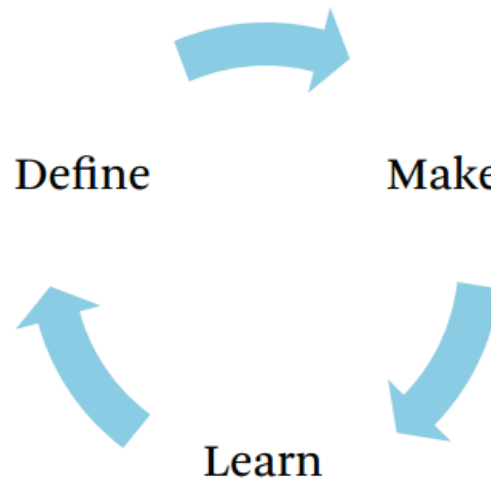




# Iterative Design of Virtual Reality Systems

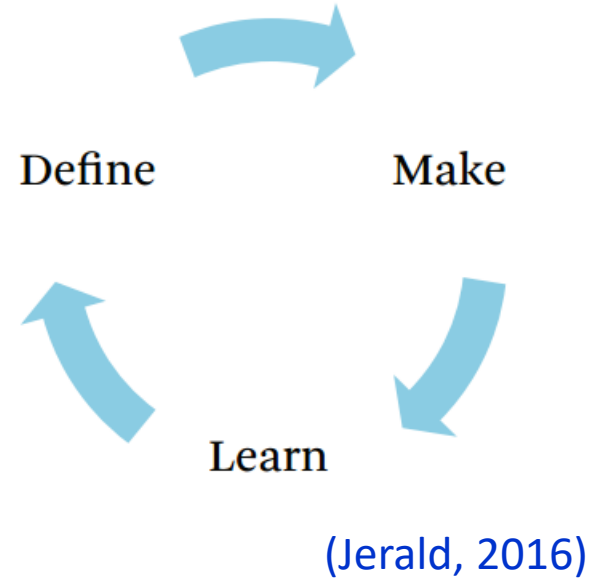


(Jerald, 2016)

# Iterative Design

Iterative design consists of :

- defining the project,
- Building prototypes,
- learning from users,
- and **continually improving upon previous ideas**



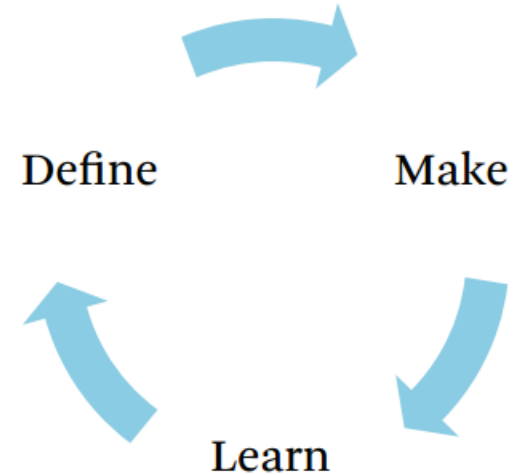
**Careful consideration of these stages is critical to optimizing user experience and comfort**

# Three stages: three questions

Define - “What do we make?”

Make - “How do we make it?”

Learn - “What works and what does not work?”



(Jerald, 2016)

**“Design entails the entire creation ... from information gathering and goal setting to delivery and product enhancement/support” (Jerald, 2016)**

# Building VR (and AR) systems and experiences

**New development platforms allow creating VR experiences very quickly**

- However:
  - Integrating advanced/innovative features and behaviors,
  - understanding software architecture,
  - and writing clean code

Take time and are essential for creating efficient and maintainable code

**Anything beyond the basics Implies good programming skills!**

# Design of VR (and AR) systems and experiences

**It is impossible to get the perfect system or experience on the first attempt!**

- VR design is very different than 2D desktop or mobile application design
- There is not nearly as much known or standardized about VR
- VR design must be **iteratively** created based off of **continual redesign**, prototyping, and **feedback from real users**

**Iterative design concepts help move toward ideal experiences as quickly and effectively as possible**

# Human-Centered Design

The measure of progress for a VR project is the user experience:

**improved experience → progress made**

Traditional software measures are very different ...

**“A brilliantly engineered solution is irrelevant if it doesn’t work for the human”!**  
(Jerald, 2016)

**Feedback from real users is critical; a UX issue main ruin the entire project**

- If it is not clear what specific processes to start with, pick some and go
- **Iterate upon the processes**, improving upon them as you use and learn
- Do not become attached to the processes, be willing to throw them out if and when they are not working

- Is not just about optimizing a set of algorithms
- **It is more imperative to decide on the right things to design** instead of perfectly designing the wrong things
- We can't afford to wait until the end of the project to see if everything works:
  - It won't!
- **Learning, change, and innovation are much less expensive at the beginning**

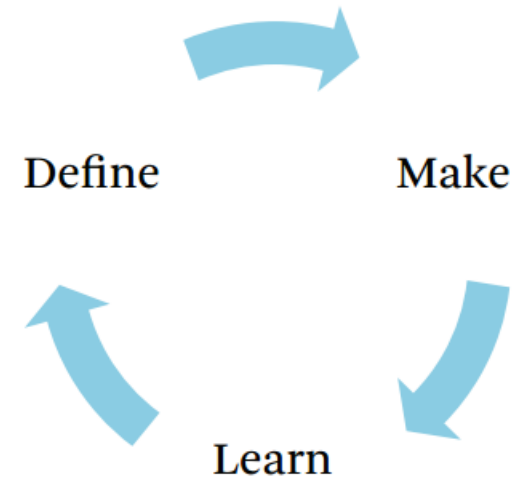


# Iterative Design

## Three stages:

- **Define** – “What do we make?” and Why?
- **Make** – “How do we make it?”
- **Learn from users** - “What works and not works?”

**continually improving upon previous ideas**

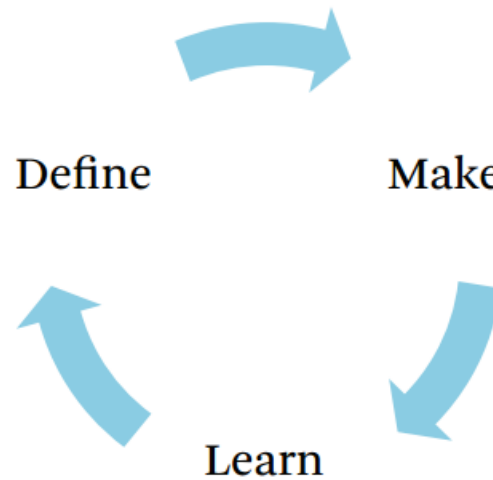


(Jerald, 2016)

## Human-centered approach!!

**Careful consideration of these stages is critical to optimizing user experience and comfort**

# Define Stage



(Jerald, 2016)

# I-Define Stage

- Is the **beginning, but continues to the end** ... (you may return to it)
- Should be described using the **user's point of view** and be understandable by anyone
- Avoid “analysis paralysis”
- **Important questions: What and Why?**

<https://www.interaction-design.org/literature/article/introduction-to-the-essential-ideation-techniques-which-are-the-heart-of-design-thinking>

# Define Stage

- **Vision**
- **Questions**
- Assessment and Feasibility
- High-Level Design Considerations
- **Objectives**
- Key players
- Time and costs
- Risks
- Assumptions
- Project Constraints
- **Personas**
- **User stories**
- **Story boards**
- **Scope**
- **Requirements (quality, functional, ...)**

Not all the projects will use all these concepts...

