called COTS -Commercial-off-the-shelf) systems).
- Reused elements may be configured to adapt their behaviour and functionality to a user's requirements
- Reuse is a standard approach for building many types of business system

Types of reusable software

- Stand-alone application systems (sometimes called COTS) that are configured for use in a particular environment.
- Collections of objects that are developed as a package to be integrated with a component framework such as .NET or J2EE.
- Web services that are developed according to service standards and which are available for remote invocation.

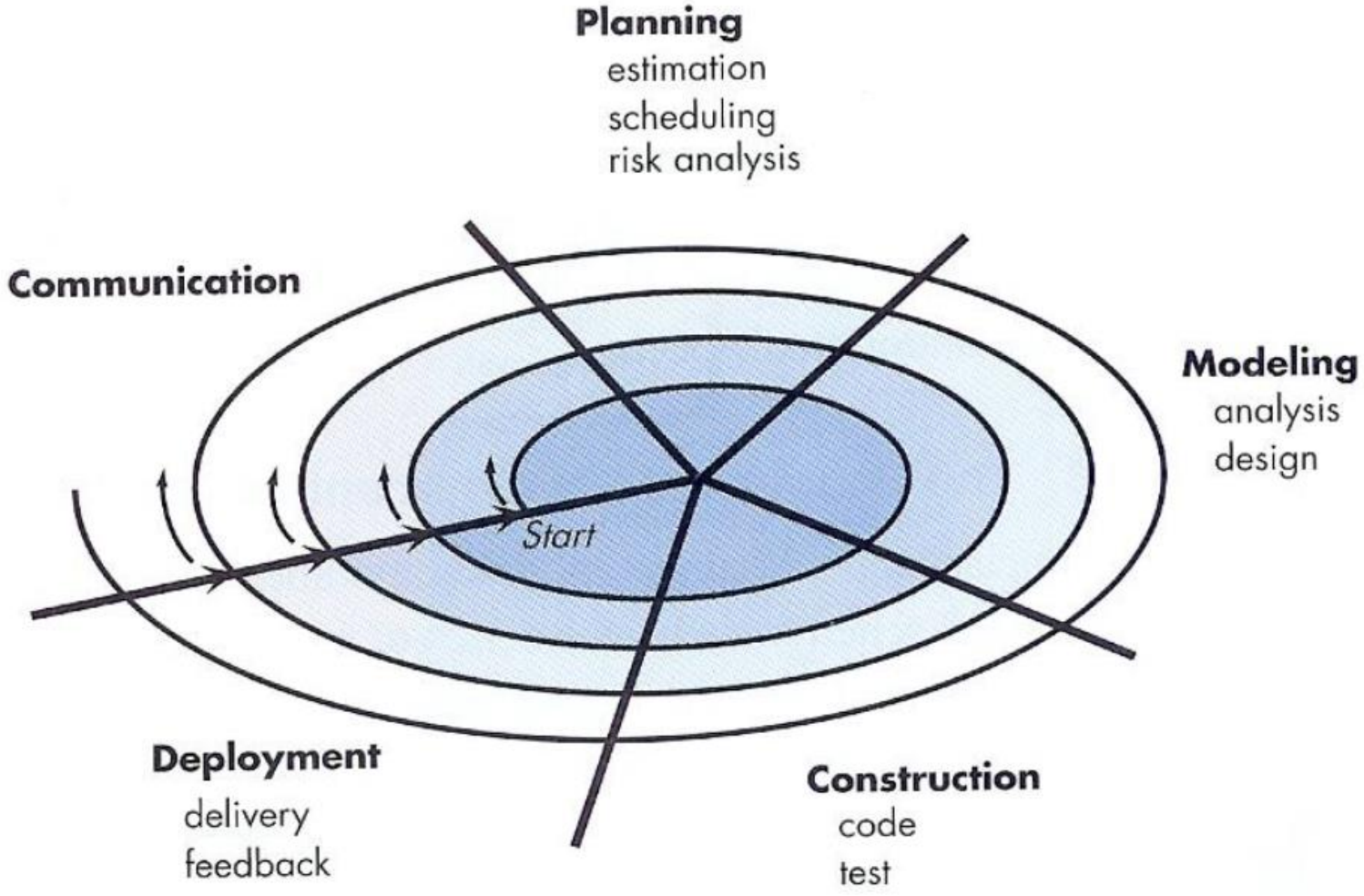
Спираловиден модел



Спираловиден модел

- Спираловидният модел е еволюционен модел на софтуерен процес, който съчетава прототипния модел и модела на водопада
- Движещият фактор е анализ на риска
- Основни характеристики:
  - *итеративен/цикличен подход*
  - *има множество от точки на прогреса (anchor point milestones)*

Спираловиден модел



Спираловиден модел

- Установяване на целите
  - *определят се цели, алтернативи и ограничения на текущата фаза*
- Оценка на рисковете и намаляването им
  - *идентифицират се и се анализират потенциалните рискове*
  - *предприемат се действия за намаляването или елиминирането им*
- Разработване и валидиране
  - *избира се модел за разработване на текущата фаза*
- Планиране
  - *преглежда се и се анализира текущото състояние*
  - *планира се следващото завъртане по спиралата*

Спираловиден модел

- Проблеми
  - *Изисква се участието на разработчици с компетентност за оценка на рисковете*
  - *Ако не се идентифицира и открие някой основен риск, това може да доведе до неуспех*
  - *Може да се окаже трудно да се убедят клиентите, че процесът на разработка е контролируем, а не е безкраен цикъл*

Метод на Формална трансформация

- Основава се на математическо трансформиране на спецификацията на системните изисквания до изпълнима програма
  - *При трансформирането трябва да се запази коректността и ясно да се покаже, че изпълнимата програма съответства на спецификацията*
- Спецификацията на софтуерните изисквания се усъвършенства вдетайлна формална спецификация, изразена с математическа нотация
  - *Дейностите моделиране и конструиране са заменени с разработване и прилагане на трансформации*

Метод на Формална трансформация

In the transformational development process. The formal specification is refined through a series of transformational into a program as shown fig 2.17.

