Modelling Techniques

Unit 7

Requirement Engineering

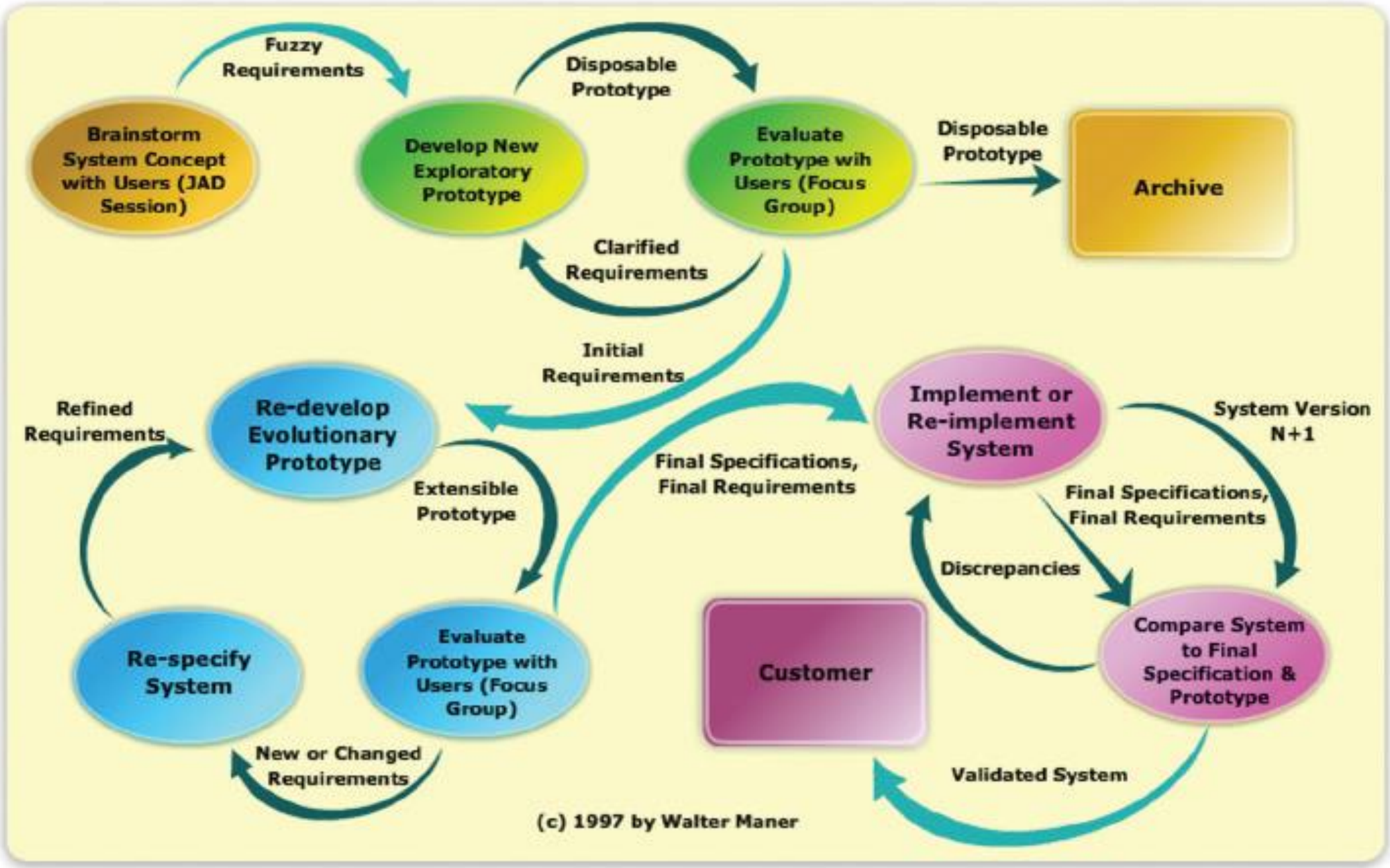
1.0 Introduction

- The software prototyping is a process that can be defined as:
  - Helps software developer to find and specify requirements, feasibility study Of development strategies, design user interface, narrow the communication gap among stakeholders, increase level of end-user involvement in software development process and visualize future application for the stakeholders.
  - Let the users have a first idea Of the completed program. Prototyping as a useful technique can be used in most of software development methodologies.
  - Process of creating an incomplete model of the future full-featured software program.

- Prototyping can be accomplished through a series of sessions with participation of various software system stakeholders.
- Due to rapid development of the software prototype in the screen-based prototyping approach, software developers will be profited by early feedback of software users and stakeholders.

- Prototyping session starts with an initial knowledge of problem domain and complementary knowledge will be acquired during this prototyping session.
- In prototyping sessions, only uses-cases related to the participants in that session will be prototyped and other use-cases will be left for the subsequent prototyping sessions with participation of relevant stakeholders.

A sample process of prototyping

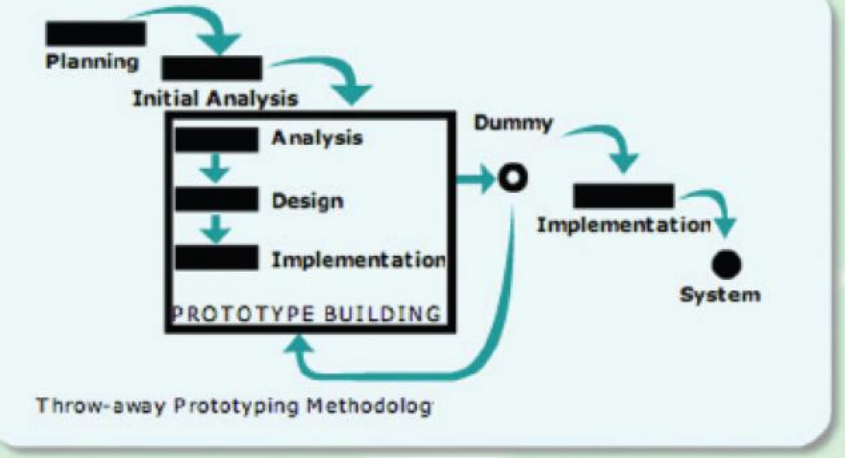


2. The Various Types of Prototypes

There are five types of prototypes for different reasons in software engineering:

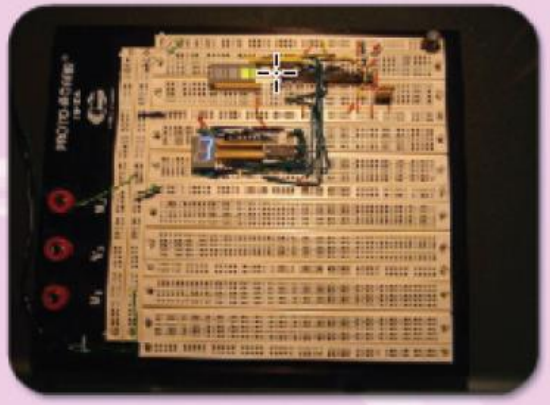
1. Throwaway prototype versus evolutionary prototype
2. A horizontal prototype versus vertical prototype
3. Architectural prototype versus requirement prototype
4. Textual prototype versus visual prototype
5. Executable prototype versus non-executable prototype