## I.1 Vision

As an engineering achievement, must be started by someone ...

What is the overall vision?

What is trying to be achieved?

Why is it being done?

With unlimited resources, what could be achieved?

## I.2 Questions

Why? ...

Example: consumers do not want to buy HMDs. They might want to effectively train for specific tasks, or bring more customers into their place of business ...

**Talk with domain experts and understand the context!!**

Some questions to understand context:

What is the inspiration/motivation?

Who are the decision makers?

Who are the principal stakeholders and intended beneficiaries?

Where will the system be deployed?

Is the VR project part of a larger vision?

Does the project play a minor or major role in the organization?

What impact will it have on the organization? ...

## I-5 Objectives

- Objectives are **high-level formalized goals** and expected outcomes **focus on benefits and business outcomes over features**
- Different than requirements (which are more specific and technical e.g., latency and tracking ...)
- Quality objectives: SMART:  
Specific, Measurable, Attainable, Relevant, and Time-bound

“The VR training system will be deployed on YY/MM/DD, and will result in a xx% increase in productivity as defined in Section X after N sessions of training.”

## I.11 Personas


Are **models of the people who will be using** the VR application

Personas help targeting specific users making the design easier

Help to prevent the design from being driven by design/engineering convenience

Personas **should be validated** in later stages

Representative users can be targeted to collect feedback for the Learn Stage


 <p>Name</p>	<ul style="list-style-type: none"><li>• Job</li><li>• Experience</li><li>• Activities</li><li>• Attitude</li><li>• Competencies</li><li>• Age</li></ul>
<ul style="list-style-type: none"><li>• Problems</li><li>• Pain points</li><li>• Needs</li><li>• Concerns</li><li>• Fears</li><li>• Desires</li></ul>	<ul style="list-style-type: none"><li>• Knowledge of VR</li><li>• Dream VR system</li><li>• Vision of VR</li><li>• VR hardware access</li><li>• Budget for VR</li><li>• Activities that fit VR</li></ul>



## Describe 3–4 characters that represent the range of your targeted users

Sketch and name

Basic description of the person

 <p>Name</p>	<ul style="list-style-type: none"><li>• Job</li><li>• Experience</li><li>• Activities</li><li>• Attitude</li><li>• Competencies</li><li>• Age</li></ul>
<ul style="list-style-type: none"><li>• Problems</li><li>• Pain points</li><li>• Needs</li><li>• Concerns</li><li>• Fears</li><li>• Desires</li></ul>	<ul style="list-style-type: none"><li>• Knowledge of VR</li><li>• Dream VR system</li><li>• Vision of VR</li><li>• VR hardware access</li><li>• Budget for VR</li><li>• Activities that fit VR</li></ul>

Challenges the person has

Relation to VR

If personas are especially important (e.g., for therapy applications), then data should be collected with interviews and/or questionnaires

<https://www.interaction-design.org/literature/article/personas-why-and-how-you-should-use-them>

## I.12 User Stories

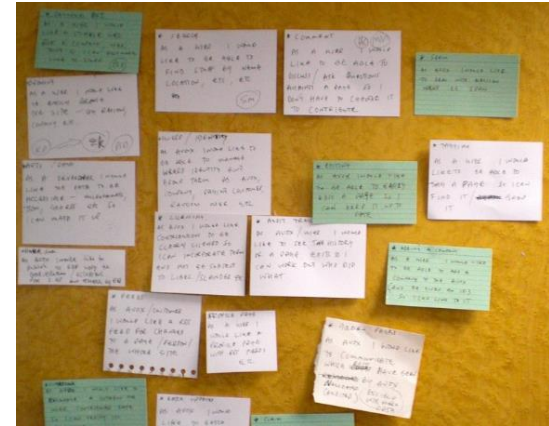
Emerged from agile development methods, are **short concepts or descriptions of features customers would like to see**

Should:

**not go into too much detail**

be written from the **user's point of view**

be written with the **client team members**



**“As a <type of user> I want <some goal> so that <some reason>.”**

Should be written with the **minimum amount of detail** necessary to fully **encapsulate the value that the feature** is meant to deliver

<https://manifesto.co.uk/how-much-detail-should-a-user-story-have/>

<https://www.interaction-design.org/literature/topics/user-stories>

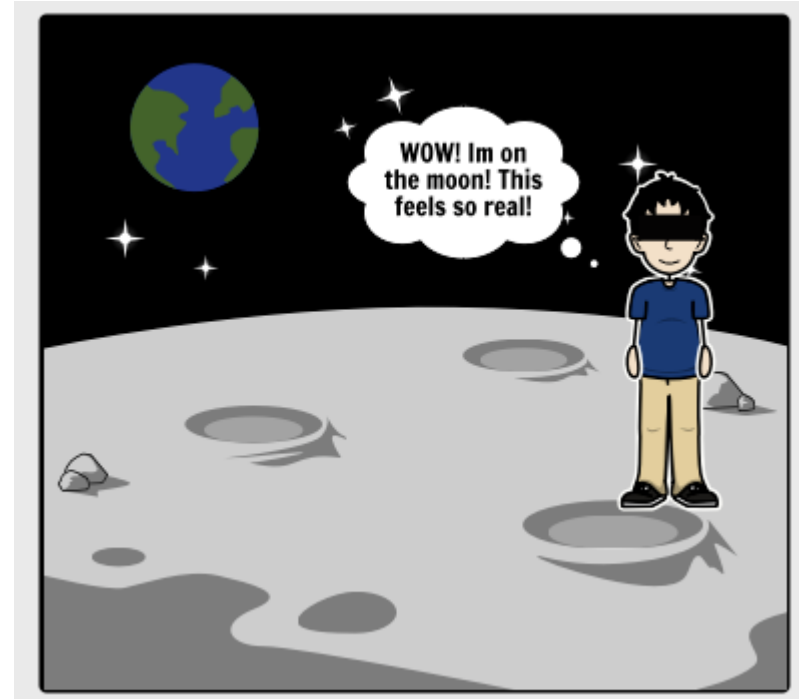
<https://www.interaction-design.org/literature/article/user-stories-capturing-the-user-s-perspective-quickly-and-simply>

## I.13 Storyboards

Are early visual forms of an experience

Particularly useful for VR

The user can be shown directly interacting with objects



<https://www.storyboardthat.com/storyboard-creator>

## I.14 Scope

Explicitly stating what will not be done can be as helpful as what will be done

Set clear expectations and allow the team to focus on what is important

**Items defined to be in scope are nonnegotiable and must be completed**

(Jerald, 2016)

### In Scope

Bimanual tracking

Ray selection

Embodied avatar

Hand-held panel

### Out of Scope

Tracking for standing and walking

Networked environment

Voice recognition

Fine-tuning of colors

### Unresolved

Travel technique

What widgets/tools to include on the panel

## I.15 Requirements

Statements that **convey the expectations** of the client and/or other key players (e.g. descriptions of features, capabilities, and quality)

Each requirement is a single **thing the application should do**

Requirements **come from other parts of the Define Stage**, such as assumptions, constraints, and user stories.