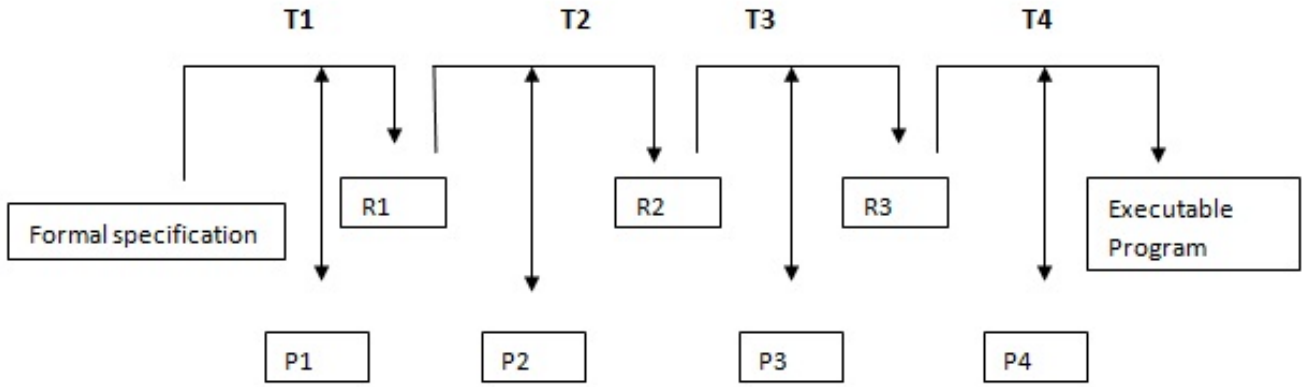


Fig. 2.17, Formal Transformation

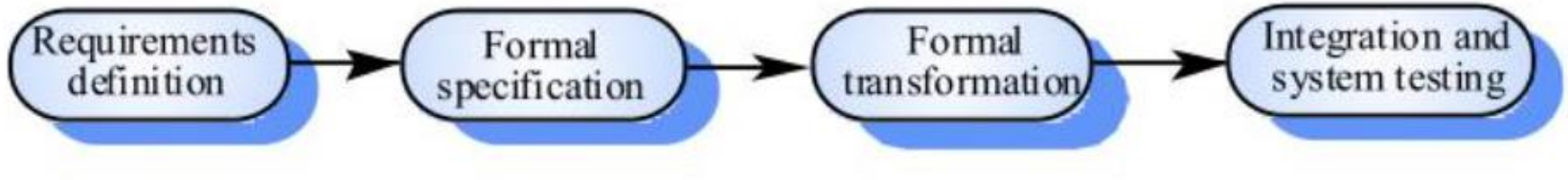
T_1, T_2, T_3, T_4, Transformational steps

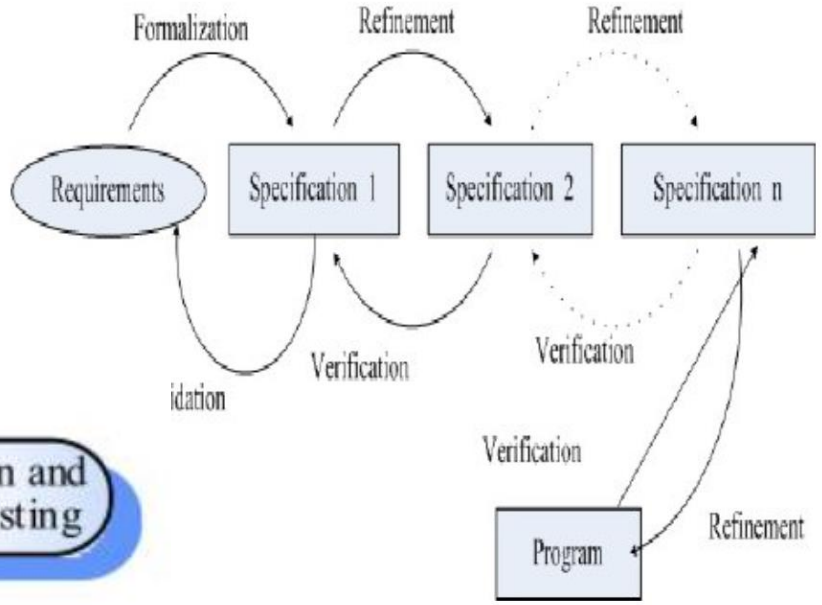
P_1, P2, P3, P_4, Process steps to transform the formal specification

R_1, R_2, R_3, Refinements

Метод на Формална трансформация

- Прилагане
  - *При разработването на софтуерни системи, които са свързани с поддръжката на човешки живот и за които трябва да са гарантирани сигурността и отказоустойчивостта на софтуера, преди да започне реалното му използване*





Метод на Формална трансформация

- Проблеми
  - *разработването на формални модели е скъп и бавен процес*
  - *необходими са разработчици със специализирани умения, както и обучение за това, как да се прилага формалната трансформация*
  - *поради сложността си изготвените модели трудно могат да се използват*
  - *някои от аспектите на софтуерна система е трудно е да се специфицират формално – например потребителският интерфейс*

Модел на софтуерен процес

- Изборът зависи основно от два фактора:
  - *Организационната среда*
  - *Същността на приложението*

Въпроси ?

Допълнителни материали

- *Ian Sommerville, Software Engineering, 10th Ed.,* ***Chapter 2***