2.1 Throwaway prototype versus evolutionary prototype

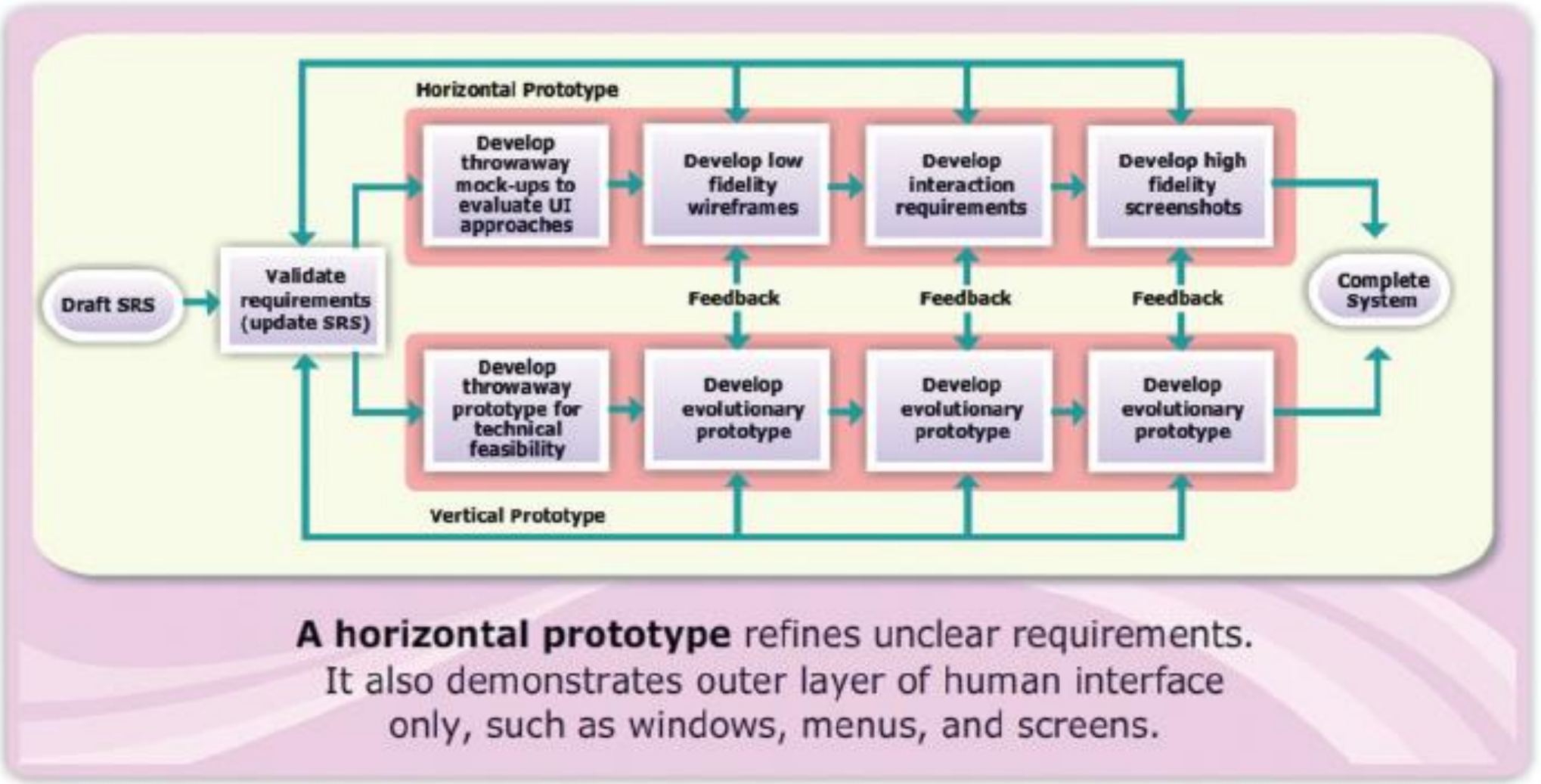


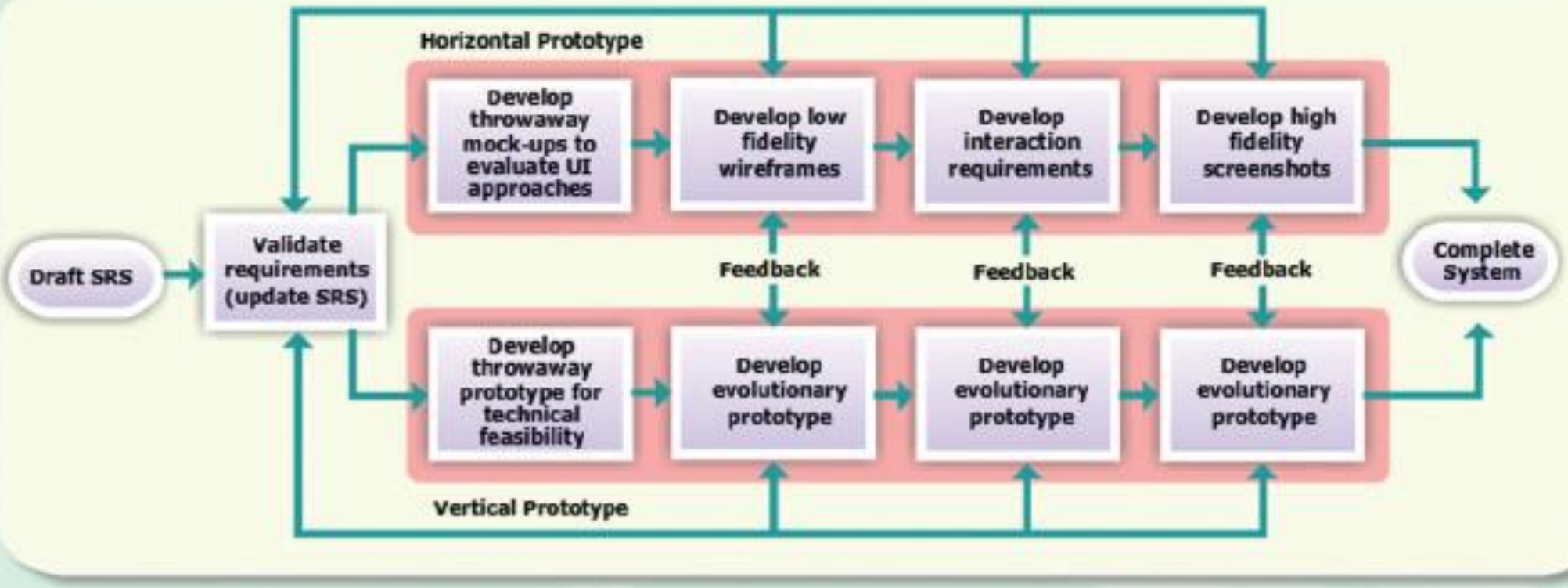
Throwaway prototypes are prototypes that will be discarded when the final system has been developed. It uses storyboards, animations or drawings.

Evolutionary prototypes are made available to users early in the development process and then extended to produce the final system.



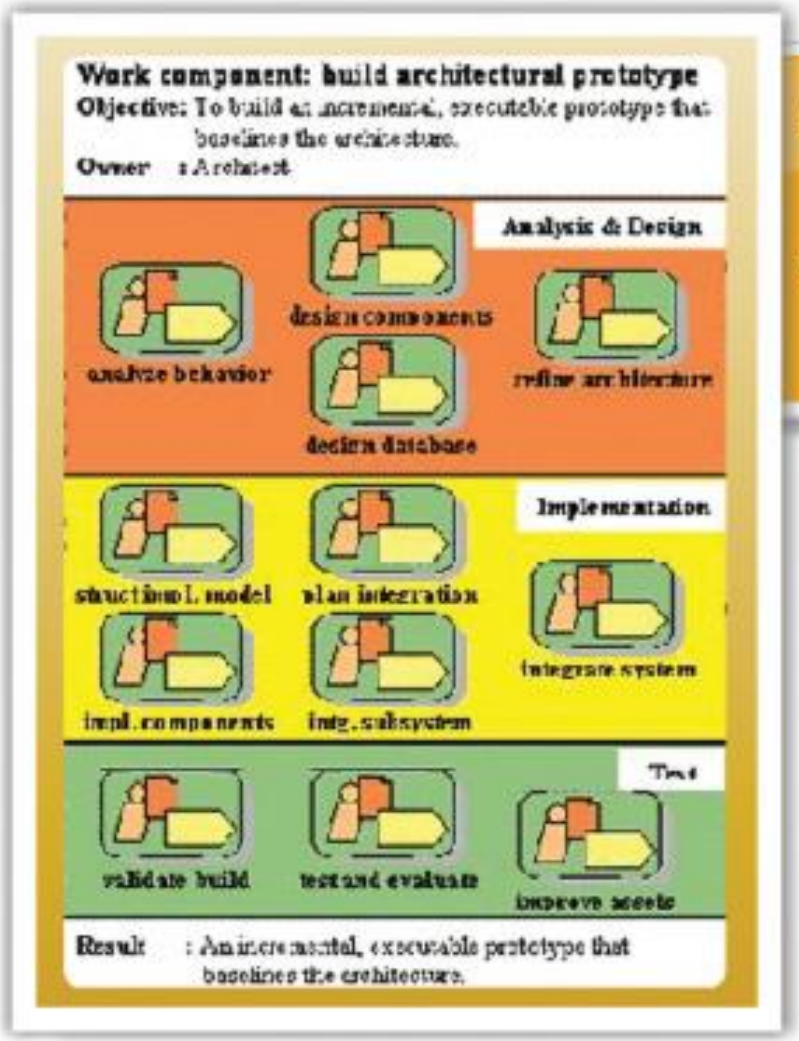
2.2 A horizontal prototype versus vertical prototype