The client and/or **other key players should be actively involved** in its definition

Requirements **do not specify implementation**

**“Any attempt to formulate all possible requirements at the start of a project will fail and would cause considerable delays”** <https://agilemanifesto.org/>

# I.15 Requirements

**Quality  
(non-functional)**

**System (accuracy, precision, reliability, latency...)  
Task Performance  
Usability**

**Functional**     ...

**Universal VR**

**End to end delay  
Frame rate  
Tracking reliability  
Input devices reliability  
View-point motion**

- **Quality requirements** (aka **non-functional** requirements) define the overall qualities or attributes of a system or application:
  - **System requirements** (independent of the user: accuracy, precision, reliability, latency, ...)
  - **Task performance requirements** (time to completion, performance accuracy, performance precision, and training transfer)
  - **Usability** (ease of learning, ease of use, and comfort)
- **Functional Requirements** specify what some part of the system does or what the user can do and include a set of inputs, behaviors, and outputs

## System Requirements - definitions

- **Accuracy** - quality of being correct (i.e., closeness to the truth).  
A system with consistent systematic error has bias and low accuracy
- **Precision** - reproducibility and repeatability of getting the same result.  
A tracked tool that shakes or jitters due to bad tracking has low precision
- **Reliability** - extent to which some part of the system works consistently.  
It is often measured as a hit rate or failure rate.
- **Latency** - the time a system takes to respond to a user's action
- **Rendering time** - amount of time it takes for the system to render a single frame.



- **Universal VR Requirements**

The following should be considered early for all fully immersive VR applications:

- Minimum frame rate = HMD refresh rate (~90–120 Hz)
- 
- Head tracking reliability
- Input devices reliability > 99.9%
- Any camera/viewpoint motion not controlled by the user will not contain accelerations for periods longer than one second

During development the scene should fade out if some cannot be maintained to make developers aware of the failure to meet these requirements

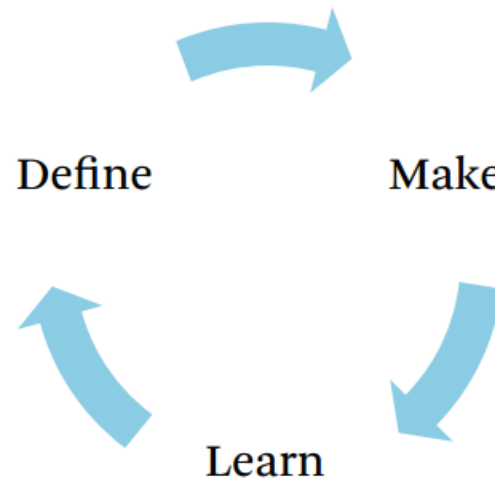
# Practical Assignment Define Stage (simplified version)

- Considering the selected topic, think about:
  - Questions to establish context (Why, what, who, where, ...)
  - Objectives of the project
  - Project constraints

## **Try to talk with subject-matter experts!**

- Define 1 or 2 personas representing the most important types of users
- Create 1 or 2 user stories and story boards
- Define the scope of the project
- From the objectives and personas define requirements (quality, functional...)

# Make Stage



(Jerald, 2016)

## II-Make Stage

- Where the specific design and implementation occurs
- **Where most of the work occurs**
- Implementing VR experiences is similar to creating other interactive S/W in many ways
- **A primary difference is the emphasis on H/W and possible impact on users**
- Often it consists in using existing tools, H/W, and code focusing on gluing the different pieces together so that they work together in harmony

# Make Stage

- Task analysis
- Design specifications
- System Considerations
- ...
- Prototypes
- Final Production
- Delivery

## II.1 Task Analysis

Analysis of **how users accomplish or will accomplish tasks**, including both physical actions and cognitive processes

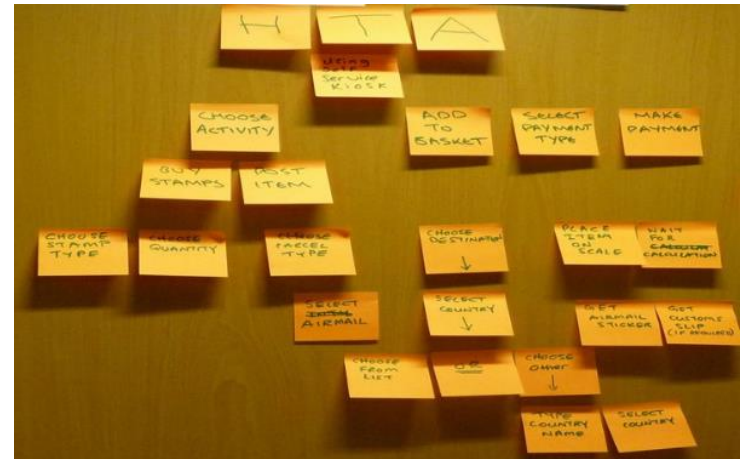
Help **organize a set of tasks** so they can be **easy and clear** to communicated and thought about in a systematic manner

If it complicates, it has failed!

Involves **understanding**:

- the user,
- what the user wants to do,
- how the user currently performs tasks,

Should **start with the actual tasks** performed in the Real World  
(but VR systems may include some magic!)



## How to do Task Analysis

**Representative users** - initially done by the team, should involve observation and interviewing representative users

**Task elicitation** - gathering information in the form of interviews, questionnaires, observation, and documentation

**Organize and Structure** – Hierarchical Task Analysis is a common method

**Review with users** - to verify understanding

**Iteration** - changes to the task analysis will occur

## II.2 Design Specification

Details how an application currently is or will be put together and **how it works**

Between defining and implementation stages

Usually **closely intertwined with prototyping**

Isn't just conducted to satisfy what was created in the Define Stage, **it elicits previously unknown:**

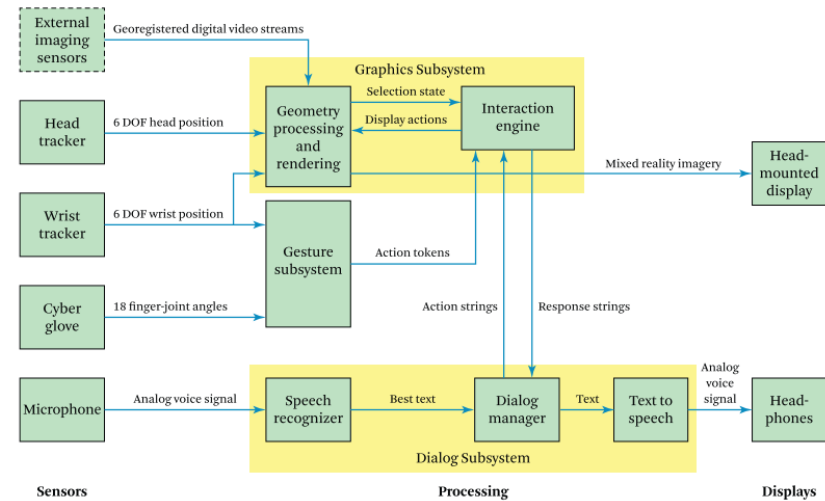
- assumptions,
- constraints,
- requirements,
- details of task analysis



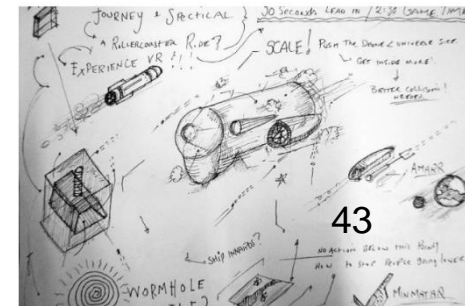
# Most common tools for building VR applications:

- sketches,
- block diagrams,
- use cases,
- classes,
- software design patterns

Bomb
+ container: GameObject # sounds: SoundManager - explosion: GameObject - damageEffect: GameObject - timer: ClockManager - health: Health - damageAmount: int
+ Start() : void + Update() : void + StopTimer() : void + ResumeTimer() : void + DamageBomb(damagePoints: int) : void - PlayWarning(soundID: int) : void - PlayAlarm() : void - Detonate() : void - Explode(size: float) : void



(Jerald, 2016)



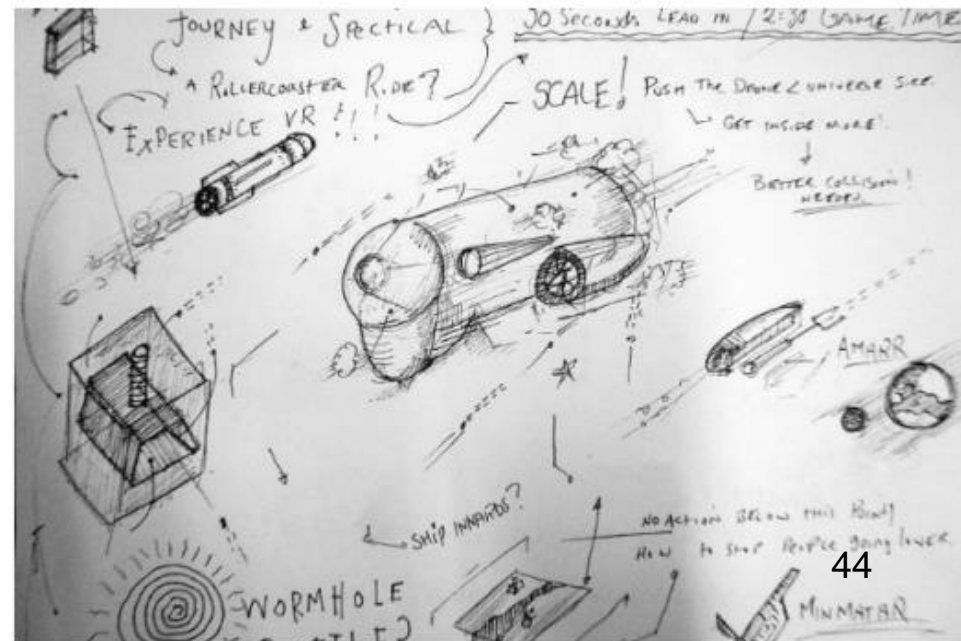
# Sketches

Rapid freehand drawings for preliminary exploration of ideas

Should be:

- Quick and timely
- Inexpensive and disposable
- Plentiful
- Minimal detail
- Ambiguity

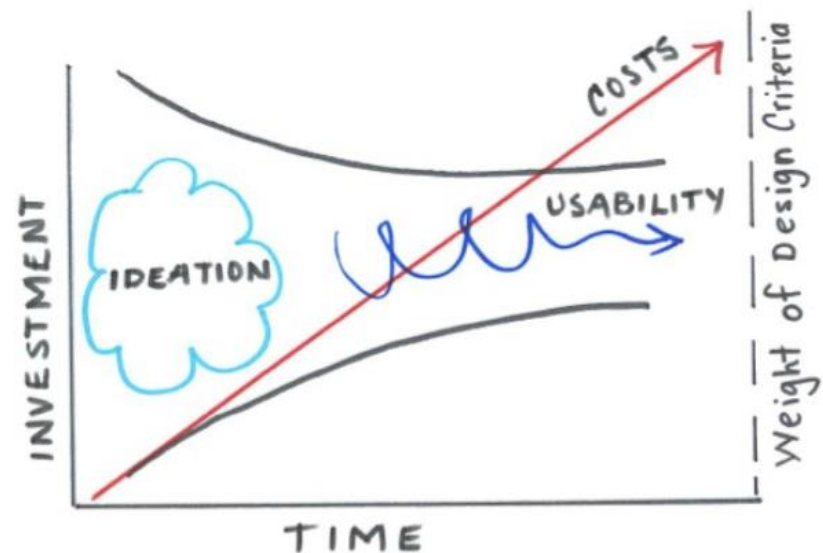
(Jerald, 2016)



<b>Sketch</b>	<b>vs</b>	<b>Prototype</b>
Provocative		Didactic
Suggest		Describe
Explore		Refine
Question		Answer
Propose		Test
Provoke		Resolve
Tentative		Specific
Noncommittal		Depiction

Sketches and prototypes are complementary and have different uses in the design funnel

<https://www.interaction-design.org/literature/article/etch-a-sketch-how-to-use-sketching-in-user-experience-design>





## Use cases

Sets of steps that **help to define interactions between the user and the system** to achieve a goal

Help developers:

- identify, clarify,
- organize interactions making it easier to implement

As interaction complexity grows, the need for explicit use cases increases

May start from user stories, but are **very different**:

- more **formal and detailed**,
- enable developers to get **closer to implementation**

## Use cases vs User stories

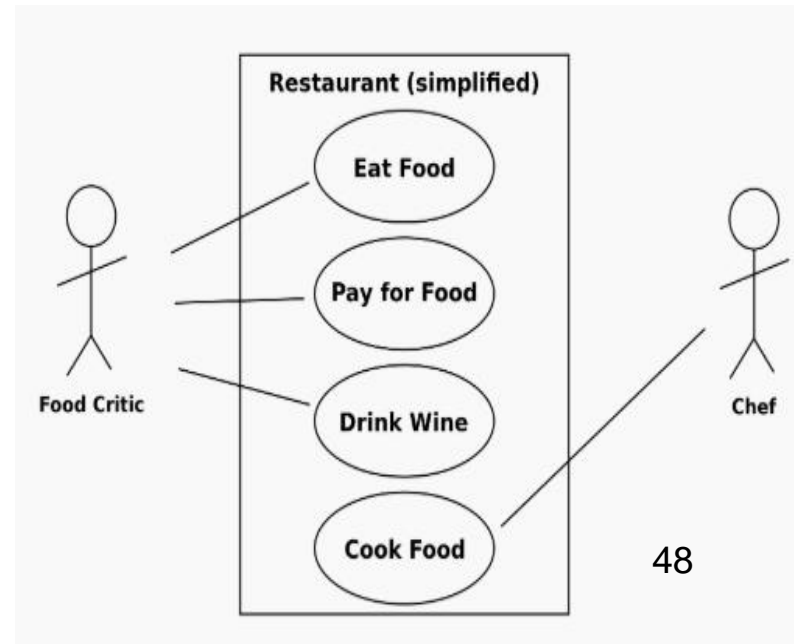
UML use cases are visual diagrams that capture requirements and actors

Use case and a user story serve different functions

Use cases address how a requirement will be handled

User stories simply capture the requirement

<https://www.interaction-design.org/literature/article/user-stories-capturing-the-user-s-perspective-quickly-and-simply>



# Classes

Templates for sets of data and methods

- Data corresponds to nouns and properties  
from information collected in previous steps
- Methods correspond to verbs and behaviors

Can be based on information from the previous steps organizing it into common structures and functionality into themes that can be implemented as classes.

It helps understand how the design will be implemented in code as the design specification evolves

# Classes and objects

A class exists by itself without having to be compiled into an executable program

A program object is a specific instantiation of a class

## Example of a program object:

Perceivable virtual object in the virtual environment

## Real-world analogy of a class:

A set of blueprints for a house

A blueprint is like a class that describes the house

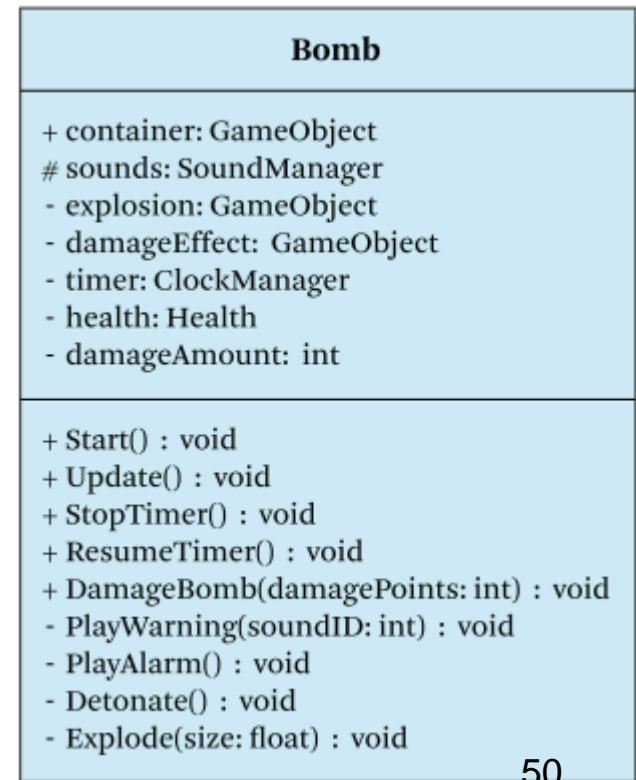
but is not a house itself

Construction workers instantiate the blueprints

into a real house

Blueprints can be used to build many houses

## Class Diagram





# Software Design Patterns

General reusable conceptual solutions used to solve commonly occurring software architecture problems

Described from a system architect's and programmer's point of view

Speed up development by using tested, proven concepts that have been useful for other developers

Provide a common language to communicate with other developers

# Software Design Patterns

Some common patterns for VR:

- factories to instantiate objects (e.g., for the user to create many similar objects);
- adapters to connect libraries (create a single way to support different HMDs)

**Example of a well known S/W pattern in interactive systems**

(Sommerville, 2016)

Name	MVC (Model-View-Controller)
Description	Separates presentation and interaction from the system data. The system is structured into three logical components that interact with each other. The Model component manages the system data and associated operations on that data. The View component defines and manages how the data is presented to the user. The Controller component manages user interaction (e.g., key presses, mouse clicks, etc.) and passes these interactions to the View and the Model. See Figure 6.5.
Example	Figure 6.6 shows the architecture of a web-based application system organized using the MVC pattern.
When used	Used when there are multiple ways to view and interact with data. Also used when the future requirements for interaction and presentation of data are unknown.
Advantages	Allows the data to change independently of its representation and vice versa. Supports presentation of the same data in different ways, with changes made in one representation shown in all of them.
Disadvantages	May involve additional code and code complexity when the data model and interactions are simple.

## II.3 System Considerations

Considering different trade-offs of a project **improves understanding** of the complex interplay of factors and how they might be implemented

- System Trade-Offs and Making Decisions
- Support of Different Hardware
- Frame Rate and Latency
- Sickness Guidelines
- Calibration

## System Trade-Offs and Making Decisions

Help the team to **choose hardware, interaction techniques, software design,** and where to focus development

### **Decisions depend upon many factors:**

- hardware affordability/availability,
- time available to implement,
- interaction fidelity,
- representative users' susceptibility to sickness, etc.

Most should be made early in the project

Even if a decision was wrong, **it can be changed at an early iteration**

**It may be appropriate to support multiple options, but ...**

It is generally better to start by choosing a single option and optimize for it before adding secondary options, features, and support

Decision	Example Choices
Hand input hardware	None, Leap Motion, Sixense STEM
Number of participants	Single user, multiplayer, massively multiplayer
Viewpoint control pattern(s)	Walking, steering, world-in-miniature, 3D multi-touch, automated
Selection pattern(s)	Hand selection, pointing, image-plane, volume-based
Manipulation pattern(s)	Direct hand manipulation, proxy
Realism	Real-world capture, cartoon world
Vection	None, short periods of self-motion, linear motion only, active, passive
Intensity	Relaxing, heart-thumping
Sensory cues	Visual, auditory, haptics, motion platform
Posture	Sitting, standing, walking around

**Common VR decisions and example choices**

(Jerald, 2016)

## **Support of Different Hardware**

Hardware can have similar or very different characteristics

Supporting different hardware of the same class can enable a wider audience

Example:

Sony PlayStation Move is the same class of input device as the Sixense STEM Controller (6 DoF tracked hand-held controllers)

Do not assume testing on one piece of H/W will apply to all H/W in that class

**Test across the different H/W and optimize as needed**

## **Frame rate and Latency**

**Frame rate should maintain at least the refresh rate of the HMD**

Current H/W makes this relatively easy to achieve

Frame rate should be carefully observed as new assets are added and code complexity increases

Even occasional drops in frame rate can be uncomfortable to users

**Consistent latency is as important as low latency**

## Sickness Guidelines

Developers should **respect the adverse effects** of VR

Waiting for feedback from the Learn Stage can dramatically slow down the iterative process and might require complete redesign/reimplementation

It is as important for programmers to **understand adverse health effects and how to mitigate them** as it is for anyone on the team

Note: read about this in chapter 19 ([Jerald, 2016](#))





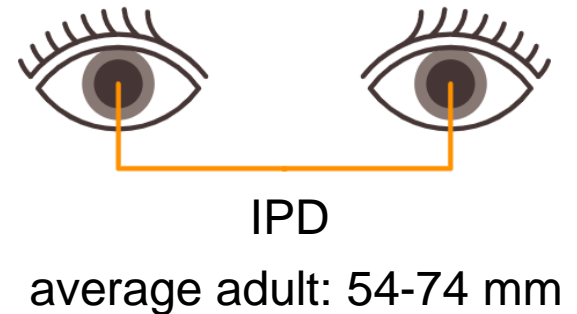
# Calibration

## Proper calibration of the system is essential

Tools should be created for developers and users to easily and quickly calibrate (if not automated)

Examples of calibration include:

- interpupillary distance,
- lens distortion-correction parameters,
- tracker-to-eye offset



**If these settings are not correct motion sickness may result**

## II.4 Prototypes

Are **simplistic implementations of what is trying to be accomplished** without being overly concerned with aesthetics or perfection

Enable the team to **observe and measure what users do**, not just what they say

A minimal prototype is built with the **least amount of work necessary to receive meaningful feedback**