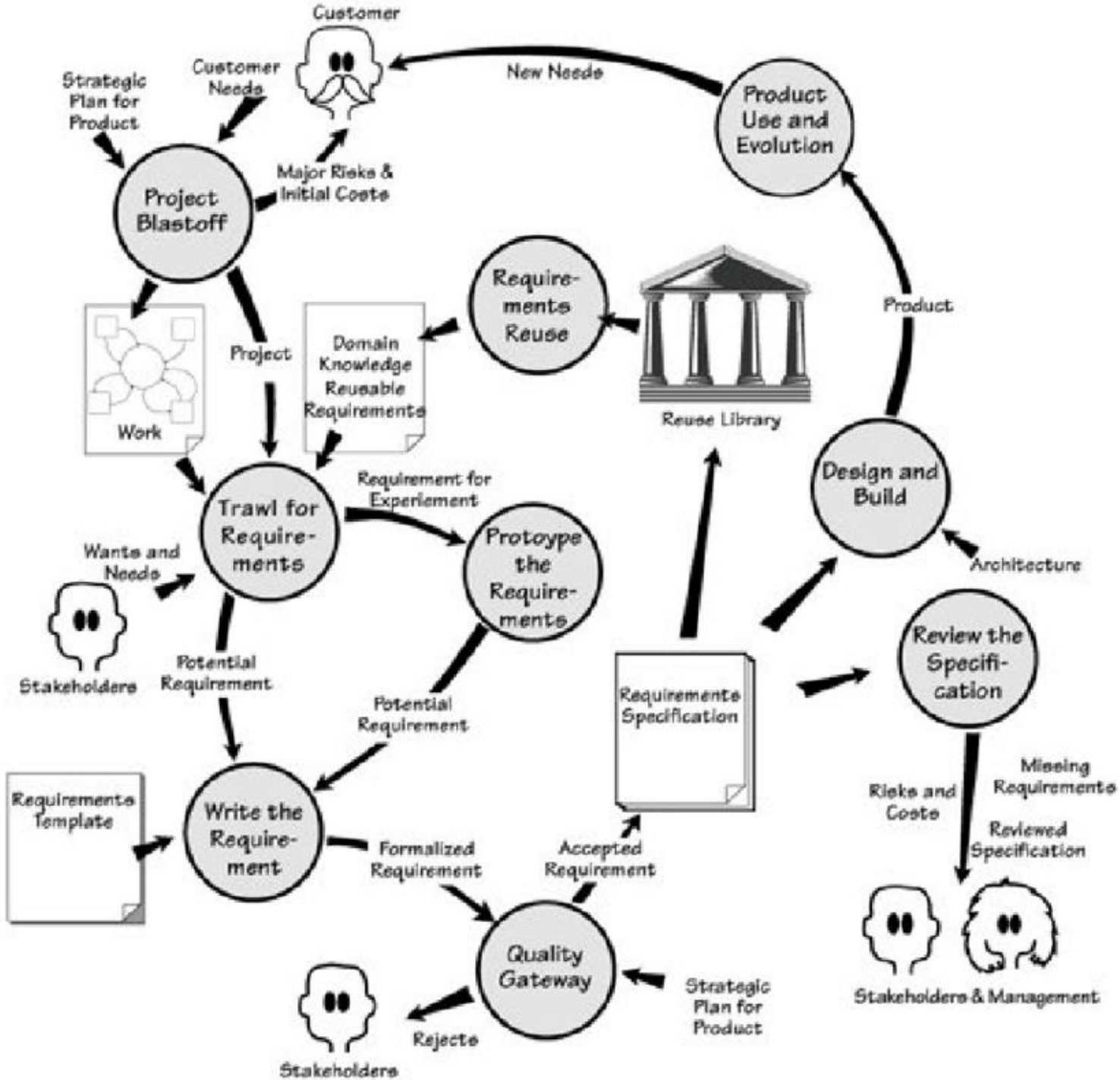


A vertical prototype refines database design and test key components. It also demonstrates a working, though incomplete, system for key functions. It is used, for example, in algorithm optimisation.

2.3 Architectural prototype versus requirement prototype

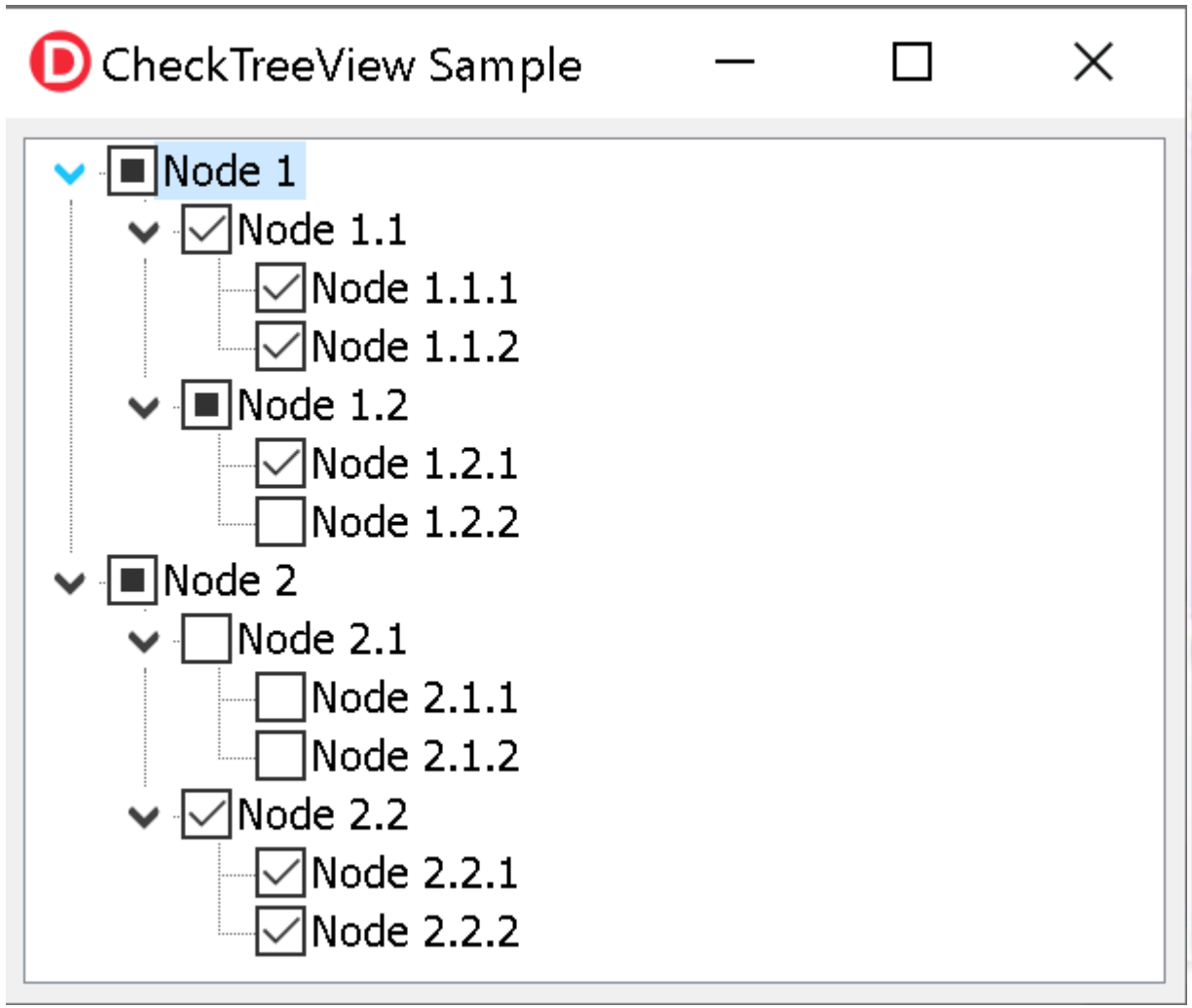


Architectural prototypes are focused on the performance and the feasibility of the technology used.

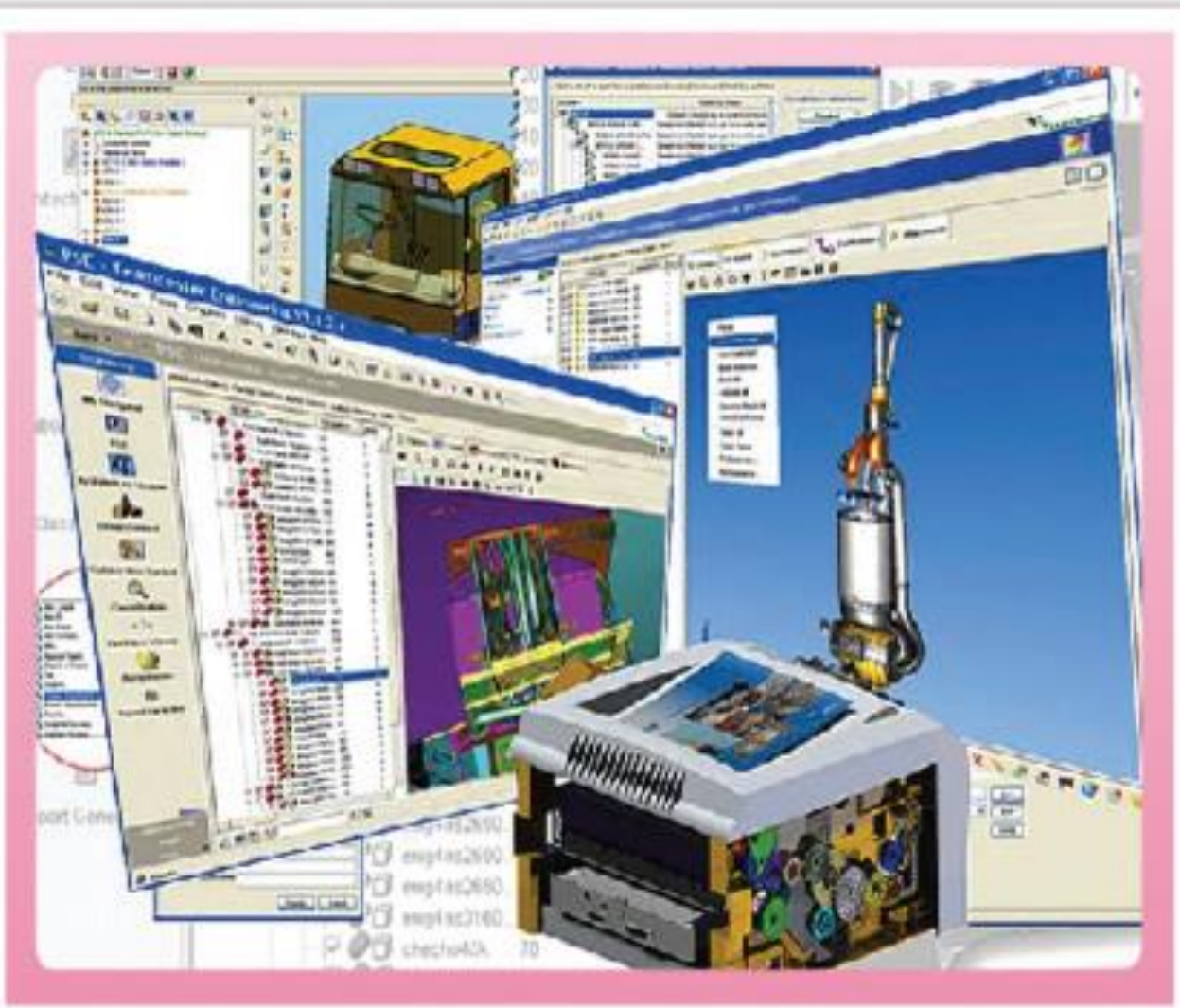


Requirements prototypes are suggested to be used in requirements acquisition and user interface design.

2.4 Textual prototype versus visual prototype

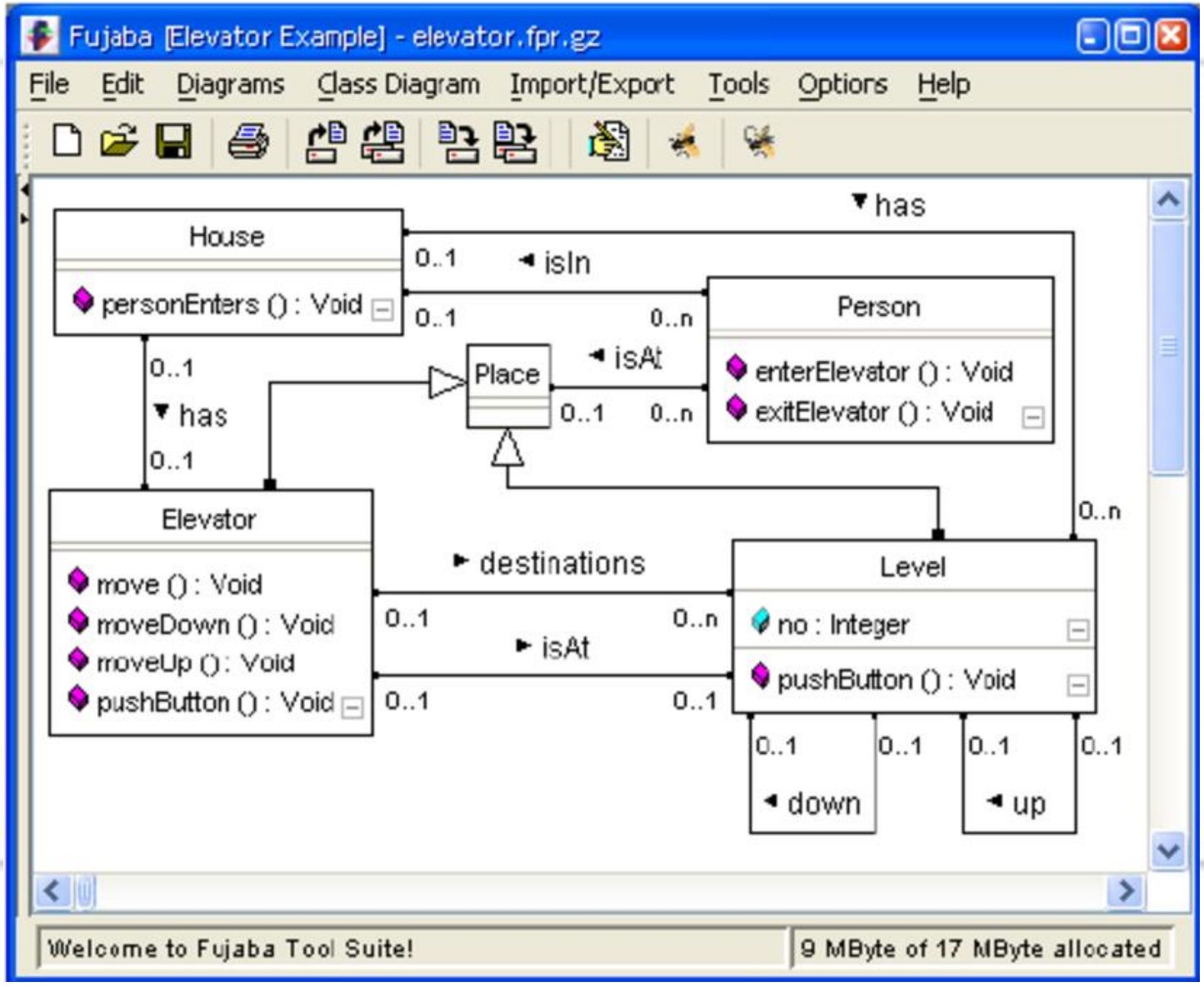


Textual methods consist of formal prototyping languages.



Visual prototyping methods include such techniques as diagrams, charts, storyboards and scenarios.

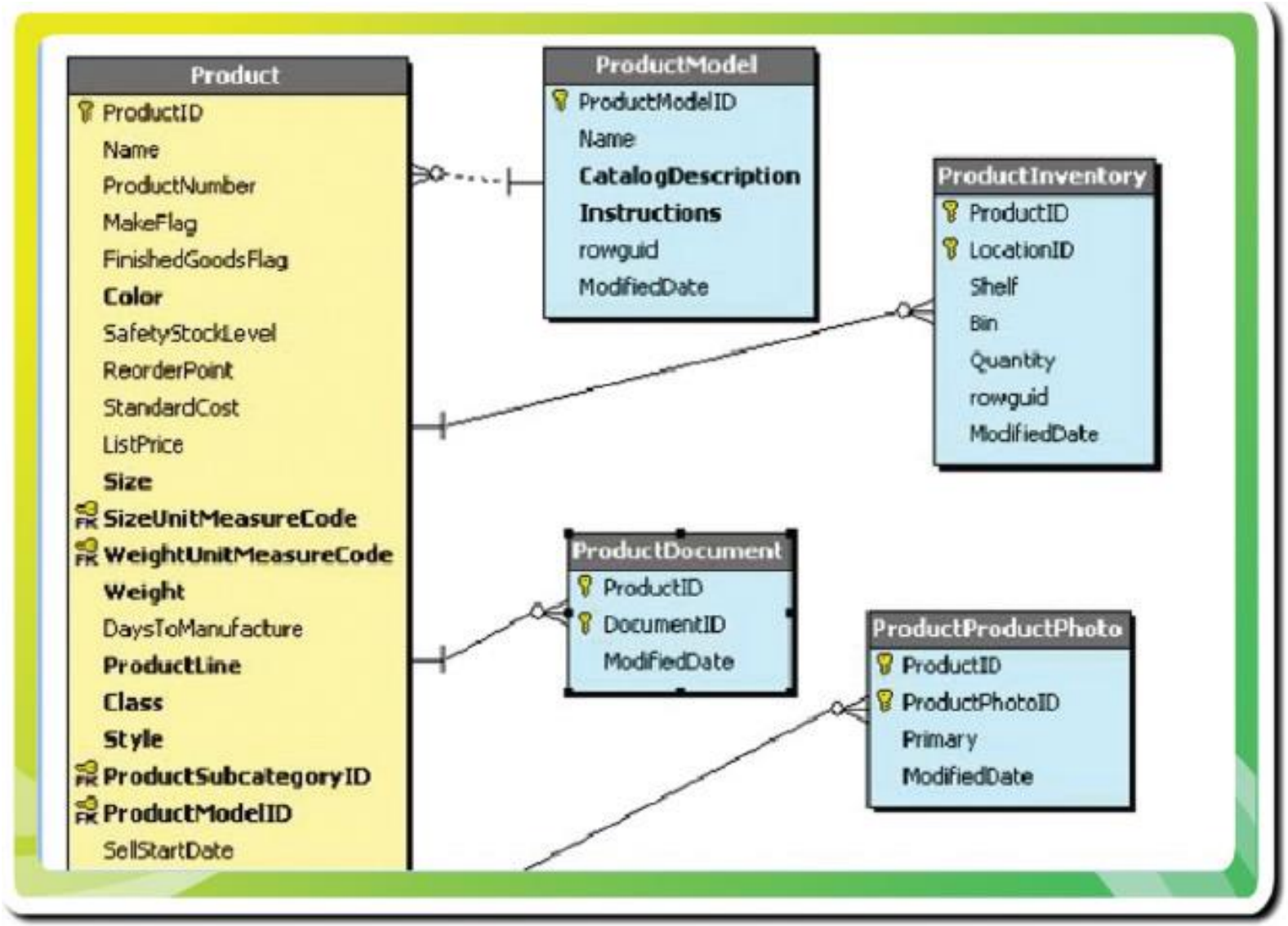
2.5 Executable prototype versus non-executable prototype



Executable prototypes are prototypes that are built as software and constructed using a high level programming language.

Non-executable prototypes are like paper prototypes and other mock-ups of the system.

3.0 Static Modelling (Screen Diagrams)



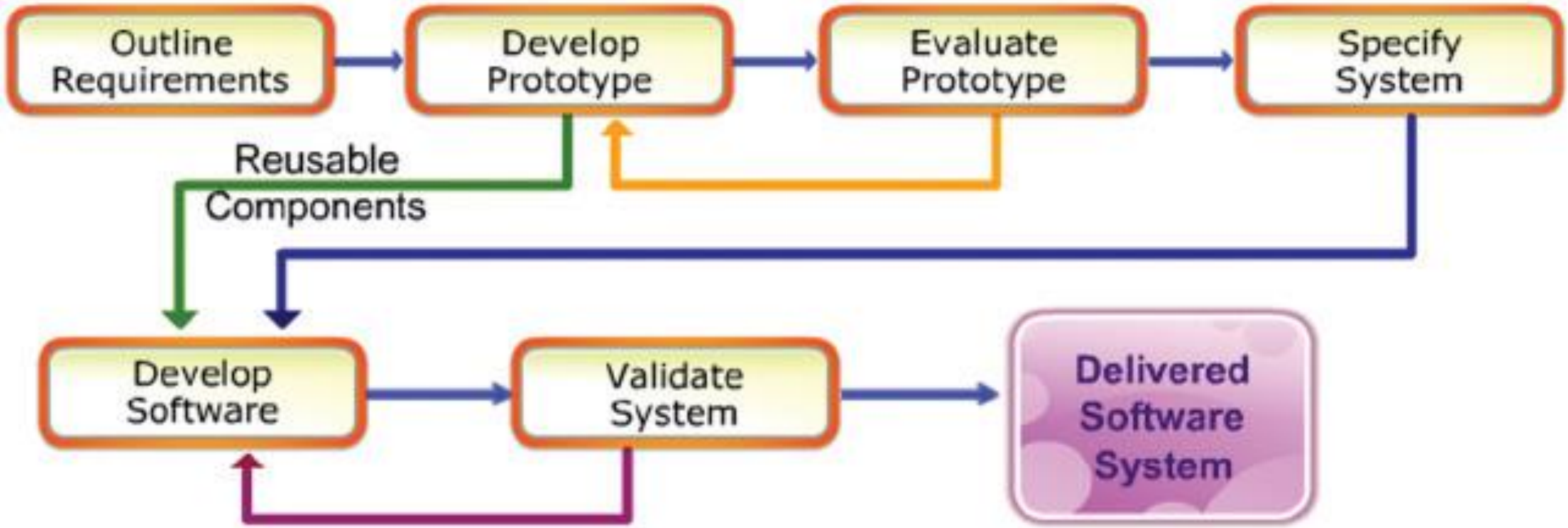
- Screen diagrams or screen-based prototyping prepares software developers with a very easy to use prototyping concepts and toolset which helps them to design a prototype of the software system at hand in a rapid and straight forward order.
- Screen-based prototyping employs the use-case driven approach in developing prototypes.
- One prototype is designed for each use case for prototyping. Storylines and user classes are created in order to show the relationship among use-cases, user’s actions and interaction between users and use-cases.

- A use-case is a sequence of actions that an actor performs within a system to achieve a certain goal.
- From this point of view, a software system can be decomposed into several use-cases.
- These use-cases may be changed during requirements engineering process.
- Changing use-cases may cause changing storyline changes, screen changes and even requirements changes.
- Also changing storylines may lead to changing screens and requirements.
- So, The completion of the screen-based prototyping approach, its toolset and its usability in a real industrial setting requires further research, which focuses on refinement of the framework, its design and implementation.

4.0 Dynamic Modelling (Throw-Away Prototyping)

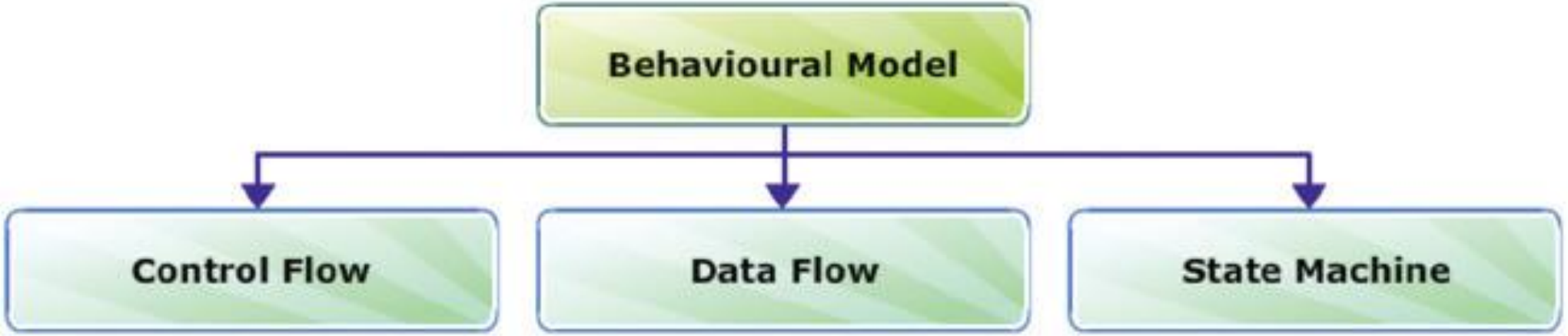
- Dynamic modelling is used to reduce requirements risk. The prototype is developed from an initial specification, delivered for experiment before then be discarded.
- The throw-away prototype should NOT be considered as a final system because:
  - Some system characteristics may have been left out in the process.
  - There is no specification for a long-term maintenance.
  - The system will be poorly structured and very difficult to maintain.

- an example of a throw-away prototyping



5.0 Behavioural Modelling

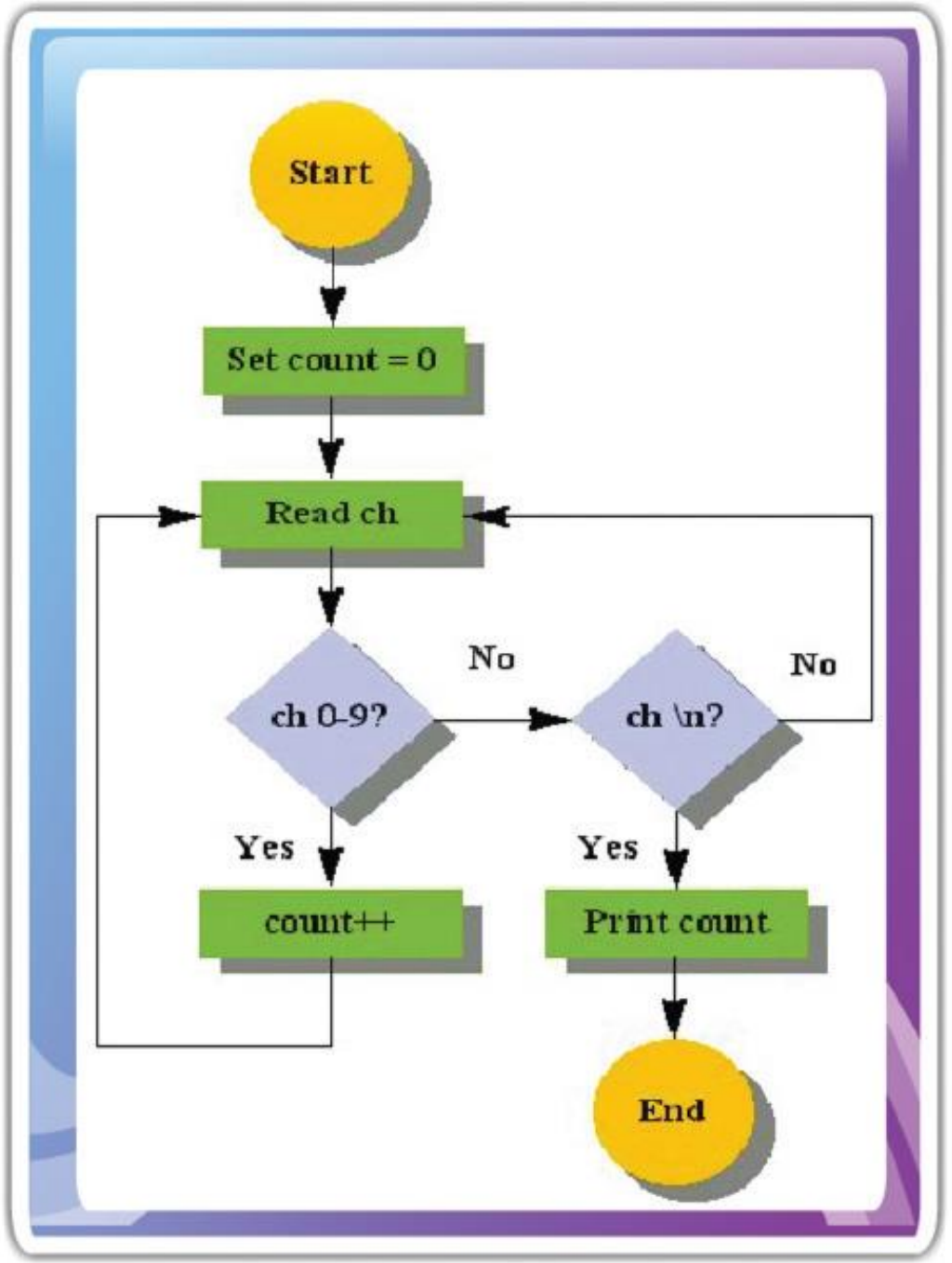
- Behaviour models coordinate a set of steps. These steps are called states, actions, or sub-activities in a unified modelling language (UML).
- Three main categories of behavioral model:



5.1 Control Flow

- Each step takes place only when another step is completed, perhaps taking into account other conditions, but without regard to the availability of inputs.
- Inputs are determined during the transitions between steps, but are not required to start a new step. No restrictions are placed on how inputs are determined.

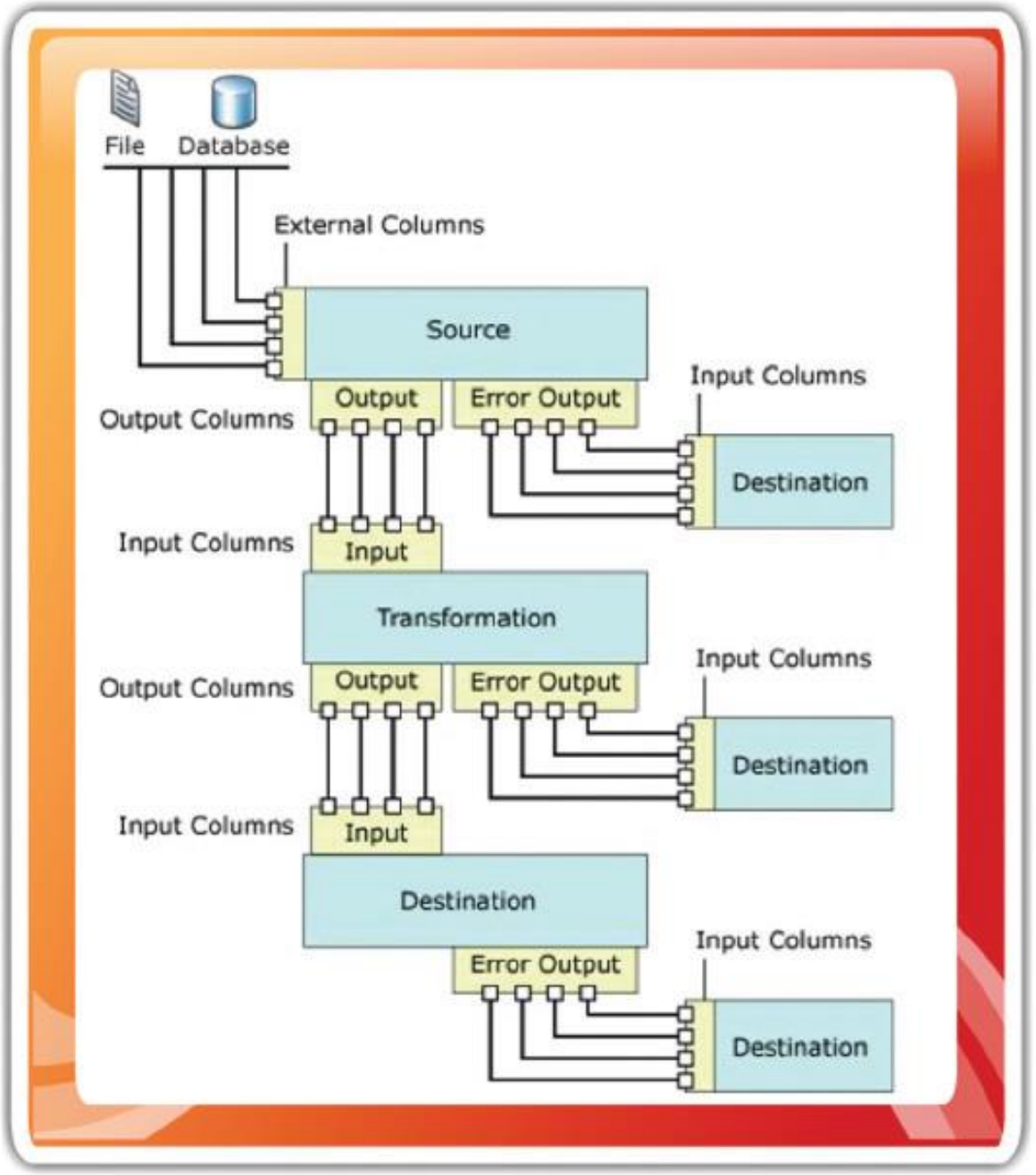
- an example of a control flow



5.2 Data Flow

- Data flow emphasises the calculation of inputs by requiring the outputs of one step to be explicitly linked to inputs of the next step.
- It takes one step when another step provides its inputs. Inputs are being passed directly from outputs from other steps.

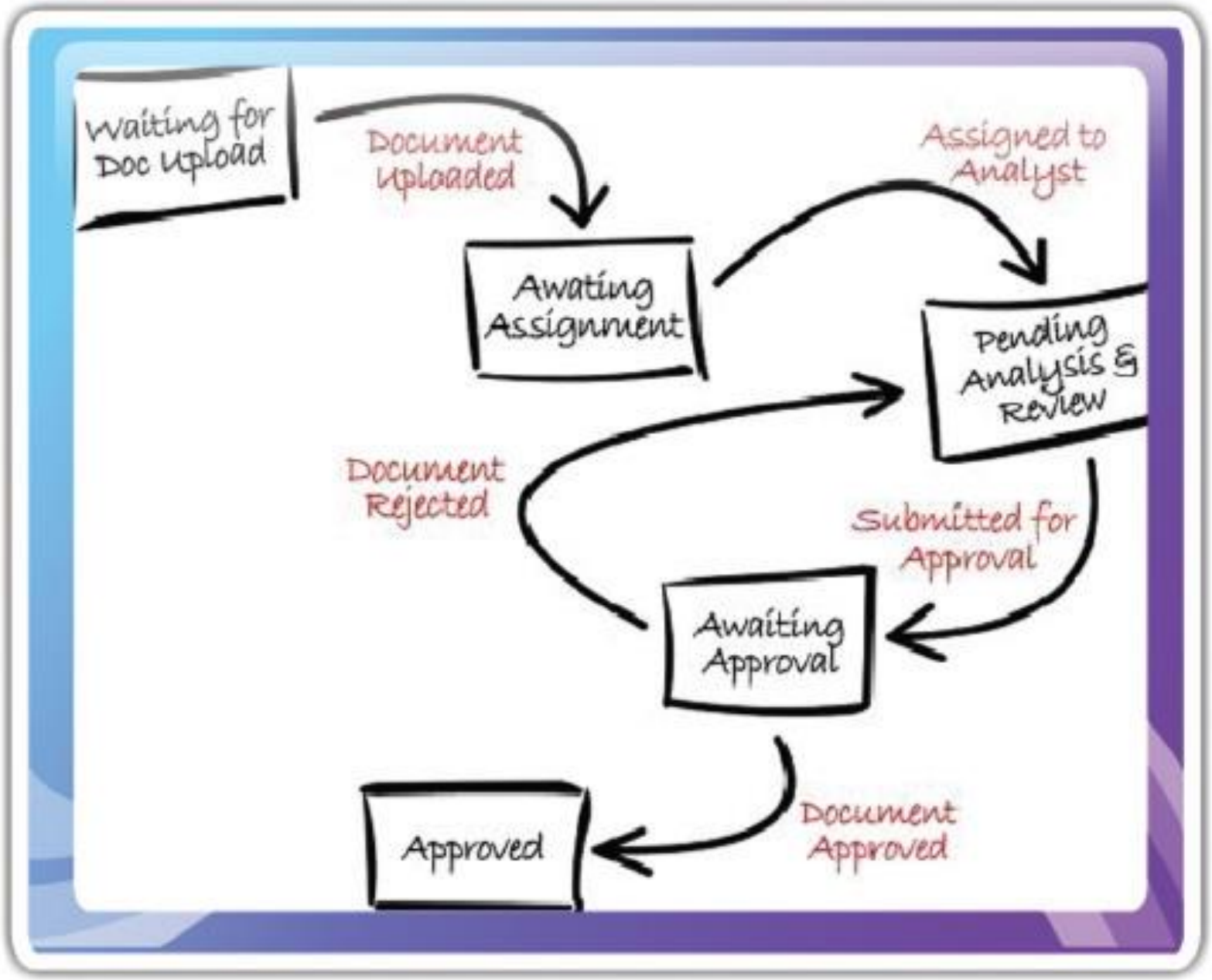
- an example of a data flow.



5.3 State machine

- Each step is based on events that occur in the environment of the behaviour being performed.
- The inputs to each step are calculated as part of the original step itself.
- State machines emphasise response to external process by requiring that each step start only when certain events happen in the environment of the system.

- an example of a state machine.



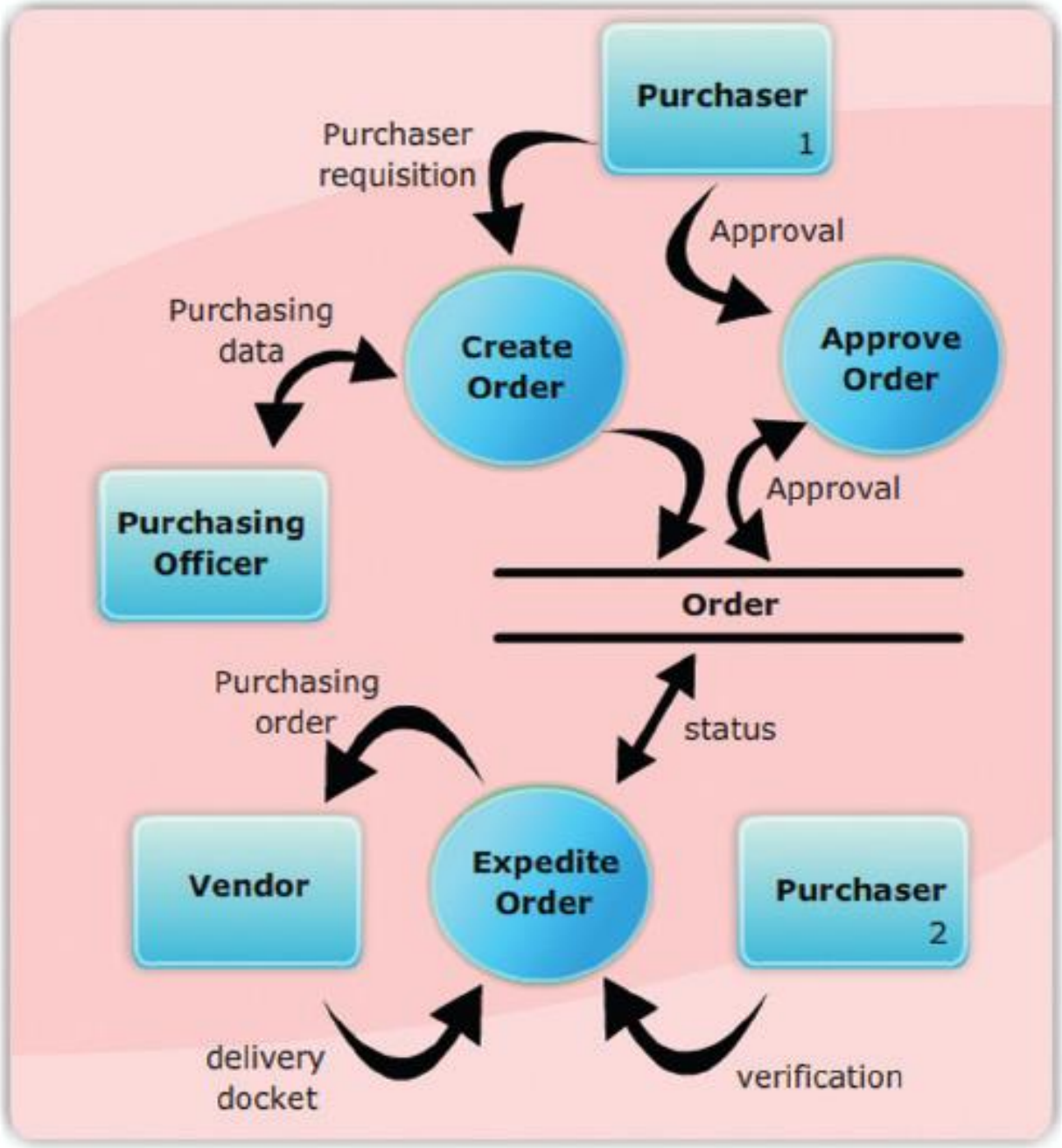
6.0 Functional Modelling

Functional modelling or functional requirements capture the intended behaviour of the system.

This behaviour may be expressed as services, tasks or functions the system has.

In product development, it is useful to distinguish between the baseline functionality necessary for any system to compete in that product domain, and features that differentiate the system from the competitors' products.

- an example of a functional modelling that has the following functions: create an order, approve an order and expedite an order.



6.1 Decision Tables

- A decision table is a business rule that consists of conditions, represented in a row and column headings.
- Actions are represented as the intersection points of the conditional cases in the table.
- Decision tables are best suited for business rules that have many conditions.
- Adding another condition is done by adding another row or column.

- Sample of a decision table

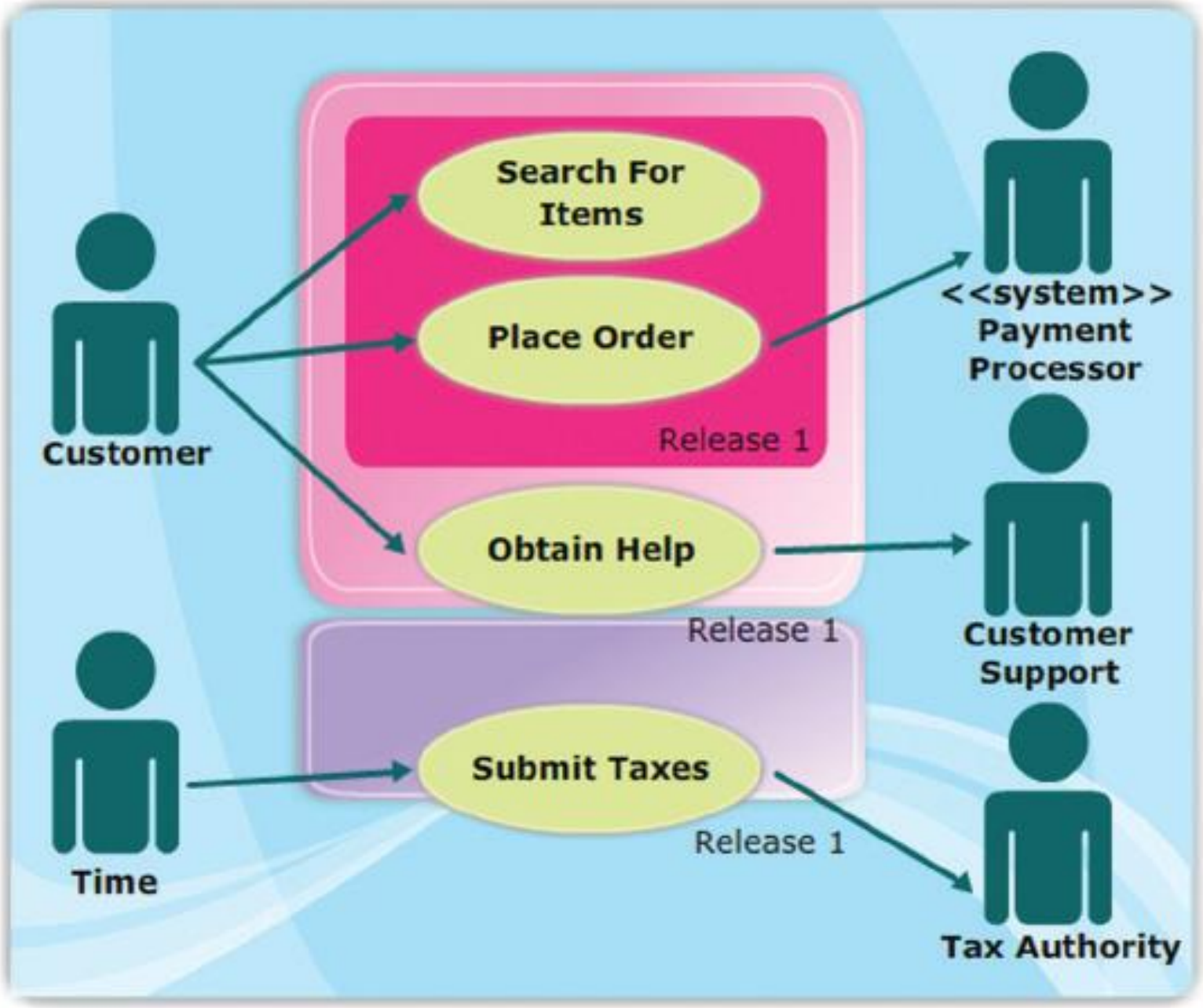
	1	2	3	4	5	6	7	8
Age > 21	Y	Y	Y	Y	N	N	N	N
Sex	1	2	3	4	5	6	7	8
Weight > 150	1	2	3	4	5	6	7	8
Medication 1	x				x			x
Medication 2		x			x			
Medication 3			x			x		x
No Medication				x			x	

- Like the if/then rule set, the decision table is driven by the interaction of conditions and actions.
- The main difference is that in a decision table, the action is decided by more than one condition, and more than one action can be associated with each set of the conditions.
- If the conditions are met, then the corresponding actions are performed.

6.2 Use Cases

- A use case defines a goal-oriented set of interactions between external actors and the system.
- Actors are people outside the system that interact with the system.
- An actor may be a class of users, roles each user can play, or other systems.

- illustrates a use case is initiated by a user with a particular goal in mind, and completes successfully when that goal is satisfied.



7.0 State-Based Modelling