Should allow find problems as soon as possible

**Only build primitive functionality that is essential**

Non-critical details can be added and refined later

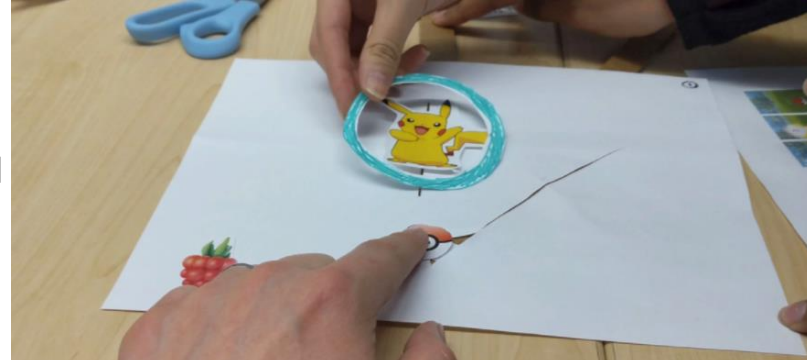
<https://www.sketchbox3d.com/creation>



## Different types of Prototypes

- Real-world prototypes
- Wizard of Oz
- Programmer prototypes
- Team prototypes
- Stakeholder prototypes
- Representative user prototypes
- Marketing prototypes

OBSERVER "COMPUTER" FACILITATOR VR USER



Nebeling, M., & Madier, K. 360proto : Making Interactive Virtual Reality & Augmented Reality Prototypes from Paper. CHI 2019

<https://dl.acm.org/citation.cfm?id=3300826&dl=ACM&coll=DL>

## Forms of Prototypes

**Real-world prototypes** - do not use any digital technology

Team members or users act out roles using physical props or real-world tools

Advantages- can often be created and tested spontaneously

Disadvantages - lack of controlling conditions, difficulty in capturing quantitative data, limitation of feedback to high-level structure/logic

**Wizard of Oz** - basic working VR application, but a human “wizard behind a curtain” controls the response of the system

The wizard typically enters commands after the user verbally states his intention

This sort of prototype can be surprisingly compelling

**Programmer prototypes** - created and evaluated by the programmers who often immerse themselves to quickly check modifications to code

This is how most programmers naturally operate conducting many mini experiments in a day

**Team prototypes** - built for those on the team not directly implementing

Are used for quality assurance testing

Feedback from others is extremely valuable

It can also reduce the belief that your VR experience is the best in the world (but it is not ...)

**Stakeholder prototypes** - semi-polished prototypes taken more seriously and typically focus on the overall experience

Should have a higher level of fidelity closer to the real product, which helps to more fully understand what the final product will be like

**Representative user prototypes** - designed for feedback; should be built and shown to users as soon as possible

Should be focused on collecting data for what is being targeted  
Users with no VR experience are ideal to test for VR sickness

**Marketing prototypes** - built to attract positive attention to the company/project

Should be polished and may get a lot of user feedback in a short time

## II.5 Final product

After the design and features have been finalized the team can focus on **polishing the deliverable**

It is time to **stop exploring possibilities and stop adding more features**

**Save new ideas for a later deliverable**

**Challenge:** when stakeholders experience a solid demo, they get excited and start imagining possibilities and requesting new features!

**But there is a limit of what is possible within the time and budget constraints**  
Adding features involves trade-offs and the new features come at the cost

## II.6 Delivery

**Demos** – provide a way to let others experience the results

**On site installations** - members of the team travel to the customer's site to set up the system and application

Do not assume installation will go smoothly; possible issues:

- electromagnetic interference
- incompatible connections/cables with a client's system
- need to train staff ...

**Continuous online delivery** - up-dates are provided often; involves plenty of opportunity for data collection (e.g. A/B testing)



# Practical Assignment Make Stage (simplified version)

After the Define Stage you should be ready to :

- **Make a simple Hierarchical Task Analysis of the main tasks**
- Consider Design specifications:

**Do some sketches, block diagrams, and use cases representing your ideas**

- System considerations:

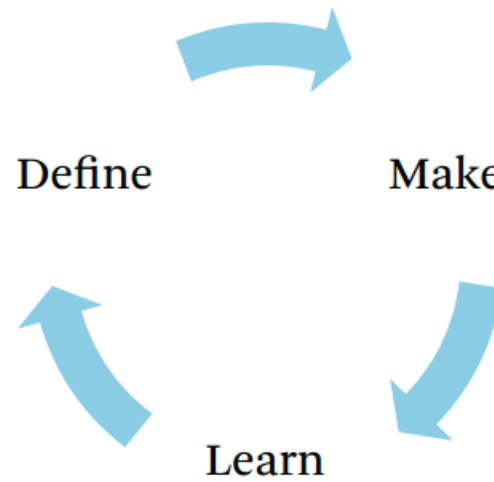
**Ponder H/W, Frame Rate and Latency, Sickness Guidelines, and Calibration**

- Develop Prototypes:

**Develop several prototypes in sequence and according to your problem**

Real world or Wizard of Oz + programmer + Team or User prototypes

# Learn Stage



(Jerald, 2016)

## III-Learn Stage

- Is about continuous discovery about **what works well and what does not**
- **Is more essential for VR than for other technologies**
- Learning may be done in several ways:
  - Code testing and improvement
  - ...
  - Formal experiments (involving many users, along months)
- **Fast feedback, data collection and experimentation** with the help of:
  - VR experts
  - subject-matter experts
  - usability experts ...

# Learn Stage

- Communication and Attitude
- Research Concepts
- Constructivist Approaches
- The Scientific Method
- Data Analysis

## **III.1 Communication and attitude**

**Effective communication is essential for learning:**

- between team members and with users

**The goal is success, but failure is a learning experience**

When communicating with users:

- Do not belittle users or their opinions
- Actively investigate difficulties to determine how to improve
- Thank for noticing problems and encourage looking for problems

## **VR Creators Are Unique Users**

Programmers often use the system in very specific ways without trying all unanticipated actions

May have adapted to sensory incongruences and do not get sick as users do

What works for the programmers probably will not work for everyone else

**Regular communication with users and taking seriously their opinions is a priority**

## III.2 Research concepts

**“Research is a systematic method of gaining new information and a persistent effort to think straight” (Jerald, 2016)**

Important concepts:

- Data Collection
- Reliability
- Validity (face validity, internal validity, construct validity, statistical conclusion validity, external validity, ...)
- Sensitivity
- Constructivist approaches (mini-Retrospectives, demos, interviews, questionnaires, focus groups, expert evaluations, formative usability, evaluation, comparative evaluation, ...)
- After action reviews

## Data Collection

**Enables specifying and comparing the effectiveness of VR aspects**

Often involves:

- automated data
- observation by human raters

Data may be:

- quantitative
- qualitative

Both provide unique insight about a design's strength and weaknesses.

Several types of data should be collected



## Data Collection (cont)

Many measures may be useful:

- time to completion, errors,
- accuracy,
- achievements/accomplishments,
- resources used, space/distance,
- body/tracker motion, latency, breaks-in-presence,
- collisions with walls and other geometry,
- physiological measures,
- preferences, ...

## Reliability

**“The extent to which an experiment, test, or measure consistently yields the same result under similar conditions” (Jerald, 2016)**

Similar results with different users, sessions, times and experimenters suggest consistent characteristics

**Measures are not perfectly reliable;** may be affected by

- stable characteristics (improve reliability)
- unstable characteristics (undermine reliability)

Examples of unstable characteristics:

users': health, fatigue, personality, ...

device/environment's: temperature, precision, errors ...

# Validity

**“Validity is the extent to which a concept, measurement, or conclusion is well founded and corresponds to real application.” (Jerald, 2016)**

A reliable measure may consistently measuring the wrong thing

Is fundamental in research to assure conclusions are valid and apply to a domain

Can be divided in several ways, as:

- face validity,
- construct validity,
- internal validity,
- external validity,
- statistical conclusion validity

## Validity (cont)

**face validity** - impression that a concept, measure, or conclusion seems genuine

**construct validity** - extent to which a measurement truly gauges the conceptual variable

**internal validity** - degree of confidence that a relationship is causal

**external validity** - the degree to which results can be generalized to other cases

**statistical conclusion validity** - degree to which conclusions are statistically correct

## Sensitivity

**“Capability of a measure to adequately discriminate between two things; the ability of an experimental method to accurately detect an effect, when one does exist.” (Jerald, 2016)**

Measures may be reliable and valid but lack sensitivity to be useful:

They should allow distinguishing among levels of the variable

## III.3 Constructivist Approaches

**“construct understanding, meaning, knowledge, and ideas through experience and reflection upon those experiences rather than trying to measure absolute or objective truths about the world. ”** (Jerald, 2016)

Focus more on qualitative data and the context in which it is collected

Several methods that can be useful in VR:

- mini-retrospectives
- demos,
- interviews, questionnaires,
- focus groups,
- expert evaluations,
- formative usability ,
- comparative evaluation

## Mini-Retrospectives

**“Short focused discussions where the team discusses what is going well and what needs improvement” (Jerald, 2016)**

Should be brief but performed frequently and identify areas to improve upon

## Demos

Common form of acquiring feedback from users and great to stay in touch

Provide a goal for the team to work toward and to show progress

Are not the same as collecting data

# Interviews

**“series of open-ended questions asked by a real person ... are more flexible and easier for the user than questionnaires. (Jerald, 2016)**

Result best when immediately after the user experienced the VR application

Very common in usability testing

Guidelines should be created in advance; should:

- Establish empathy
- Be in natural settings
- Be short (<30min)

<https://www.interaction-design.org/literature/topics/user-interviews>





# Questionnaires

**“written questions that participants are asked to respond to in writing or on a computer” (Jerald, 2016)**

Easier to apply and allow more private responses

Very common in usability testing

May include:

- Close-ended questions
- Partially closed-ended questions
- Open-ended questions, ...

<https://www.interaction-design.org/literature/article/useful-survey-questions-for-user-feedback-surveys>



## Focus Groups

**“similar to interviews but occur in group settings ... more efficient ... can stimulate thinking as participants build off of each other’s ideas” (Jerald, 2016)**

In VR participants typically use a prototype demo and give feedback

Extremely useful in the early phases of VR design

<https://www.interaction-design.org/literature/article/how-to-conduct-focus-groups>



## Expert evaluation

**“Systematic approaches conducted by experts that identify usability problems from the user’s perspective with the intent to improve the user experience”**  
(Jerald, 2016)

May be the most efficient method of improving upon the usability of the system

Should involve different types of evaluations in succession



(Jerald, 2016)

## **Expert evaluations (cont)**

Different methods should be used for different aspects of the system and at different phases of the project.

**Expert guidelines-based evaluation** (aka heuristic evaluation) - traditional 2D interface guidelines are not appropriate for VR

**Formative Usability Evaluation** - to improve usability through observation of representative users

**Comparative Evaluation** (aka summative evaluation) - comparison of several systems, applications, methods, or techniques to determine which is more useful

## Expert guidelines-based evaluation

Also known as heuristic evaluation

Identifies potential usability problems by comparing interactions to established guidelines (heuristics)

Traditional 2D interface heuristics are not appropriate for VR

should be done early in the development cycle

<https://www.interaction-design.org/literature/article/heuristic-evaluation-how-to-conduct-a-heuristic-evaluation>



# Formative evaluation

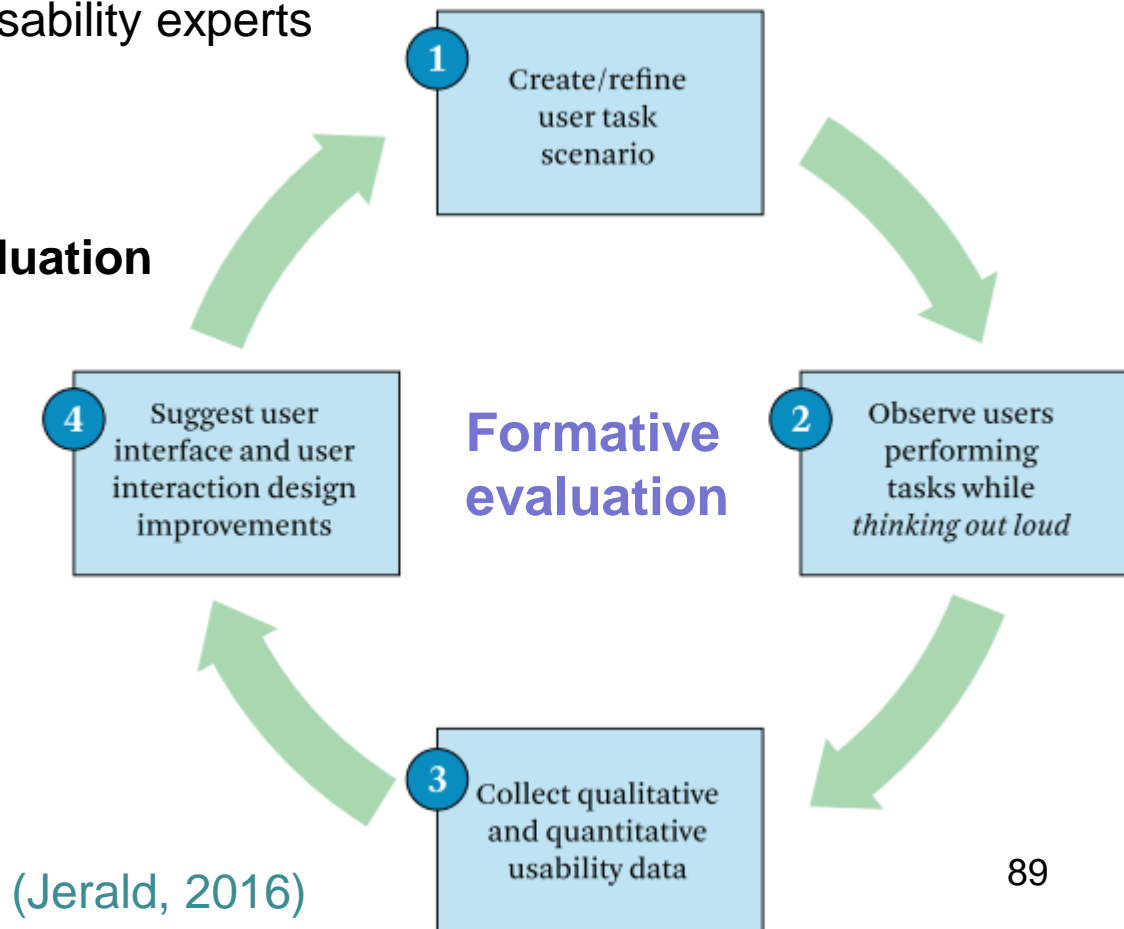
To assess and improve usability through observation of representative users

Should be conducted by VR usability experts

## Comparative/summative evaluation

Involves:

- A set of user tasks
- To collect quantitative data



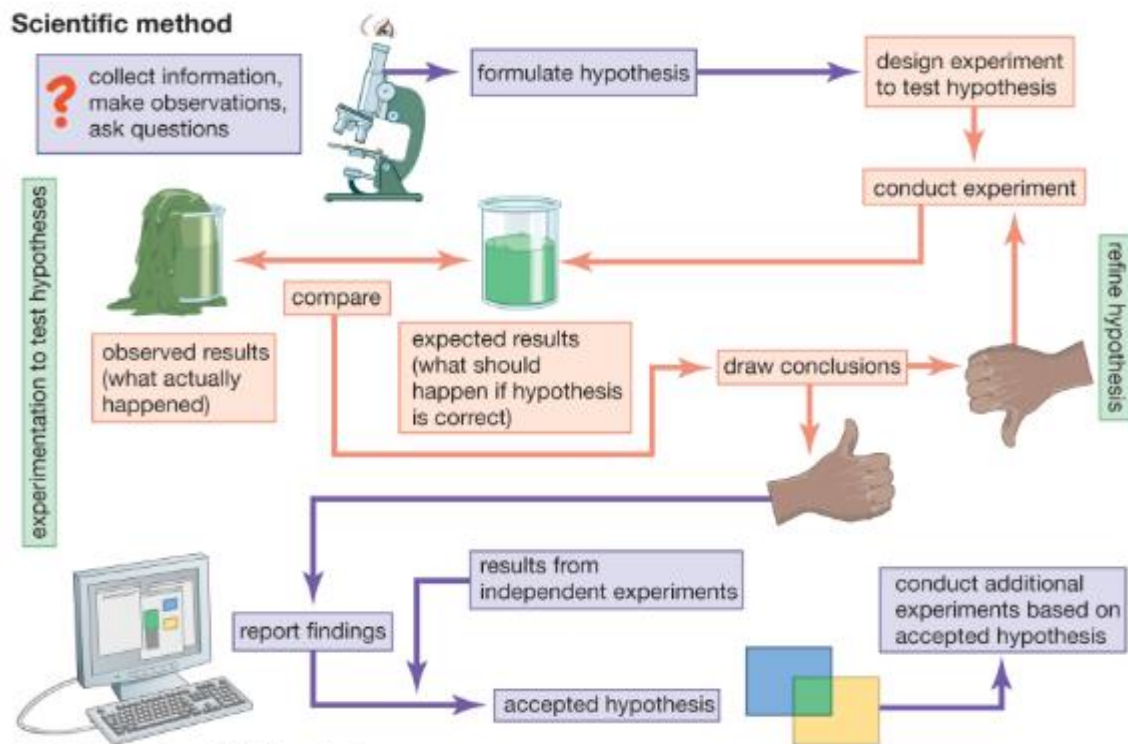
## III.4 The Scientific Method

“an ongoing iterative process based on observation and experimentation. It typically begins with observations that lead to questions ... are then turned into testable predictions called hypotheses. These hypotheses can be tested”  
(Jerald, 2016)

The results of experiments more often lead to more questions than answers

The cycle of predict, test, and refine can be repeated many times

<https://www.britannica.com/science/scientific-method>



# Steps in preparing an experiment

**Problem exploration** - understand what is to be studied; can be done by:

- learning from others
- trying oneself and observing others interacting with existing VR applications
- using constructivist approaches
- using concepts discussed in the Define and Make Stages

**Hypothesis** – i.e. predictive statements about the relationship between two variables that are testable

Example:

“Interaction technique X results in less time to complete task Y than interaction technique Z.” (Jerald, 2016)



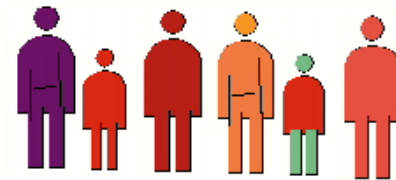
**Variables** - different variables must first be precisely defined

- **Independent or input variables** – what is controlled (e.g. interaction method)
- **Dependent or output variables** – what is measured (e.g. times and errors)
- **Confounding factors or secondary variables** – other factors not controlled that may affect the dependent variable
- Considering threats to internal validity may help find confounding factors

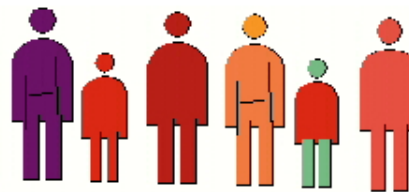
**Experimental Design** – refers to how participants are allocated to the different conditions (or Independent Variable levels) in an experiment:

- **Within subjects or Repeated measures** – every participant experiences each condition
- It requires fewer subjects and has less variability
- Cannot be used when carryover (learning, fatigue, ...) effects are expected to be strong

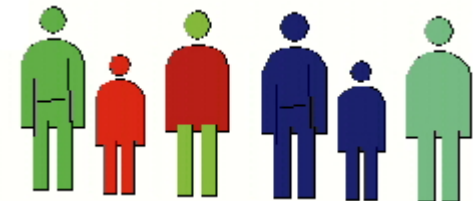
**Condition A + B**  
or  
**Condition B + A**



- **Between subjects or independent measures** – every participant only experiences one condition
- It requires higher number of subjects and has more variability
- Has no carryover effects



**Condition A**



**Condition B**

**Pilot study** – small- scale preliminary experiment that acts as a test run for a more full-scale experiment

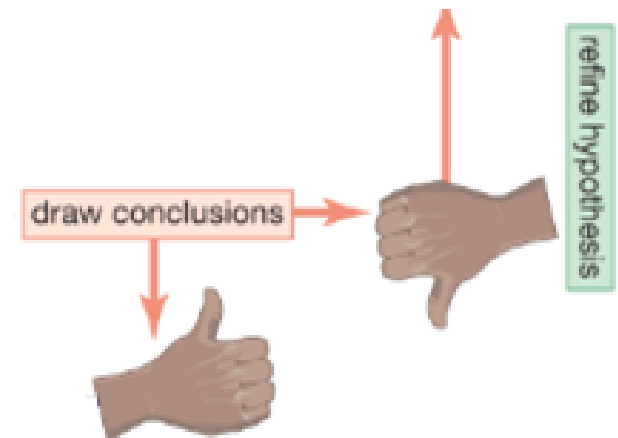
- test the experiment and improve it

**Experiment**– with an adequate number of representative participants and maintaining the conditions

**Data Analysis** – using adequate statistical methods to draw conclusions

Was the Hypothesis confirmed?

**Iterate if necessary !!**



## True Experiments vs. Quasi-Experiments

**True experiments** use random assignment of participants to groups to remove major threats to internal validity

**Quasi-experiments** don't use true random assignment of participants to conditions

Quasi- experiments are more likely to have confounding factors and it is more difficult to claim a cause-effect relationship

But are easier to set up...

Sometimes it is *impossible* to manipulate the independent variable (IV) and assign participants randomly

;E.g. left-handed or right-handed people

Sometimes it is *unethical* to manipulate the IV and assign participants randomly

E.g. sick and healthy people

## Main bibliography

- Jerald, J., *The VR Book: Human-Centered Design for Virtual Reality*, ACM and Morgan & Claypool, 2016
- Sommerville, I., *Software Engineering* (10th ed.). Pearson, 2016
- The Interaction Design Foundation Encyclopedia  
<https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed>