Development of a virtual reality application and general methodological considerations for seeking code for a possible continuation

Thesis for the Degree of Doctor of Philosophy (PhD)

Beatrix Katalin Szabó
Supervisor: Dr. Attila Gilányi

University of Debrecen
Doctoral Council for Natural Sciences and Engineering
Doctoral School of Informatics
Debrecen, 2025

Hereby I declare that I prepared this thesis within the Doctoral Council of Natural Sciences and Engineering, Doctoral School of Informatics, University of Debrecen in order to obtain a PhD Degree in Engineering at Debrecen University.
The results published in the thesis are not reported in any other PhD theses.
Debrecen, ... August 2025.

……..
signature of the candidate

Hereby I confirm that Beatrix Katalin Szabó candidate conducted her studies with my supervision within the Applied Information Technology and its Theoretical Background Programme of the Doctoral School of Informatics between 2018 and 2025. The independent studies and research work of the
candidate significantly contributed to the results published in the thesis.
I also declare that the results published in the thesis are not reported in any other theses.
I support the acceptance of the thesis.

Debrecen, ... August 2025. *…….*
signature of the supervisor

DEVELOPMENT OF A VIRTUAL REALITY APPLICATION AND GENERAL METHODOLOGICAL CONSIDERATIONS FOR SEEKING CODE FOR A POSSIBLE CONTINUATION

Dissertation submitted in partial fulfilment of the requirements for the doctoral (PhD) degree in Informatics

Written by Beatrix Katalin Szabó
electrical engineer (MSc), ergonomics engineer

Prepared in the framework of the
Doctoral School of the University of Debrecen
(Applied Information Technology and its Theoretical Background programme)

Dissertation Supervisor: Dr. Attila Gilányi

The official opponents of the dissertation:

Dr. .....

Dr. .....

Dr. .....

The evaluation committee:

chairperson: Dr. .....

members: Dr. .....

Dr. .....

Dr. .....

Dr. .....

The date of the dissertation defense: ……

Table of Contents

1. Introduction.....1
2. Using a virtual reality headset in the simulation of a control room.....4
2.1. Introduction.....4
2.2. Hardware, operating system, and game engine.....5
2.3. Integrating the headset with the application.....7
2.4. Implementing an autofocus feature.....7
2.5. Headsets: problems and perspectives.....12
2.5.1. Physical characteristics.....12
2.5.2. Resolution and graphical data communication throughput.....12
2.5.3. Vergence-accommodation conflict.....13
2.5.4. Drift.....14
2.5.5. Combining with sensors and providing stimuli for other senses.…15
3. Rigged hand model for the Blender Game Engine.....16
3.1. Introduction.....17
3.2. Leap Motion's representation of the hand.....19
3.3. A step-by-step guide to the rigging.....21
3.4. Interfacing with Leap Motion.....28
3.5. Summary.....31
4. Interaction in an immersive virtual reality application.....33
4.1. Introduction.....34
4.2. NUIs.....35
4.3. Gestures.....38
4.4. Designing gesture-based interaction.....43
4.5. Application: interaction in a virtual control room.....45
4.6. Conclusion and future directions.....48
5. Immersion in virtual reality literature.....50
5.1. Introduction.....51
5.2. Two glimpses into the history of VR and immersion.....52
5.3. Immersion as a transmedial phenomenon.....53
5.4. Immersion in VR literature.....56
5.5. A tertiary literature review on immersion.....66
5.6. Conclusion.....68
6. Web search of software developers — Features and tips.....69
6.1. Introduction.....69

6.2. Features of developers' web searches....70
6.3. Exploratory searches....74
6.4. A few more web search tips for developers....79
6.5. Discussion....81
7. A systematic mapping study on how developers search for source code on the web....82
7.1 Introduction....82
7.2 Background....83
7.2.1 Definition of source code....84
7.2.2 The notion of source code search/seeking....84
7.2.3 Study types....85
7.2.4 Data collection methods and types of collected data....89
7.3 The systematic mapping process....92
7.3.1 Research questions....96
7.3.2 Planning and executing the study....96
7.4 Answers to the research questions....96
7.5 Conclusions....118
Appendix A: Phases of the systematic mapping study....121
Appendix B: Inclusion / exclusion criteria....128
Appendix C: The initial set....132
Appendix D: Primary studies....132
Appendix E: Relevant but redundant studies....133
Appendix F: The used search tools....133
Appendix G: The search queries....135
G.1 The Basic Query....135
G.2 The queries for Google Scholar and Google....136
Appendix H: The exception list for the duplicate eliminator....153
References....154

1. Introduction

Virtual reality (VR) is gaining increasing importance, not only in entertainment and games, but also in education, training and research. One such field of application is the simulation of the operation of nuclear power plants. Experimenting on a virtual model of a nuclear power plant is risk-free: a situation which would be dangerous to try out in real life (such as serious malfunctions jeopardizing nuclear safety) can be tried out without real risk. Physical replica simulators (where a computer simulates the nuclear processes but the control room is physically replicated) have been in use since decades, but they are costly to make and maintain, especially, if the plant to model consists of a multitude of physical control organs. Simulating these in virtual reality is a cost-effective and flexible solution, if the simulator provides a sufficiently realistic experience for its users. The progress in virtual reality techniques makes that experience increasingly possible.

In the first chapters of this dissertation a particular software development project is discussed, a virtual reality model of a nuclear power plant control room, realized in the open source Blender Game Engine (BGE). I continued the development of an already existing application (made by colleagues [114]): I applied a VR headset to it [294] and integrated into it a complex, realistic-looking hand model for the interaction [37]. The main interaction tasks for the project were resolved [262], but the BGE was discontinued while our project was still in progress.

Developing a large, animated virtual reality model for a particular game engine usually requires considerable effort and time. As far as realism is concerned, substantial progress has been made recently in the field of VR. Low- or even medium-resolution graphic models are no longer competitive. However, due to organizational reasons, no decision has been taken yet on using another solution, which, if we want to stay up-to-date, would probably require considerable

investment at least in hardware, as well as a lot of additional development work fulfilling such requirements as the use of high-definition images.

In the course of designing the interaction for the solution with the Blender Game Engine, a methodological issue has come up: the notion of immersion, which is an important element in a virtual reality application. I studied the pertinent literature and after a while it became clear that there exist two incompatible views on immersion in the literature. To clear up this issue, I proposed new terminology to differentiate between the two notions of immersion [295]. I performed two literature reviews on the notion of immersion: a non-systematic review [295], and a tertiary review with systematic methods [297].

Adapting the control room model to a new software and hardware base would have involved fundamental reworking of the data transfer mechanism between the simulator and the VR graphics), and it would probably have exceeded the deadline for the dissertation. In preparation for a possible continuation of the work, I studied the methodologies for source code seeking in general, to ease the development of a future new solution for our VR task, if a decision is taken to continue the development work in the realm of open source software. Thus, I advanced from the particular towards the general.

The rationale behind this decision was that source code is rarely developed from scratch, using solutions worked out by other developers can considerably speed up a project, therefore it can be highly advantageous for a developer to adapt solutions created by other developers, but, first of all, he/she should be able to find the solution.

The rest of my thesis sums up the results of my investigations into code search.

I wanted to get nearer to the problem of how to find code online in general, how developers perform such code searches on the web, and what the practices of experienced developers are. Methodological aspects, plus results of ~~a~~ two literature

reviews with practical considerations and tips for developers' web search for aiding their development work are presented.

The chapters below contain the essence of the practical, virtual reality-oriented development work that I have performed and published, and my more theoretical activity in the form of literature reviews on immersion and code search. The development work came to a halt with the discontinuation of the Blender Game Engine in which the development had been conducted, but there has been no decision on the choice of a new hardware and software platform yet.

By reviewing in the literature the manifold and to some extent incompatible definitions of a central concept in virtual reality, namely, immersion, I wanted to clear up some methodological issues. I also wished to ease a tiny bit the development work of current and future developers with theoretical considerations and practical tips, by assessing through literature reviews how developers search for code and what advice could be given to them. This could be useful in the possible continuation of our virtual reality project as well, and also for the developer community in general, as code search is a widespread and often rewarding activity of developers who can spare a lot of time and effort by adopting solutions worked out by other developers, provided that they are able to find those solutions.

The pertinent bibliography is discussed separately in each chapter of the thesis.

This thesis uses first person singular (“I”), and I am the first author of all of the included publications, but my supervisor Dr. Attila Gilányi has contributed to some papers that I have written about web search of developers and immersion. I am immensely grateful to him for providing help, knowledge, encouragement and inspiration all throughout my PhD work, not to speak of submitting one of the completed papers himself when I fell ill and was incapacitated.

2. Using a virtual reality headset in the simulation of a control room[1]

2.1. Introduction

Simulators for operator training have long been in use for nuclear power plants. These are traditionally full-scope simulators with an expensive physical replica of the control room. Recent advances in virtual reality headsets provide affordable means for presenting a stereoscopic view of a virtual model of the control room. While the commercially available headsets are still not perfect, they offer more realism than flat screens, showing a stereoscopic view. This chapter provides some details on how a virtual reality headset has been utilized for viewing the virtual control room of the Paks Nuclear Power Plant, modeled with the Blender Game Engine, with an added autofocus feature based on a pragmatic method. Some aspects of the solution are outlined, and, partially based on the experiences gained in the project, current problems and future trends of virtual reality headsets are discussed.

The full-scope replica simulator of the Paks Nuclear Power Plant, mainly for operator training, has been in use since 1988 (with several refurbishments). It contains a physical replica of the main control room which is a traditional control room containing a lot of pushbuttons and switches. Three operators and their supervisor work in the huge control room which has panels in multiple rows providing information and control, as can be seen in Fig. 2.1. A virtual 3D model of this control room was created a few years ago [114] and connected to

[1] This chapter is based on the conference paper [294] by the author. ©2022 IEEE.

the software system of the simulator. It uses the stand-alone version of the Blender Game Engine. The first implementation used flat screens. Development continued with the integration of a virtual reality headset. The user, wearing the headset, sits at a table. The goal was to create a demo application using affordable hardware.

2.2. Hardware, operating system, and game engine

At the start of the project, the two most promising and affordable head-mounted displays available were the Oculus Rift CV1 and the HTC Vive. The Oculus Rift CV1 was chosen, which has 1080×1200 resolution per eye, a 90 Hz refresh rate, and 110° field of view. The separation of the lenses is adjustable by a dial at the bottom of the device, in order to accommodate a wide range of interpupillary distances. The Rift has integrated headphones and rotational and positional tracking. In this project only its rotational tracking (gyroscope) is used. The touchless Leap Motion hand movement detector controls the navigation in the control room and manipulates the switches and buttons [262].

Our application has been created under 64-bit Windows 10, in a PC with Intel i7 processor and NVIDIA GeForce GTX 1080 graphics card.

Fig. 2.1 The real control room (source: [114])

The virtual control room has been realized in the Blender Game Engine, a component of the Blender (https://www.blender.org/) suite, a free and open-source 3D creation suite (cf. [45], [123]). The version used in the project is 2.76b. The Blender Game Engine is no longer developed. It would be possible to switch to another game engine. Right now the most widely used game engine for similar applications is Unity, which is able to import models made with Blender, but the numerous scripts would have to be reworked.

Similar applications modeling control rooms of nuclear power plants in virtual reality, found in the literature (all dated after this project started), are [39] [162] [163] [179] [275].

The application described in [179] uses the Unity game engine with a HTC Vive headset with Tobii Pro VR Integration eye tracking, and the HTC Vive controller. The other publications are all related to the Finnish Loviisa nuclear power plant and the APROS process simulation software. [39] reports using

Unity, the Oculus Rift headset with rotational and positional tracking, Oculus remote controller, keyboard, and mouse. [163] and [162] list Unity, a Varjo headset and Valve Index controllers. [275] mentions Unity, the Oculus Rift and Varjo VR-1 headsets.

2.3. Integrating the headset with the application

For the open-source BGE, the open-source package OpenHMD (http://www.openhmd.net/) has been selected for the project. OpenHMD is able to drive various VR headsets, including the Oculus Rift, the picture appears in the headset in extended mode (treating the screen of the headset as a monitor) and the package is able to handle the gyroscope input of the headset as well.

For O*penHMD*, the open-source *HIDAPI* package (*https://github.com/signal11/hidapi*) was used.

2.4. Implementing an autofocus feature

When the Oculus Rift was integrated into the project and tests were made with it, it became clear after a while that a fixed-focus camera is inadequate, as the operator sometimes has to view a faraway panel, and at other times he/she has to have a close look at e.g. a switch which he/she wants to manipulate. No matter what focus was fixed before the program run, it was not adequate for all cases.

The stereo camera in the BGE has been built on the basis of real cameras rather than human eyes. It provides two pictures for the two eyes, and certain parameters can be set in it, but the possibilities for runtime adjustments are few.

The eyeballs of the two human eyes are almost parallel when looking at a faraway object. In this case the two pictures that the left and the right eye get, respectively, are almost the same. However, when looking at a near object, the eye muscles do not only change the focus of the eye lens (this process is called accommodation) but the eyeballs also turn a bit inwards around their vertical axes (this automatism is called vergence), so the two pictures that the two eyes get can differ considerably. They can be roughly approximated as images partially shifted left and right (this approximation does not model properly the inward-turning movement of the eyes when looking near). A good overview of the subject of stereoscopic view in head-mounted displays, with explanatory drawings and a list of further literature can be found in [65].

In the BGE it is possible to adjust the focus of cameras, even programmatically during the program run, and it is also possible to set a camera parameter called “eye separation” (which is a somewhat misleading name, as this distance is not equivalent with human interpupillary distance but it is the distance of the two “subcameras” in the stereo camera). However, tests revealed that the product of the values of focus and eye separation is always 30, if you change one of the values, the other will automatically change accordingly. This seems to be the hard-coding of a “rule of thumb” mentioned e.g. by

http://paulbourke.net/stereographics/stereorender/, even though its origin is not clear. So, in the BGE, you have only one parameter in your hands when you want to focus the camera.

In principle, Blender provides a possibility to specify an object the camera should focus on continuously, but in practice this does not work in the standalone BGE (at least not in the version used for the project).

There existed Blender addons which controlled the focus. However, neither of them supported the BGE used in this project.

Nevertheless, with the use of raycasting [12], it is possible to implement an autofocus feature programmatically.

A solution should use computing resources as sparingly as possible, as this is a real-time application which fulfills several functions periodically: rendering a stereoscopic picture for the headset, handling rotational input from the headset and hand movement input from the Leap Motion, updating the 3D model accordingly, handling data input from the simulator database and animating display objects accordingly, sending data about user actions (such as pressing a button) to the simulator database. Python scripts can be run in the model, but this is interpreted code which runs more slowly than compiled code, so, when possible, complicated calculations should rather be performed by the inherent, compiled code of the BGE and not by an interpreted script.

As this version of the application does not include eye tracking, it is assumed that the object of interest is at the middle of the screen (the user should position himself/herself so that whatever he/she wants to focus on should be in the middle of the screen).

In order to adapt the stereo picture so that the image of the object of interest gets in focus, we have to determine how far that object is. The BGE function *getScreenRay* can cast an invisible ray from the camera towards a point on the screen (specified with x and y coordinates within the [0,1] range). The maximal length of the ray may be specified. The return value is the first object hit or *None*. On the basis of the object's identity, the distance between (the base of) the object and the camera can be calculated. (A more correct distance calculation could use the exact hit point where the ray hits the object, but this simpler method proved adequate.)

To keep calculations to the minimum in the runtime application, it was decided that the optimal eye separation value as the function of object distance should be determined experimentally. For this, an experimental build was made from the application. In this version the user can change the position of the camera and aim it at any object in the control room by orienting the view (with head rotation) in the headset so that the object of interest gets to the middle of the screen. Then, by using raycasting (described above), the identity of the object and its distance from the camera can be determined. After the object has been targeted, the user can manually increase or decrease the value of the eye separation in small increments/decrements. When he/she judges that the stereo picture in the headset is satisfactory, he/she presses a key which records the eye separation value set, together with the distance of the object. From these data, an algorithm can be derived, determining an eye separation value as a function of distance. As computing resources should be used sparingly, the algorithm had to be simple. A few keyboard keys were used for the various operations. The user had to handle these keys blindly (note: the real runtime application does

not require the use of the keyboard during the simulation run while the user wears the headset).

Separate keys were used for moving the camera nearer to or away from the control room panels. Another pair of keys were used to increase/decrease eye separation value for the renderer, by a small amount. Yet another key had to be pressed when the object to focus on was in the middle of the screen and pressed again when the stereo experience was adequate (after adjusting eye separation). Then it made a "snapshot": it logged camera position, the name of the object which was in "focus" in the middle of the screen and its distance from the camera, and it also recorded the eye separation value. A further key suspended raycasting (to prevent accidentally hitting other objects with accidental head movements during the eye separation adjustment attempts). Raycasting was automatically resumed when the key to make the snapshot was pressed.

The functions *bge.render.getEyeSeparation* and *bge.render.setEyeSeparation* were used for getting and setting eye separation, respectively.

Based on the measured results, a primitive algorithm has been constructed. It uses only a few comparisons and no mathematical calculations (not even interpolations): Depending on the range the distance value falls into, a corresponding eye separation value is selected from 10 distinct eye separation values.

According to tests made with the above algorithm in effect, focusing has become adequate, some numerical displays and texts on control panels are more clearly discernible when the autofocus works. (More formal tests are planned,

in which users' opinion about the implementation of the autofocus feature will be asked in a questionnaire.) For the user it takes a little time to get used to setting the view so that the object of interest gets to the middle of the screen. (This would become unnecessary with the implementation of eye tracking.) It must be noted that with changing the eye separation value, the focus value automatically changes, which in turn automatically changes the field of view, so some zooming occurs. However, according to the first tests, this is tolerable.

2.5. Headsets: problems and perspectives

Current headsets do not really have the potential for becoming daily wear yet. They suffer from a number of problems. These may deteriorate the user experience and might even cause discomfort and sickness. There are areas where improvement is possible.

2.5.1. Physical characteristics

Physical characteristics such as size, weight, shape, the feel of the material, adjustability to heads of different shapes and sizes all influence whether a headset can be worn comfortably at length. It also matters how much it heats during use and how much ventilation it provides. Field of view can also be an issue.

Read more in [159] and *https://www.jabil.com/blog/top-augmented-and-virtual-reality-challenges.html.*

2.5.2. Resolution and graphical data communication throughput

The resolution of the Oculus Rift used in our project is not really adequate, the control room looks a bit "pixelated". There are already better, affordable headsets available with a "somewhat" better resolution, the question is how

much is enough. This also depends on the field of application: a technical application like ours does not demand as high a resolution as an artistic application using refined shapes and colors would.

Having higher resolution also means that considerably more graphical data has to be sent from the graphics card to the headset. The Oculus Rift CV1 is able to use the HDMI port of a PC, but headsets of higher resolution must use the Display port, because of the higher throughput demands (which are also affected by the refresh rate).

However, throughput demands can be reduced with a trick called foveated rendering: using eye tracking, you are able to tell where the user looks, and it is sufficient to display only that area in high resolution. while the surrounding area may be displayed in lower resolution. More info on foveated rendering: *https://www.cnet.com/tech/mobile/vr-games-can-look-amazing-with-this-game-changing-imaging-tech/* Varjo's solution optically combines two displays together: *https://varjo.com/blog/introducing-bionic-display-how-varjo-delivers-human-eye-resolution/*

It must be mentioned that throughput demands and the advances in wireless communication technologies (5G etc.) influence whether quality headsets must be tethered (connected to the computer with a cable) or could be wireless.

2.5.3. Vergence-accommodation conflict

Two characteristics of the workings of the human eye, accommodation and vergence, have already been mentioned in the previous section. The development of these two biological mechanisms has taken millions of years. They operate in tune, and, unfortunately, current headsets cause a conflict

between them. The essence of the problem is that the screens presenting the two images for the two eyes are at a fixed distance from the eyes, causing them to focus at this distance. However, a 3D object can be anywhere in the virtual space. In the case of near objects, the vergence mechanism, making the two eyeballs rotate around their vertical axes towards the object, comes into effect, while the focus still remains fixed. This mismatch may cause discomfort, nausea, "simulator sickness". There are attempts to resolve this problem (see [154], [151]), but a really good solution has failed to emerge so far.

2.5.4. Drift

Sensors measuring rotation are not infinitely precise. Small errors accumulate and influence the measured values. Such error is called drift. E.g. in our project the *OpenHMD* package used does not perform any drift correction. When you put down the headset on a table but you leave the rotation detection on and rotate the view in accordance with the "measured" rotation, you will notice that the view will slowly rotate, even though the headset is completely motionless. Such drift has to be compensated, especially if the headset is used for more than a few minutes at a time. The SDK-s provided by the vendors usually – more or less – perform this compensation, but a good compensation is usually possible only when you are able to compare the measured values with values obtained from other tracking systems parallelly active. (For drift compensation for the Oculus Rift, see

https://freecontent.manning.com/wp-content/uploads/2015/05/oculus-sdk-drift-correction-and-prediction.pdf
https://www.roadtovr.com/oculus-rift-sdk-updated-to-v0-2-1-initial-magnetometer-drift-correction-and-more/ .)

2.5.5. Combining with sensors and providing stimuli for other senses

Most available headsets are already combined with at least rotational tracking sensors to track head rotation, but often also with positional tracking. Hand-held controllers and audio output (headphones) are also common. The commercial success of a headset may highly depend on how much its vendor can provide a solution as complete as required. In the future this may include haptic feedback as well. Eye tracking has already been mentioned above, it can be used not only for foveated rendering but also as a means of interaction with the virtual world.

3. Rigged hand model for the Blender Game Engine[2]

In a lot of virtual reality applications, it is necessary to ~~display~~represent and animate the user's hand in the virtual world. Such is the case for our virtual model of the control room. It increases realism of the virtual reality application, if realistic-looking models of control organs such as pushbuttons and switches are operated with a realistic model of the user's two hands. In our solution, only the hands are displayed, and not the arm and the further parts of the user's body, as those play no role in the interaction.

In game engines, armatures are often used for modeling parts or the whole of the human body. The process of creating the armature and assigning parts of the mesh representing flesh and skin to the “bones” of the armature is called rigging. After performing rigging, moving the armature will result in the movement of the mesh, thus the virtual hand (or other object) is animated. The Blender Game Engine has been one of the few open-source game engines available. This chapter gives a detailed description of the process of successfully building a complex rigged hand model for the BGE and gives guidance for driving this hand model with input data from the Leap Motion hand movement detector. The rigged hand model has been implemented, using the hand mesh from the LibHand library and an armature specifically built to match the hand representation of the Leap Motion. The model was intended for navigation and interaction in the virtual model of a power plant control room.

[2] This chapter is an abridged and re-worded version of the article [37] of the same title, without the Python code samples and the references to them.

3.1. Introduction

In the world of virtual reality, the term “rigging” is used for the preparations to move (animate) virtual objects. Often these virtual objects are the whole or parts of the human body, especially the human hand. Depending on the requirements of use and the technical possibilities of the actual application (a game, a virtual tour of an existing or imaginary place, an instructional application etc.), the complexity of hand models varies greatly. A rudimentary 3D hand model may consist of five cuboids or cylinders representing the fingers. A complex hand model is more realistic in shape, with sophisticated coloring, it may even display small freckles on the skin of the hand.

These virtual reality applications are often realized in game engines, where the more complex animated hand models are usually constructed from armatures, which can be regarded as “skeletons” consisting of bones. These virtual bones and the bone structure they belong to may quite closely resemble the real bones of a real hand, or they may bear only a passing resemblance to them. During animation, the bones themselves are usually not displayed. Instead, “flesh” and “skin” (which are – usually complex – meshes in the game engine), which have been “glued” to them, are displayed. In this context, rigging is the process of creating the armature and the “gluing” (assigning parts of the mesh to particular “bones” of the armature). After the rigging has been performed, moving the armature will result in the movement of the mesh, thus the virtual hand (or other object) is animated.

The Blender Game Engine is a component of Blender (*www.blender.org*), a free and open-source 3D creation suite (cf.[45], [123]) used for making real-time

interactive content. The version used in the project is 2.76b, under the Windows 10 operating system.

The Leap Motion hand tracking device (*www.leapmotion.com)* has been commercially available since 2013. Its new, much improved Orion API (application programming interface) was introduced in 2016, so earlier criticism of the accuracy of the device are no longer valid. Among other applications, there have been attempts to integrate the device into the Blender Game Engine as well. The simple hand model published by Magnus Benjes at *www.magben.de/?h1=3d* (now available at *https://web.archive.org/web/20200809221629/http://www.magben.de/?h1=3d*) works fine, but it does not use armature and rigging, the cylinders representing the fingers are driven directly by position and orientation data from Leap Motion. It is not easy to build for the BGE a more sophisticated, armature-based rigged hand model with which it is possible to simulate (and animate with Leap Motion data) a hand using a complex mesh instead of objects of simple geometry like cuboids or cylinders. Those who tried that path have usually reported failure and switching to another game engine, mostly to Unity. The probable reason for the failures is the peculiarity and complexity of the way armatures are handled in the BGE. However, as Blender is a game engine which is completely free and open-source, and most alternative tools are not, a solution in the BGE could come in handy for those wishing to use an open-source, free game engine. This paper gives a detailed description of the process of successfully building such a hand model and gives guidance for driving this hand model in the BGE with input data from the Leap Motion. It must be noted that there is a similar project mentioned at *http://blog.leapmotion.com/manipulating-rigged-hand-with-leap-motion-in-three-js/*

but it uses the *Three.js* API, not the BGE, and the poster admits to having “tweaked almost all data to make the model behave nicely on the screen”.

3.2. Leap Motion's representation of the hand

The relatively inexpensive but reasonably accurate, optical-based hand detector device is connected with a cable to the USB port of a computer. It uses infrared LEDs and cameras and gives quite detailed position, orientation, length and width information about the individual bones of the hand, and also some gesture detection (fist etc.). It provides APIs for various programming languages. A feature list can be found at *blog.leapmotion.com/getting-started-leap-motion-sdk/*.

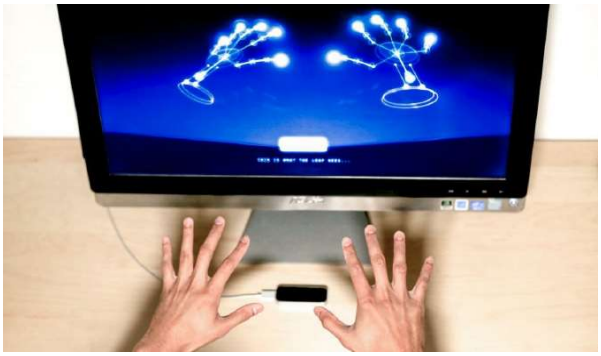

Fig. 3.1. The Leap Motion device [74]

The bones of the (right) human hand can be seen in Fig. 3.2 (with the palm facing the viewer).

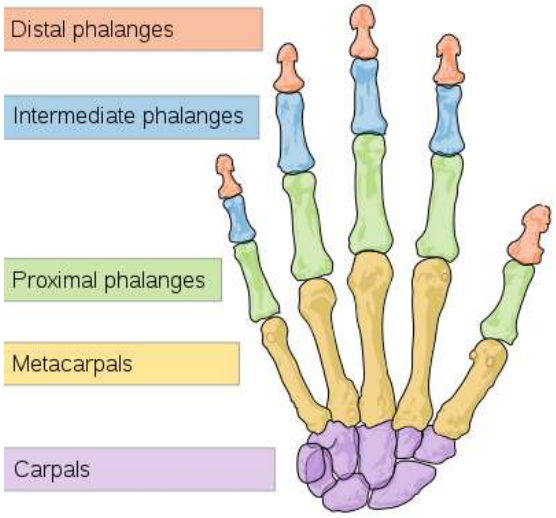

Fig. 3.2. The bones of the human hand
(*https://commons.wikimedia.org/wiki/File:Scheme_human_hand_bones-en.svg*)

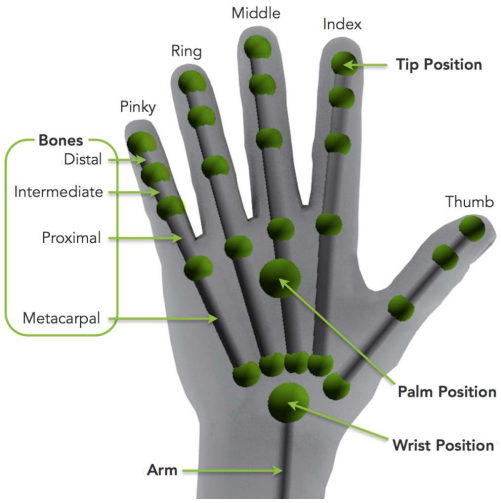

Fig. 3.3. Leap Motion's hand representation (adapted from
http://blog.leapmotion.com/getting-started-leap-motion-sdk/)

Leap Motion's representation of the hand (including a part of the armbone) can be seen on Fig. 3.3.

3.3. A step-by-step guide to the rigging

This section describes the main steps of creating a usable rig for a relatively complex hand mesh in the Blender editor, to be used later in the BGE with the Leap Motion hand detector. Some familiarity with Blender's editor and the BGE is necessary for understanding the details. There are code samples (in Python which is used by Blender) in the Appendix of the original paper. The method can be adapted for hand meshes other than the one used here, and also for other hand movement detectors.

The hand mesh chosen is the mesh in the Blender file published at the LibHand website (*www.libhand.org,* updates may be found at *github.com/libhand/libhand*). For the mesh, a new armature had to be constructed, to match the hand representation of Leap Motion.

The rig is for the right hand. You will later be able to derive the left hand from the right hand by mirroring the armature.

The basic idea behind this particular rigging solution is controlling the individual bones in the armature through *Copy Rotation* constraints which can be set in the Blender editor and will work in the BGE. For each bone, a cuboid of similar size is created. When the BGE application runs, each cuboid will get the orientation information for its corresponding bone from the Leap Motion device (through Leap Motion's Python API). Through *Copy Rotation* constraints, the orientation of each bone will be exactly the same as the

orientation of the corresponding cuboid. However, as the author's experiments have revealed, due to the peculiarities of the BGE (some constraints are not completely or correctly implemented), these constraints will work properly only if the orientation of the armature never changes. It should always remain in an upright position (in a Z-up coordinate system), and only the orientation of its bones may change. (It should be noted that neither the armature itself, nor the bones can be seen when the virtual reality application runs, only the hand mesh "glued" to the bones of the armature is visible. Thus, regardless of the theoretical orientation of the armature, the bones will get the correct orientation. Consequently, the mesh will be correctly deformed and moved.) The position of the armature will change according to the position of the hand (as detected by Leap Motion).

The difficult part is to make a usable rig for the hand mesh so that the rest position of all of the armature's bones is the default upright position (otherwise the *Copy Rotation* constraints will not work correctly). For this project, the following rigging method has been worked out: The armature bones should first be posed normally, so that they match the hand mesh that you use, then the parenting of the mesh to the armature should be performed. After that, the armature bones should be posed to an upright position (with a script). By performing a trick with the modifier of the mesh, you can set this default upright pose as the new rest pose, without corrupting the rig. The resulting armature and the mesh rigged to it will look strange, but the mesh, controlled by the bones, which in turn are controlled by the orientation of the cuboids, will be animated correctly during the BGE run.

The cuboids used for the *Copy Rotation* constraints may have an additional function (provided that not only their orientation but also their position is updated according to the data taken from Leap Motion): if you need to handle collisions in your application, these simple cuboids, made invisible, may be used as a "shadow hand" which can collide with other objects, while the more complex hand mesh is left out of collision detection, thus conserving computer resources (this trick is often used in 3D games).

After this introduction, let us see the actual steps of the rigging.

A. Make an armature which fits your mesh

The way of creating armatures is described in Blender tutorials. First create the correct bone structure. During this first phase the exact bone lengths are not a priority, you should care about the correct topology only. It is not necessary to restrict the degrees of freedom for the bones, this will be taken care of by Leap Motion (*blog.leapmotion.com/skeletal-tracking-101-getting-started-with-the-bone-api-and-rigged-hands/*).

The next phase is setting the correct bone lengths. To facilitate this, you can make a run with the Leap Motion and print out the measured bone lengths, and you can set the bone lengths accordingly (manually or with a script).

For the mesh taken from LibHand, the armature in Fig. 3.4. has been constructed by the author. (This armature fits Leap Motion's hand model more closely than the original armature of LibHand and has a continuous topology.)

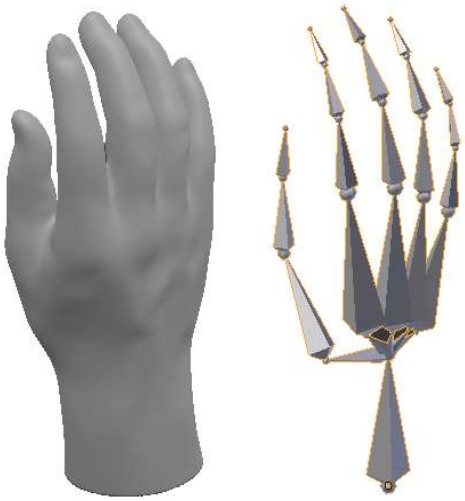

Fig.3.4. Hand mesh (from *www.libhand.org*) and armature (developed for this project)

In this armature, apart from the very first bone (the armbone), each bone has a “parent” bone to which it is connected. To ensure this continuous topology, certain “fictitious” bones had to be introduced, even though they have no corresponding bone in the hand model of Leap Motion. They connect the wrist position (see Fig. 3.3) with the origins of the fingers. These fictitious bones differ in one thing from the "real" bones: they are set as non-deforming bones, so that the mesh will not be “glued” to them. (All other bones should be set as deforming bones.) The *Inherit Rotation* setting should be on for all bones. The lengths of the fictitious bones can be calculated from the end points of the connecting bones (which you can obtain from a Leap Motion run).

The third phase is to pose the bones so that they match the hand mesh as closely as possible. The posing can be aided by posing your real hand and taking note of the orientation of the bones as provided by the Leap Motion. You may use

scripts for this but you will probably have to adjust some bones manually as well. This is usually an iterative process. Then perform “Apply Rotation” (for mesh and armature) in the Blender editor.

B. Zero all bone rolls

You may do this manually or with a script, in Edit mode (see sample script in the Appendix of the original paper).

C. Make the origin points the same for both armature and the hand mesh

1) For each of these objects, set the origin to the object's lowest point center:
In Edit mode, move the 3D cursor to the desired origin point (to the point where the armature begins, the lowest round point in Fig. 3.5). Press Shift+S and choose the option Cursor to Selected.

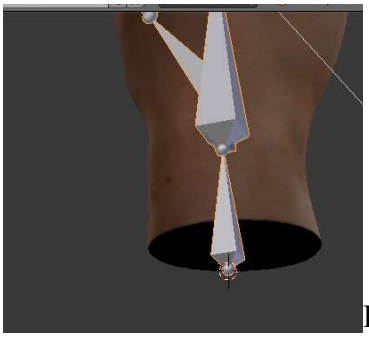

Fig.3.5. Setting the origin point

In Object mode, select one of these objects and press ctrl/alt/shift c, then choose Set origin to 3D cursor.

Do the same for the other object.

2) In Object mode, position the hand mesh and the armature so that their origin points occupy the same place:
Make the XYZ coordinates of the two objects the same, by copying the coordinates of one into the other.

D. Temporarily upscale mesh and armature

This step is recommended by the author because, in her experience, upscaled objects seem to rig better (at least when parenting is performed with automatic weights). Make sure that no other object shares the space with the upscaled objects. Do the scaling (e.g. by a factor of 100) in Object mode, then apply scale.

E. Perform automatic assignment of mesh points to the bones

Parent the mesh to the armature: set Object Mode, select the mesh, shift-select the armature, press ctrl/p and select the “Armature Deform”, “With Automatic Weights” option.

F. Downscale the rigged armature

In Object mode, scale the armature (not the mesh) back to its original size and apply scale.

G. Align all bones to vertical

This can be done with a script. First create an auxiliary armature with one bone being in a vertical position (that is the default). Then align each bone in the armature of the hand to this new bone (you can find a sample script in the Appendix of the original paper).

H. Make this pose the new rest pose

This subsection is based on *nixart.wordpress.com/2013/03/28/modifying-the-rest-pose-in-blender/*.

In Object Mode, select your deformed hand mesh object.

In the object's Object Modifiers stack (denoted by the spanner symbol), copy the Armature Modifier by pressing the Copy button.

Apply the first Armature Modifier (the top one), but keep the bottom one. The latter will replace the old Armature Modifier and will allow to pose your object with respect to your new rest pose. At this point, the object will still be deformed twice. That is because we need to apply the current pose as the new rest pose.

Select your armature and set Pose Mode.

Select "Apply Pose as Rest Pose" in the Pose menu (which you can invoke by ctrl/a). This will clear the double deformation and put your object in your new rest pose.

I. Create cuboids

Create a cuboid object for each bone in the armature. All cuboids should have a vertical orientation. If you want to use these cuboids later for a "shadow hand", then make the lengths of the cuboids match the lengths of the armature bones.

J. Create Copy Rotation constraints

Set Pose mode. For each bone, create a Copy Rotation constraint to drive the bone in the armature with the rotation of the corresponding cuboid. The constraint subtype should be *World → Local with Parent*.

You can set these constraints manually, but it is easier to do it with a script (see the Appendix of the original paper).

3.4. Interfacing with Leap Motion

Our rigged hand model is ready, now we want to drive it runtime in the BGE, with input from the Leap Motion device. Our task is to set the position of the armature and the rotation of the cuboids controlling the rotation of the bones.

A. Interfacing the BGE with the Python API of Leap Motion

The Python API of Leap Motion (Orion 3.2.1) uses Python 2.7, while Blender 2.76b uses Python 3.4. (Please note that different Blender versions may use different Python sub-versions.) Therefore, a Python wrapper has to be generated, this can be done with the SWIG interface generator, using the method described in *https://web.archive.org/web/20230528073139/https://support.leapmotion.com/hc/en-us/articles/360004362237* (the referenced page uses Python 3.3 but the method also works for Python 3.4).

B. Driving the cuboids with Leap Motion data

As already mentioned, an example with a simple hand model can be found at *https://web.archive.org/web/20200809221629/http://www.magben.de/?h1=3d)*. Its hand model consists of simple geometric shapes.

For a complex, armature-based hand model to work, we need detailed information from the hand movement detector. Fortunately, Leap Motion provides information about how much the bones are rotated around their own longitudinal axis (this rotation is called the “roll” of the bones in armatures). To illustrate the importance of this, let us consider Fig. 3.6.

If the finger is modeled as a simple cylinder, then the above mentioned rotation will not matter. However, if we make a fingernail on the cylinder (or “glue” a mesh, which includes a fingernail, to the cylinder), then this rotation will matter, as the fingernail will be visible.

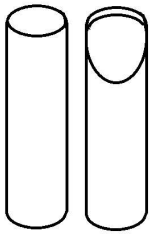

Fig. 3.6. Finger models without and with fingernail

The y basis vector is one of the three orthonormal basis vectors provided by the Leap Motion API. This vector is perpendicular to the longitudinal axis of the bone, it can be visualized as originating in the center of the fingernail and pointing outwards (in Fig. 3.6 the y basis vector would point from the fingernail directly towards the viewer). With the aid of this vector it is possible to set the bone roll for each finger correctly. This holds for all bones which have a representation in the Leap Motion hand model.

However, in our armature there are also fictitious bones, for which we get no basis vectors from Leap Motion.

C. Calculating the orientation of the fictitious bones

The orientation of such a bone (as a vector) can be calculated from its origin point and end point: the end of the armbone and the start of the metacarpal bone (these data are provided by Leap Motion). Thus we can set the orientation of the

bone. The roll of the bone does not matter because the bone has been defined as a non-deforming bone, so no mesh (“fingernail”) is “glued” to it.

The update of the hand mesh with the data obtained from Leap Motion is performed in the BGE in a loop consisting of the steps which can be seen in Fig. 3.7. Steps 1,2,3,4 are executed in the Python script written for the project, step 5 is performed automatically by the BGE.

The above mentioned code has been integrated into the BGE application, together with gesture detection and navigation – these latter are still under development and are not discussed in this paper. The total number of Python source lines written for the BGE application so far exceeds 1500. A few dozen of these source lines have been adapted from *www.magben.de/?h1=3d*, the rest is original code specifically developed for the project.

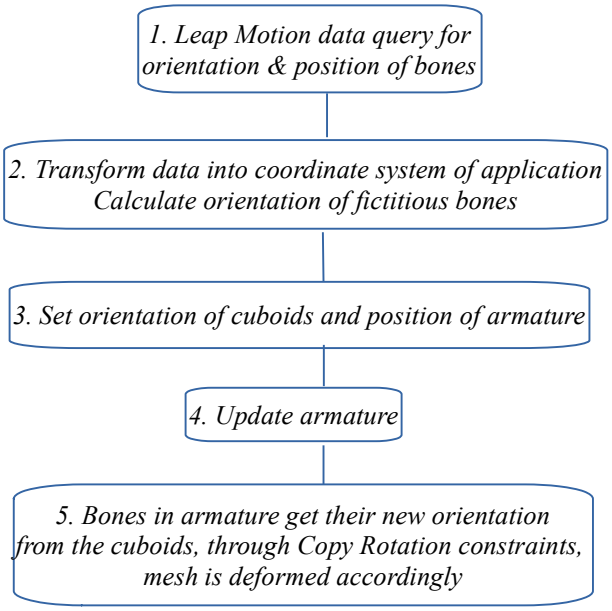

Fig. 3.7. Steps of updating the hand mesh

3.5. Summary

The hand model described above has been tested in the BGE and it reasonably accurately follows the user's hand movements, for the right and the left hand as well.

The use of such a hand model is especially advantageous in immersive virtual environments, with the user wearing a head-mounted display and not seeing his/her real environment. In such a situation, he/she cannot see the input devices, either, and if these input devices require touching (like a keyboard, a mouse, a game controller), this touch should be performed either “blindly”, or a virtual representation of the input device should be displayed in the virtual environment, and this would usually require displaying a virtual model of the user's hand as well. Other possible applications would use only the user's hand as the “input device”: by controlling the virtual hand with his/her real hand, the user could interact with objects in the virtual space directly, and hand gestures could be used for navigation and other actions as well (example: clenching the hand into a fist could mean that the user wants to quit the program). In these applications, it is also advisable to display the virtual hand, to provide feedback for the user.

We planned to use the virtual hand for navigation and interaction in the virtual model of a conventionally equipped nuclear power plant control room [114]. The navigation was planned to be performed by hand gestures. The visual representation of the user's hand in the virtual space provides feedback for the user in the navigation, and shows the hand's position and orientation, relative to

the physical devices in the control room, for the interaction with these devices. The interaction with the virtual models of physical devices (switches and pushbuttons) of the control room can be realized by colliding the virtual hand with these devices and sensing pushing (of buttons) and turning (of switches) from the dynamics of the hand movement. As affordable haptic (tactile) feedback devices are still in experimental phase, the feedback about the collision could be visually provided or indicated by audio effects. Thus staff could exercise and even experiment in the virtual control room, without having to use the real control room or a physical replica of it.

4. Interaction in an immersive virtual reality application[3]

Immersive VR (virtual reality) applications usually require styles and devices of human-computer interaction other than the conventional ones like mouse and keyboard, as the user typically wears a head-mounted display and does not see his/her real environment. Moreover, such interaction is usually three-dimensional. This is where relatively new (or at least previously not dominant) paradigms come into the picture: natural user interface (NUI) and gestures. The paper overviews some definitions for these and discusses the communicative aspects of gestures. The above mentioned paradigms affect the process of interaction design, which has to be even more thorough, more conscious than before. If we use gestures for input, the actual requirements of the application have to be analyzed in detail and decision about gesture "vocabulary" has to be taken. In the paper, an actual implementation under development is discussed: the interaction – consisting of navigation in 3D space and manipulation of virtual objects – in an immersive VR application simulating the conventional control room of a nuclear power plant. The implementation is touchless, using a hand movement detector.

[3] This chapter is based on the conference paper [262] of the same title by the author. ©2019 IEEE.

4.1. Introduction

The definitive book about cognitive infocommunications (CogInfoCom), which describes its theoretical foundations, states that *"one can find in it the influence of well-established research fields (e.g. augmented cognition, human-computer)"* [29], p. ix]. Chapters 4.7 and 4.11 of the book are dedicated to human-computer interaction (HCI) and virtual and augmented reality, respectively, and it is noted that *"HCI is strongly relevant to CogInfoCom when it comes to the analysis and design of functionally motivated user-device interactions"* [29], p. 49.

Immersive VR (virtual reality) systems can be described as VR systems where "*Users wear displays that fully immerse a number of the senses in computer generated stimuli. The stereoscopic head-mounted displays (HMD) are a distinctive feature of such systems* [44, p. 59]." Such applications usually require styles and devices of HCI other than the conventional ones like mouse and keyboard, as the user does not see his/her real environment (which includes input devices). Moreover, the spatial part of the interaction usually occurs in three dimensions. This is where relatively new (or at least previously not dominant) paradigms come into the picture: the so-called natural user interface (NUI) and gestures on which the interaction may be based. Some CogInfoCom-related papers mention or discuss gestures and/or NUI from a practical point of view [265] [43] [238] [204] [215] [216] [252] [264] [209] [168] [64] [266] [267] [214] [152] [206], but the definition of these concepts has not been in the focus of the papers ([205] excepted).

4.2. NUIs

Baranyi and Csapó note, "*The inspiration to create engineering systems capable of communicating with users in natural ways is not new* [28], p. 141." Nevertheless, technological advancement has recently created new possibilities for this paradigm.

Natural user interfaces are regarded as the third generation of human-computer interfaces, after command line interfaces (CLI) and graphical user interfaces (GUI). CLIs are based on text input typed by the user (and usually text output is provided by the computer), while GUIs are mostly based on the desktop metaphor and the WIMP (windows, icons, menus, pointers) style of interaction.

The definitions for the term NUI are numerous and somewhat diverse. According to Glonek and Pietruszka, "*Term NUI stands for the ways of interaction with a device based on methods other than a mouse and a keyboard that would at the same time be as natural and intuitive for a human being as possible* [105], p. 28."

A book entirely dedicated to NUIs, written by two touchscreen specialists, gives a somewhat vague definition, linking NUI to the emergence of new input devices: "*Decades of cumulative experience in creating interfaces for new technology led us to two important realizations. First, that new input devices do not, in and of themselves, facilitate a better user experience—we argue that the iPhone and Microsoft Surface UIs are highly successful in spite of, rather than because of, the use of a touchscreen. The second realization is that these input devices, while not themselves creating a better user experience, could be*

enablers for the creation of a UI that is more natural to use, and could fundamentally change the way we interact with technology. We dub this the natural user interface [283, p. ix]."

As far as this definition is concerned, it must be noted that touchscreens have been around for several decades. However, with the ever-increasing use of smartphones they have become popular, also affordable, they have gained new, more sophisticated features and the use of some already extant features has become widespread (e.g. multitouch – the ability to detect input from multiple points of contact simultaneously – , better sensitivity etc.). Increased affordability and sophistication actually characterize most of the input devices NUI is based on, even though the devices are hardly completely new.

Another expert in the field, Joshua Blake, has defined NUI as "*A natural user interface is a user interface designed to reuse existing skills for interacting directly with content.*" [46], p. 2.

The three important features of the above definition are:

• NUIs are designed, i.e. they must be carefully planned, with the future use and users in mind.

• NUIs reuse existing skills, where 'skills' mean mostly non-computing skills: human-human communication – verbal and non-verbal –, and human-environmental interaction. An interface will feel "natural" when it builds upon these existing skills. According to Blake, "*Computing power and input technology has progressed to a point where we can take advantage of these existing non-computing skills. NUIs do this by letting users interact with computers using intuitive actions such as touching, gesturing, and talking, and*

presenting interfaces that users can understand primarily through metaphors that draw from real-world experiences [46], p. 2."

• NUIs have direct interaction with content. Blake explains it as "*(...) the focus of the interactions is on the content and directly interacting with it. This doesn't mean that the interface cannot have controls such as buttons or checkboxes when necessary. It only means that the controls should be secondary to the content, and direct manipulation of the content should be the primary interaction method* [46], p. 2."

Wang et al. state that "*NUIs build on traditional human-tohuman interaction models, intend not to be instrumented through artificial control devices (that is why they are named as 'natural') and aim at being or becoming "invisible" after a learning process (the system is naturally giving the user the feeling that he is continuously and instantly successful). Voice control, gaze tracking or gesture identification are taken as a basis to implement NUIs* [278], p. 535."

Harpstead et al. view the matter this way: "*(...) we reviewed the HCI literature on natural interactions and identified four common characteristics of systems that support naturalness: they (1) support the goals of the user, (2) do what the user expects, (3) allow the user to work the way they want, and (4) leverage users' experience to minimize training* [83], p. 2."

Wigdor and Wixon note that "*we see natural as referring to the way users interact with and feel about the product, or more precisely, what they do and how they feel while they are using it.*" "*Users focus on content and so should the interface. Provide the absolute minimal number of interface elements that are required for the interaction. For some interactions that is no interface*

beyond the content." They also warn that "The greatest challenge in building a NUI system is making it learnable." Also, "*Never rely on an action being 'natural' (a.k.a. 'guessable'). It's not.* [283], p. 9."

Based on the above listed definitions and features and her own developer's experience and reasoning, the author of this paper proposes the following, somewhat colloquial definition:

A NUI is a human-computer interface which serves and is based on human nature more closely than the currently widespread traditional interfaces. (Here "human nature" means common human characteristics and skills we are born with or develop in our early or more advanced years. It must be noted that cultural differences apply.) Obviously, this definition may become outdated after a while if and when NUIs start dominating the field. However, GUIs based on the desktop metaphor will surely stay with us for a while.

It should also be emphasized that the learning of the NUI should always be facilitated.

4.3. Gestures

Media philosopher Vilém Flusser has defined gesture in this manner:

"*(...) one way of defining 'gesture' is as a movement of the body or of a tool attached with the body, for which there is no satisfactory causal explanation. To understand a gesture defined in this way, its 'meaning' must be discovered. That is exactly what we do all the time, and it constitutes an important aspect of our*

daily lives. But we have no theory of the interpretation of gestures and are restricted to an empirical, 'intuitive' reading of the world of gestures, the codified world that surrounds us. ... If someone punches me in the arm, I will move, and an observer is justified in saying that this reaction 'expresses' or 'articulates' the pain I have felt. There would be a causal link between the pain and the movement, and a physiological theory to explain this link, and the observer would be right to see this movement as a symptom of the pain I have suffered. Such a movement would not be a 'gesture' according to the suggested definition, for the observer would have explained it in a satisfactory manner [88], pp. 3–4."

By emphasizing the lack of the causal explanation, Flusser actually says that gestures are intentional, deliberate.

Later in his book Flusser notes that "*the communicative aspect of a gesture overshadows all else*" [88], p. 161.[4]

Several similar definitions can be found in the literature:

"*A gesture is a motion of the body that contains information* [165], p. 310."

"*Gestures are the motion of the body which is used with an intention to communicate with others. And to make this communication effective and successful both sender and receiver must have the same set of information for a particular gesture* [137], p. 633."

"*As a gesture, in the context of HCI, we define a coordinated and intended movement of body parts to achieve communication* [63], p. 1."

[4] For those wanting to see this field from the viewpoint of communication and sensory perception, Part III of [21] is recommended.

"*Gestures are not just movements and can never be fully explained in purely kinesic terms. They are not just arms waving in the air, but symbols that exhibit meanings in their own right* [63], p. 105."

Thus, based on the above definitions of Flusser and the others, we can say that a gesture is a movement of the body (or of a tool which can be regarded as the body's extension, but it is also operated by the body, and this includes the brain in a brain-computer interface), and *a gesture always has a "meaning", a message,* i.e. it is somehow *coded*. It must be noted that not only humans but also animals can make gestures.

The "coded" quality means that gestures should be interpreted according to a common "code book" shared by the gesturer (the one who gestures, the sender) and the receiver/interpreter(s) of the gesture. A well-known example illustrating the importance of this is the shaking (moving from side to side) of one's head: this gesture means "no" (denial or disagreement or rejection) in most countries and cultures.

However, there are a few countries such as Bulgaria and Greece, where this particular gesture indicates affirmation, i. e. "yes" [100].

Some authors narrow down gestures to hand gestures, which may be true for a lot of HCI applications, but this approach is too restricted overall, as people can also communicate with their legs, head movements, facial expressions etc.

The question may arise whether a simple keystroke is a gesture or not. The author of this paper is of the opinion that this depends on the context, the coding. Let us take the pressing of the space bar on a keyboard as an example. If a text interface (CLI) writes, "Press the space bar to continue", then by

pressing that key you convey the message that you want to continue, so this can be regarded as a gesture (even if the interface is not a NUI). If a bored child randomly hits the space bar on a keyboard (and perhaps the computer connected to the keyboard is not even switched on), we can say that this is not a gesture in itself. However, if the child wants to attract mom's attention by randomly pressing a key or keys, then the action already has a message, and thus may be regarded as a gesture (it is another thing whether mom notices the gesture and decodes it correctly).

It must be noted that gesture control of machines existed long before electronic computers came into existence. An example is a music instrument called the Theremin, invented in 1920 [102]. It is an early type of electronic music synthesizer, controlled by touchless gestures, the frequency and the volume of the generated sound can be modulated by hand. As far as the use of gestures in HCI within NUIs is concerned, there are several potential pitfalls to be aware of.

A critical expert in the field of design and human-computer interaction, Donald A. Norman, has listed several problems with NUIs and especially gesture control in his article provocatively titled “Natural User Interfaces Are Not Natural” [210], pp. 6, 9, 10:

He states that “*Most gestures are neither natural nor easy to learn or remember.*” He cites cultural differences (the same gesture may mean different things in different cultures).

He remarks that physical gestures may have side effects. In extreme cases they can even do damage. He illustrates this with a case of the Nintendo Wii game

controller used in a virtual bowling game. "*(...) the 'natural' interface was to swing the arm as if holding a bowling ball, and then, when the player's arm reached the point where the ball was to be released, to release the pressure on the hand-held controller's switch. Releasing the pressure on the switch was analogous to releasing the ball from the hand and it was readily learned and employed. Alas, in the heat of the game, players would also release their hand pressure on the controller which would fly thorough the air, sometimes with enough force to hit and break the television screen on which the bowling lane was being displayed. (...) Here, the gestural convention was too natural.*" He is of the view that it is "*unlikely that complex systems could be controlled solely by body gestures because the subtleties of action are too complex to be handled by actions – it is as if our spoken language consisted solely of verbs.*"

According to him, "*Because gestures are unconstrained, they are apt to be performed in an ambiguous or uninterruptable manner, in which case constructive feedback is required to allow the person to learn the appropriate manner of performance and to understand what was wrong with their action. As with all systems, some undo mechanism will be required in situations where unintended actions or interpretations of gestures create undesirable states. And because gesturing is a natural, automatic behavior, the system has to be tuned to avoid false responses to movements that were not intended to be system inputs.*"

He states, "*Gestural systems are indeed one of the important future paths for a more holistic, human interaction of people with technology. In many cases, they will enhance our control, our feeling of control and empowerment, our convenience, and even our delight. But like all technologies, gesture-based*

systems will come at a cost. Different systems will devise different conventions. There will be a learning curve. People with handicaps will have to be accommodated. And there will be an entirely new source of material for comedians. Imagine the problems when a system has a repertoire of dozens of gestures, all of which mean something, but not all of which may be known by person near the device. I am reminded of those old movie comedies of people in formal clothing at auctions doing silent bidding. One person sneezes and thereby purchases an unwanted painting. A couple argues, and as they wave their hands at one another, the hand waving gets interpreted as ever-escalating bids."

4.4. Designing gesture-based interaction

Wigdor and Wixon provided this guideline, based on their long practice in the field:

"*Understand that there is no such thing as a 'natural gesture' — you need to design the set of gestures in your system* [283], p. 209."

What does this mean in practice? The "set of gestures" means a gesture "vocabulary": these particular gestures will be recognized and interpreted by the system. As discussed above, gestures are always coded, each has a meaning. One of the first question the designer has to ask is whether all interaction in the system can be handled by gestures. If there are too many functions to be fulfilled, it would be impractical to go beyond a certain number of distinct

gestures – some gestures would be too similar to each other and thus could be misinterpreted.

Also, the user would have to keep a too large “vocabulary” in his mind. In such a situation you should either not design a gesture interface, or create a hybrid system integrating NUI gestures with more conventional methods of interaction, like a soft keyboard occasionally displayed in the application with keys to be pressed (as discussed above, these key presses may be considered gestures but they cannot really be considered part of a NUI).

The set of possible gestures is highly dependent on the *hardware* which has to detect these gestures. You should not choose a hardware which is not accurate and reliable enough for the purpose. Do not rely on descriptions and hype, thoroughly test gesture recognition with the hardware before designing the interaction in detail. For the recognition of gestures, some hardware manufacturers provide a gesture recognition library, in other cases you have to develop your gesture recognition code yourself.

Another decision to be taken early on is the question of *modality*. A gesture may have different meanings in different contexts. Example: a sweeping gesture may mean the intention to move away in a navigational situation, to scroll a document when reading, to dismiss some object etc. If there are different modes in the interaction, then it should always be made clear to the user which mode is actually in effect, and how to go into another mode when desired. The number of modes should be heavily restricted, too many modes are impractical and confusing.

A lot of thought should also be given to how the user will *learn* the use of the system, the gesture vocabulary, the various modes. It is practically inevitable to educate the user in advance about the use of the application. This education should be short, effective, entertaining and quickly refreshable. It should also be decided what kind of *help* (visual or other, if any) should be given to the user during the use of the application.

Then there is the problem of *user fatigue* (bodily and mental fatigue as well) which can occur especially when the user has to repeat the same physical gestures over and over, perhaps in a body position which is not entirely comfortable.

All this points to the necessity of careful design and extensive and rigorous testing.

4.5. Application: interaction in a virtual control room

The full-scope simulator of the Paks Nuclear Power Plant has been in service since the 80's. Even after several refurbishments to the simulator itself [140] [141] [134] [135], the replica control room, which is a conventional control room containing a lot of conventional displays, pushbuttons and switches, is still relatively unchanged. The personnel of the four units of the power plants all receive their training on this simulator, which exists only in one copy, and is much in use. It is also used for testing new control room equipment. With the advancement of technology, it has become possible to create a virtual reality application, the 3D model of the control room [114], which has been integrated with the software simulating the plant, thus it has become possible to use copies

of the simulator with the virtual control room. This version of the virtual control room uses large monitors for output, keyboard and mouse, optionally a Nintendo Wii Nunchuk controller as input. The realism could be increased by using immersive virtual reality technology. For the immersive version, the virtual control room model, which uses the Blender Game Engine (*www.blender.org*), has been integrated with the OpenHMD package (*www.openhmd.org*) which makes it possible to use HMDs like the Oculus Rift. The Blender version used is 2.76b, under the 64-bit Windows 10 operation system. The interaction has had to be completely rethought.

After careful consideration of the technical possibilities and the costs, the relatively accurate and inexpensive, touchless Leap Motion hand movement detector (www.leapmotion.com) has been chosen as the input device, for gesture-based control and for interaction with the – virtually represented – physical input devices in the control room. (A good survey of the Leap Motion's use for 3D interaction and gesture control can be found in [22]. The device has been used in several CogInfoCom-related projects [64] [266] [267] [271] [113] [272].) For displaying the user's hand in the virtual space, a complex rigged hand model [37] has been developed for the BGE. (The virtual hand is necessary for giving feedback to the user about his/her hand movements and for handling the pushing of buttons and the turning of switches on the panels.) When using the application, the user is seated at a desk and has a HMD on his/her head. His/her hand is above the Leap Motion device. He/she can freely navigate in the control room by using hand gestures and will be able to handle all pushbuttons and switches. Only the user's virtual hands – and not his/her other body parts – are displayed in the VR application. The navigation is

realized by gesture-based control. The currently implemented version uses the following operation modes:

• Translational mode (implemented, in beta phase): the user is able to move along the 3 axes of the Cartesian coordinate system. This mode is in effect when only the left hand is within the detection range of the Leap Motion, the right hand is out of sight. The user moves his/her hand along 1, 2 or 3 coordinate axis/axes simultaneously, and the view – and the user's virtual body – moves with it. The most important direction is along the horizontal axis. As the user sometimes has to travel large distances in the huge control room, acceleration has been implemented. This mode ends when the user puts his/her right hand into and/or pulls his/her left hand out of Leap Motion's "sight". When the user's left hand is clenched into a fist, the movement of the hand does not result in the movement of the view – this sub-mode is necessary to make it possible for the user to return his/her hand into the field of detection ("sight") of the sensor and perform further movements. This solution is a bit analogous to a feature of 2D interaction with a mouse: when the mouse reaches the edge of the mouse pad (or of the area provided for the mouse), the user can lift it, return it to the center and continue the movement as if it continued from the edge of the mouse pad.

• Rotational mode (implemented, in beta phase): the user is able to rotate his/her body around the vertical axis, by rotating his/her right hand. This mode is in effect only when the right hand is within the detection range of the Leap Motion and the left hand is out of sight.

• Object manipulation mode – under development – is in effect when both hands are visible. They can interact with buttons and switches in the control room.

In the case of pushbuttons, the collision-handling features of the BGE are used (collision sensors can be deployed among the so-called logic bricks, and when the hand object collides into a pushbutton object, the sensor activates a Python code). The turning of the switches requires self-developed gesture recognition code which has already been written and tested. It determines whether, after the hand-switch collision, there is a rotational movement of the hand and, if so, whether it is clockwise or counterclockwise. It is based on analyzing the change of rotational data (pitch, yaw, roll) obtained from the hand detector.

Additionally, head rotation input – provided by the gyroscope of the HMD – is possible all the time. Unlike the rotational mode described above, this may occur around all 3 axes, and it affects only the user's view (not the position or orientation of the rest of the user's body) within the control room.

4.6. Conclusion and future directions

Definitions and features of natural user interfaces and gesture control have been discussed. Guidelines for designing gesture-based control for applications have been provided. An actual immersive virtual reality application, which is under development and uses a set of gestures detected by the Leap Motion hand movement detector, has been outlined.

Several questions remained open, e.g. how to signal when the user, represented by the hands, collides into a panel during navigation. The use of tactile feedback devices was not really an option, as the representation of the human body is only partial in the model, and tactile feedback devices – even for the hand – still do not give satisfactory services. Therefore, the use of different modalities (like an auditory display), as suggested by [29], p. 48 could be a solution, but these ideas should be experimentally validated for the actual application.

5. Immersion in virtual reality literature[5]

This chapter gives an overview of how virtual reality (VR) literature and related sources describe the notion of immersion, its sub-types and similar terms. Two literature reviews have been conducted: one was a non-systematic review, the other a tertiary review (i.e. a review of reviews) using systematic methods. We first discuss the non-systematic review [295].

There are two main schools giving definitions for immersion in VR, and their definitions are quite different. One school, associated with B. Witmer and M. Singer, regards immersion as a psychological state of the user. The other school, based on the works of M. Slater, sees immersion as an objective characteristic of a VR system. Some authors call the first type of immersion "psychological immersion" and the second type "technological immersion". The authors of the present paper bring up the possibility of using alternative terms: "immersedness" for the immersion concept of the first school, and, for that of the second school, "immersingness" (denoting the actively immersing quality of the system) and "immersing" when referring to the act of immersing the user. The term "immersedness" is unambiguous, contrary to the term "psychological immersion" which could also mean a system characteristic when the system is created with the ability to try to manipulate the user with psychological means.

[5] This chapter is based on a conference paper [295], ©2020 IEEE, and a tertiary literature review [297].

5.1. Introduction

Virtual reality (VR) researchers, designers, developers, educators, evaluators and consumers often talk about “immersion” and “immersive” VR applications. E.g. in the 10th IEEE International Conference on Cognitive Infocommunications in 2019, there were 16 papers in which such terms occured [13] [38] [42] [48] [57] [58] [67] [68] [79] [80] [90] [143] [157] [186] [255] [262]. (Papers which contained “immersion”, “immerse”, “immersed”, “immersive”, “immersiveness” or “immersivity” were counted in.) But do they always mean the same thing by the term “immersion”? It is worthwhile to look into the concept more closely.

The current Merriam-Webster dictionary[6] defines the verb *immerse* as

“1: to plunge into something that surrounds or covers
especially: to plunge or dip into a fluid

2: engross, absorb
// completely immersed in his work

3: to baptize by immersion”

and notes that the first known use of the verb is from the 15th century.

(The Cambridge Dictionary[7] lists two definitions similar to the first two given above.)

[6] https://www.merriam-webster.com/dictionary/immersing
[7] https://dictionary.cambridge.org/dictionary/english/immerse

The first meaning listed above relates to a physical experience, the second meaning is a metaphorical use of the first, the third one is derived from the first one and refers to a physical experience (albeit with a spiritual aspect).

In the context of VR, the second, metaphorical meaning listed above is the relevant one.

J. Murray writes in her book [199], p. 99: *"Immersion is a metaphorical term derived from the physical experience of being submerged in water. We seek the same feeling from a psychologically immersive experience that we do from a plunge in the ocean or swimming pool: the sensation of being surrounded by a completely other reality, as different as water is from air, that takes over all of our attention, our whole perceptual apparatus. We enjoy the movement out of our familiar world, the feeling of alertness that comes from being in this new place, and the delight that comes from learning to move within it. Immersion can entail a mere flooding of the mind with sensation (...) But in a participatory medium, immersion implies learning to swim, to do the things that the new environment makes possible."*

She also states that in the fictional world we do not simply suspend our disbelief but we also create belief actively, and *"we use our intelligence to reinforce rather than to question the reality of the experience."* [199], p. 107]

5.2. Two glimpses into the history of VR and immersion

The ability to create a virtual world and getting immersed in it is deeply rooted in human nature and it has preceded computers by at least tens of thousands of

years. Prehistoric cave paintings are proofs of this. These paintings can be regarded as media with the purpose of storytelling, realized with pigment on stone [241], p. 5. D. Mioduser writes about these paintings: *"The representations are a human creation, but their presentation on the walls of the cave evoked a complex relationship between the creator and his creation with regard to the represented reality."* [196], p. 174.

Dated considerably closer to our age, but still long before computers existed, ancient Greek literature can be regarded as another case: The authors of [19] bring up Homer as an example, and claim that Homer intends to immerse his audience mentally, with various means [19], p. 35. They name three narrative strategies in his works: spatio-temporal immersion, identification, and the covert narrator [19], p. 38. Among the means of spatio-temporal immersion, they list epithets, similes etc. *"The texts should feed the reader with enough spatial and perceptual clues in order to create a detailed («vivid») mental «picture» of the storyworld."* [19], p. 39. Of identification, they write that Homer provides the readers/listeners *"with a viewpoint within the narrated scene, encouraging them to identify with that viewpoint."* [19], p. 41. The narrator should be as invisible as possible, Homer's works contain only two brief Muse invocations and a few narratorial comments [19], pp. 44–45.

5.3. Immersion as a transmedial phenomenon

Immersion is transmedial, i.e. not connected to a specific medium. The two historical examples mentioned in the previous section concern paintings and literature – two different forms of media.

In his book [104], O. Grau concentrates on visual art and mentions several examples of old and new media of illusion such as wall paintings of Pompeii, entire spaces of illusion like the Chambre du Cerf in the Papal Palace at Avignon (France), baroque ceiling panoramas, battle scene panoramas and their present-day successors: 360º images, and also films and VR. Each era strove to create maximum illusion with the technical means available at the time [104], p. 5.

F. Rose writes in his book that all good fiction is immersive. *"Books, movies, television, virtual worlds —century after century, we port our willing suspension of disbelief to whatever new and more immersive medium appears."* [232], p. 162.

According to the already quoted Murray: *"Once the illusory space is created, it has such psychological presence that it can almost divorce itself from the means of representation."* [199], p. 103.

M-L. Ryan has written extensively about the theories of immersion (mainly in the context of literature). She discusses emotional immersion, spatial immersion *("a sense of place and a model of space"*), spatio-temporal immersion (*"how to transport the reader onto the scene"*) and temporal immersion, which is *"the reader's desire for the knowledge that awaits her at the end of narrative time."* [4], pp. 121, 130, 140.

Narrative can greatly contribute to immersion in a story. K. Brooks discusses three attributes of narrative: time, context and participation. He notes that these *"map well from oral storytelling to new media and VR applications"* [144], p. 4.

As far as time is concerned, Brooks distinguishes two types of it in the narrative: presentation time and narrative time. Presentation time is *"a real-time experience encompassing the narrative experience"*, describing *"the period of the narrative; that is, the time during which one is in the movie theatre or the time during which one is participating in a VR experience. There is a direct and linear relationship to the clock in this aspect of time. It is the version of time that we experience in our lives on a day-by-day and minute-by-minute basis."* On the other hand, narrative time exists inside the narrative, it is *"a very flexible and fluid treatment of time (...). Its treatment and scale are completely dependent on the level of detail the storyteller needs to convey at any particular part of the story. (...) When we are drawn into narrative time so completely that we lose track of presentation time, we are immersed."* [144], pp. 4–5. (Cf. in music theatre, especially in opera, it can happen on the stage that a character receives a fatal wound or drinks lethal poison but still has time to sing a complete aria before he/she dies. Actually, the aria, which is usually about the character's feelings and thoughts of farewell, is in narrative time, not presentation time.)

Of context, Brooks says: *"Context is the space in which a story happens: setting of time and place, political and social conditions, individuals present, their goals and agendas, etc."* [144], p. 7. Similarly to the case of time, we can talk about presentation context and narrative context as well. The former relates to the physical environment where the narrative is experienced, the latter is the environment presented by the story [144], p. 7.

The third attribute is participation, it is desirable to enable people to participate physically and mentally as well [144], p. 10.

Brooks also emphasizes the importance of bringing all three attributes together [144], p. 13.

J. Loomis et al. state that throughout human history, artifacts, intended for the representation and recreation of external reality, have progressed from the literature, graphic arts and sculpture of earlier times to photos, films and audio recordings of the modern age. In recent times, computer technologies such as 3D graphics have been creating increasingly realistic artifacts, blurring the distinction between reality and representation [181], p. 557.

Ryan makes this bold statement: "*VR is not so much a medium in itself, as a technology for the synthesis of all media toward a total experience."* [235], p. 112.

5.4. Immersion in VR literature

In VR literature, there are two main schools defining the notion of immersion, and their definitions are quite different.

The first school is most often associated with the names of B. Witmer and M. Singer and regards immersion as a psychological state of the user. The second school is based on the works of M. Slater who sees immersion as an objective characteristic of a VR system.

The first school is in tune with the sources quoted in the previous sections and bases the definition of immersion on the well-known paper [286]. In this paper,

the authors define three interrelated terms: presence, involvement and immersion.

They define **presence** as "*the subjective experience of being in one place or environment, even when one is physically situated in another".* They emphasize that individuals can experience aspects of the virtual environment and events in their real environment at the same time [286], p. 225.

They give the following definition for **involvement**: *"Involvement is a psychological state experienced as a consequence of focusing one's energy and attention on a coherent set of stimuli or meaningfully related activities and events."* Becoming more involved increases the users' sense of presence [286], p. 227.

Finally, the authors define **immersion** as *"a psychological state characterized by perceiving oneself to be enveloped by, included in, and interacting with an environment that provides a continuous stream of stimuli and experiences. A VE*[8] *that produces a greater sense of immersion will produce higher levels of presence. Factors that affect immersion include isolation from the physical environment, perception of self-inclusion in the VE, natural modes of interaction and control, and perception of self-movement."* They name the helmet-mounted display (HMD) – nowadays rather called head-mounted display – as a typical device providing the above mentioned isolation. They also underline the importance of interaction: *"When users interact naturally with a VE, able to both affect and be affected by the VE stimuli, they become more immersed in that environment. Perceiving oneself as moving inside a simulated*

[8] Virtual Environment

environment or directly interacting with other entities in that environment will also increase one's sense of being immersed. Immersing people in a simulated environment is what VEs are designed to do, and that is why VEs have the potential to produce presence. (...) In our view, immersion, like involvement and presence, is something the individual experiences." [286], p. 227.

S. Agrawal et al. essentially adopt the views of the first school. They propose the following definition: *"Immersion is a phenomenon experienced by an individual when they are in a state of deep mental involvement in which their cognitive processes (with or without sensory stimulation) cause a shift in their attentional state such that one may experience disassociation from the awareness of the physical world."* [11], p. 5. They add: *"The three recognized reasons which can lead (independently or along with other reasons) to psychological immersion are the subjective sense of being surrounded or experiencing multisensory stimulation, absorption in the narrative or the depiction of the narrative, and absorption when facing strategic or tactical challenges."* [11], p. 5.

The authors also mention the term *transportation*, which, according to them, *"can be viewed as immersion which is fundamentally focused on the narrative"* [11], p. 6. Another term they bring up is *envelopment*, which is often used in the context of spatial audio. According to them, the difference between envelopment and immersion is that envelopment is perceptual while immersion is cognitive [11], p. 7.

Also in the vein of Witmer and Singer, E. Adams and A. Rollings state: *"Immersion is the feeling of being submerged in a form of entertainment, or*

rather, being unaware that you are experiencing an artificial world." [8], p. 25. They identify three sub-types of immersion in the context of game playing: tactical, strategic and narrative immersion. Tactical immersion is also known as the "Tetris trance". The game goes at such a speed that the player cannot think of anything else, he/she just wants to survive. The game typically offers small challenges to the player, one after the other. These challenges are quite alike. When the player is in strategic immersion, then he/she plans ahead in order to win the game, focusing on optimization (but not on the story or the characters). The rules of the game should be clear to him/her, in order to facilitate planning. There should not be too many unpredictable elements in the game. When in narrative immersion, the player feels being inside the story of the game. Good books or films can evoke the same feeling, but in VR games the player can also participate actively in the plot [8], p. 26.

E. Brown and P. Cairns have interviewed gamers about their gaming experiences and identified three levels of immersion: engagement, engrossment and total immersion [55].

L. Ermi and F. Mäyrä [84] also follow Witmer's and Singer's line. They have set up the SCI Gameplay Experience Model, based on their experimental research of game playing. They distinguish three dimensions of a gameplay experience: sensory immersion (S) related to the audiovisual execution of the game; challenge-based immersion (C) where challenges in the game can be related to motor skills and/or mental skills; and imaginative immersion (I) which is is the area in which the game offers the player a chance to use his/her imagination, empathize with the characters etc.

D. Arsenault [73] has suggested calling the imaginative immersion term of Ermi and Mäyrä fictional immersion instead, saying that we can be immersed in a story without exercising our imagination. He also suggests to use the term systemic immersion instead of challenge-based immersion, because the game player adopts the game's system i.e. the game's rules, and forgets about the rules of the real world, and also because we can be immersed in a system without necessarily being challenged by it.

Some literature mentioning immersion discuss the notions of user experience (UX) and flow as well:

According to D. Janssen et al.: *"The users' mental experiences in a VR environment are generally summarized by the term 'user experience' (UX), which can further be subdivided into certain theoretical constructs like immersion, presence and flow (...) Flow is defined as a reflection-free merging in smooth ongoing activities that have been under control despite high strain (...)."* They also say that in a state of flow the person's requirements and competences are balanced [136], p. 21.

The authors of [81] regard flow as a sort of psychological absorption and underline that flow is always enjoyable [81], p. 19.

C. Jennett et al. opine that immersion is a precursor for flow [138], p. 6.

D. Weibel and B. Wissmath have empirically examined the relationship between (spatial) presence and flow in the context of various computer games. They regard both flow and presence as immersive experiences. They have found that *"presence and flow are distinct constructs, which do hardly share common variance"* and come to the conclusion that while presence refers to the

feeling of being there in the mediated world, flow refers to being involved in the gaming action. Their analyses have shown that flow and presence are influenced by motivation and immersive tendency [281], p. 1.

Flow is also discussed by Agrawal et al., who remark that the literature on the experience of flow is inconsistent [11], p. 6].

Another related concept, *cognitive absorption* is also worthy of mention. R. Agarwal and E. Karahana [10] have defined it as a state of deep involvement with software. According to them, it is exhibited through the following five dimensions: temporal dissociation (the inability to register the passage of time while engaged in interaction), focused immersion (the experience of total engagement where other attentional demands are, in essence, ignored), heightened enjoyment (capturing the pleasurable aspects of the interaction), control (representing the user's perception of being in charge of the interaction) and curiosity. (A review of cognitive absorption literature can be found in [236].)

According to the definition variations of the first school quoted and discussed above, immersion is a psychological state of the user. The second school, M. Slater and his co-authors and followers (e.g. [2], [71]) define immersion quite differently. They are of the view that immersion is an objective characteristic of a VR system (but they define presence similarly to the first school). They call immersion *"a quantifiable description of a technology. It includes the extent to which the computer displays are extensive, surrounding, inclusive, vivid and matching."* They state that the more senses it serves, the more extensive a display is. The surrounding quality means realistic spatial distribution of

sensory signals (including visual stereopsis and spatial audio). Inclusiveness means shutting out sensory data coming from physical reality. Vividness depends on the characteristics of the displays, including richness, information content, resolution. They also describe what they mean by the term "matching": *"(...) immersion requires that there is* ***match*** *between the participant's proprioceptive feedback about body movements, and the information generated on the displays."* Turning one's head should cause a congruous change in the visual display and – where applicable – in the auditory display as well. Body tracking (at least head tracking) is mandatory for matching [251], pp. 164–165].

They state that immersion also requires a virtual representation of the participant in the VE. This representation is called the virtual body. Interaction, which is the ability to modify the virtual world, is a dimension of immersion [251], p. 165].

They outline the difference between (their definition of) immersion and presence. According to them, immersion is *"an objective description of what any particular system does provide. Presence is a state of consciousness, the (psychological) sense of being in the virtual environment, and corresponding modes of behaviour. (...) Behaviours in the VE should be consistent with behaviours that would have occurred in everyday reality in similar circumstances."* [251], p. 165. In another paper, Slater says: *"Presence is a human reaction to immersion."* [193], p. 2.

Slater states in a third paper that he defined the term immersion *"to mean the extent to which the actual system delivers a surrounding environment, one which shuts out sensations from the 'real world', which accommodates many*

sensory modalities, has rich representational capability, and so on (...).” He distinguishes his meaning of the word from the other definition by calling it *“system immersion”* vs. *“immersive response”* [250], pp. 1–2.

The definition by Biocca and Delaney is also in Slater's vein: *“Immersive is a term that refers to the degree to which a virtual environment submerges the perceptual system of the user in computer-generated stimuli.”* [2], p. 57. M. Lombard and T. Ditton call the immersion defined by Biocca and Delaney *“perceptual immersion”* [180].

To distinguish the definitions by the two schools from each other, some authors use the terms *“psychological immersion”* and *“technological immersion”* (e.g. [20], [81]). Other authors adopt the term “psychological immersion”, but prefer the expressions *“perceptual immersion”* or *“sensory immersion”* to “technological immersion” [11].

Sherman and Craig call the two types of immersion *“mental immersion”* and *“physical immersion”* [241], p. 10.

Agrawal et al. criticize Slater’s definition by saying that it *“implies that increasing the number of channels and loudspeakers augments immersiveness, irrespective of the content, context, and individual preferences.”* [11], p. 4. They state that immersion as defined by Slater is a facilitator for “psychological immersion”. They accept the views of Witmer and Singer on immersion, but they also introduce the concept of *“immersive potential”* which is the system's or content's potential to elicit immersion [11], p. 5. This is close to Slater's concept of immersion.

R. Skarbez et al. try to bridge the gap between the two schools: *"Slater's immersion is what makes it possible to experience Witmer and Singer's immersion"* [248], p. 96:3.

Actually, the two schools see the term immersion from two different perspectives (see Figure 5.1). The first school takes the **user's view: the user immerses himself/herself in the VE and/or lets the VE immerse him/her**. The second school takes the **creator's view: the system, created by the designers and developers, immerses the user**.

The following two simple drawings illustrate the difference between the two concepts of immersion:

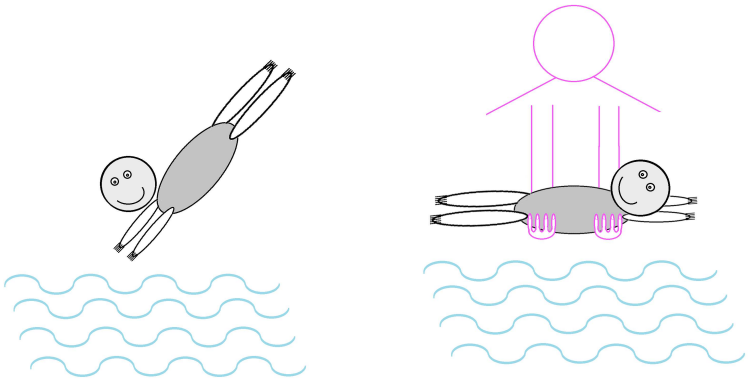

Fig. 5.1 The two concepts of immersion

It would be advantageous to distinguish these two concepts of immersion with distinct, self-explanatory terms. We propose to use the term pair **"immersedness" – "immersingness"**. "Immersedness" can be used for the immersion concept of the first school, denoting a psychological state. The second type of immersion could be called "immersing" for the act of immersing the user, and "immersingness" referring to the actively immersing quality of the

VE. As an alternative to the latter, **"immersiveness"** could also be used. The term "immersingness" underlines the directionality of this type of immersion a bit more than "immersiveness" does: it is the system which immerses the user, it is the system that has the actively immersing quality.

The term *immersedness*, for the immersion concept of the first school, is totally unambiguous, while "psychological immersion" or "mental immersion" can sometimes be taken as a characteristic of the system, i.e. *immersingness*, when the system tries to influence the user in a psychological way. An example of such manipulation could be to shape the virtual world in a way which matches the previously obtained characteristics, preferences of the user.

The word "immersedness" actually sporadically appears in scholarly writings. It is used in mathematics, e.g. [9] but the mathematical term has nothing to do with psychology or VR. One publication in the field of visual arts has been found: [155] gives an overview of the development of technologically mediated experiences of nature, and in the context of panoramas (large 360 degree paintings ideally covering the viewer's entire field of view, cf. [104]) mentions the various visual effects applied *"to heighten the feeling of realism and immersedness"* [155], p. 5. There is also one paper which uses the word in the context of bodily movements as gestures, and it states: *"How strong the bond towards this natural medium – the bodily self – is can be the degree of natural immersedness inside our bodies."* [100], p. 101. In all non-mathematical uses, the term clearly refers to a psychological state.

The word "immersingness" occurs only in a posting by an online graphics artist (*https://www.deviantart.com/itswalky/art/Some-SP-background-sets-*

199472245). The word is used rather colloquially but it seems to refer to the immersive ability of a virtual space.

5.5. A tertiary literature review on immersion

We have also performed a tertiary literature review (a review of reviews) on the definition of immersion. From this review it has became even clearer that immersion is a multifaceted phenomenon, and it is unlikely that a single and compact definition will ever be globally accepted. We have only chosen literary sources which give at least 8 definitions or definition-like descriptions of immersion. These sources hardly quote each other, they do not form any clusters. We are not trying trying to decide which definition is "the real one". Some articles only quote other sources, and the definitions obtained by them are quite diverse, but there are also articles which, on the basis of the definitions they found (and were not satisfied with), try to synthesize new definitions. The following new (synthesized) definitions cam be found in these sources:

Noirin Curran, who has probably provided the most detailed, in-depth analysis of game immersion, proposes the following definition [299]:

"Immersion is a subjective state of intense involvement in an imaginary setting, where an individual may either identify with or adopt a character or a setting (or both). If the individual identifies with a character they may adopt some, or all, of the character's attributes. If an individual identifies with a setting some or all of the demands of the setting become the attentional focus. An immersive response can vary in strength and duration, at its most extreme, causing loss of

a sense of self or reality and feeling as if one is the character or is 'in' the setting."

Zhang Chenyan [300] has constructed the following definition of immersion "in the virtual reality that fits into the QoE assessment paradigm" [where QoE means quality of experience]: *"Immersion in a virtual environment is a technology-mediated illusion that, through mimetic system offering priming stimuli and cues, engulfs one's senses and leads to the alignment of one's attentional focus to a synthetic yet perceptually authentic reality, by taking the visuo-spatial and emotional perspectives of the virtual agent(s), depending on one's imaginative facilities and mental dispositions and tendencies."* Zhang [300] also defines immersion (in a similar vein) as *"the sensory and perceptual experience of being surrounded by an environment perceived by the user as the real and prominent one: this artificial world is able to engage the user cognitively, emotionally, and physically, suspending attention from the concrete world",*

Agrawal et al [11] propose the following definition for immersion: *"Immersion is a phenomenon experienced by an individual when they are in a state of deep mental involvement in which their cognitive processes (with or without sensory stimulation) cause a shift in their attentional state such that one may experience disassociation from the awareness of the physical world."*

Jan-Noël Thon [298] mentions subtypes of immersion but does not define the overall concept.

A common feature in all these definitions is that immersion is regarded as a subjective experience,

5.6. Conclusion

In this chapter, the two widespread and quite different definitions of immersion have been reviewed, and various related definitions have been discussed, on the basis of two literature reviews. A proposal has been made for using a pair of unambiguous terms to distinguish between the two immersion concepts. From modern reviews it seems that nowadays immersion is regarded as a subjective experience and not as an objective property of a system.

6. Web search of software developers — Features and tips[9]

6.1. Introduction

Software developers (people developing software) frequently search on the web to support their software development activity. This chapter is based on the literature and also on the author's long practice as a software developer, discussing characteristics of developers' web searches and giving a few recommendations and tips, especially for complex, exploratory searches. As far as the author knows, no such summary combined with tips, directed at software developers, has been published before.

The efficiency and the outcome of ~~these~~ developers' searches — such as finding an appropriate software library or suitable documentation or usable example code — may fundamentally affect the success of their work.

There exist several ~~useful~~ resources on how to search on the web in general: books ([77] [234] [233]), articles ([47]), and online resources.[10]

However, according to the literature, the web search of software developers have some specificities. Knowing how their peers search, and being acquainted with some tips can make developers perform their own web searches more consciously and efficiently.

---

[9] This chapter is loosely based on the journal article [263].

[10] http://www.rba.co.uk/search, http://searchresearch1.blogspot.com/, https://docs.google.com/document/d/1ydVaJJeL1EYbWtlfj9TPfBTE5IBADkQfZrQaBZxqXGs, https://sites.google.com/site/resourcesandsearchstrategies/, https://www.google.com/insidesearch/searcheducation/index.html

(By searches of software developers I mean their searches performed in the capacity of software developers.)

6.2. Features of developers' web searches

This section summarizes the most important findings in the literature.

[290] studied professional developers' web searches, with multiple methods (search logs analysis, interviews with developers, surveys). The most common search tasks were identified as searching for explanations for unknown terminologies, explanations for exceptions/error messages, reusable code snippets, solutions to common programming bugs, and suitable third-party libraries/services. The most difficult search tasks were searching for solutions to performance bugs, solutions to multi-threading bugs, public datasets to test newly developed algorithms or systems, reusable code snippets, best industrial practices, database optimization solutions, solutions to security bugs, and solutions to software configuration bugs.

[227] analyzed the search log of a major general purpose search engine and identified software engineering related queries with the aid of a machine learning based classifier. 2.61% of all the web search sessions were software engineering related. (While the term "software engineering related queries" perhaps does not absolutely correspond to "developer queries," we can disregard the difference for our purposes.) Software engineering search sessions were shorter than other sessions, had a higher word count, and had a higher rate of term additions and removals in query reformulations. Software engineering

search queries had a lower click rate (i.e., the rate of clicking to a web page after the query) than other search queries. (The authors opine that this can be attributed to software engineering related search tasks being more difficult than other search tasks. I must remark, however, that we do not know the proportion of cases when the result preview already gave the answer and there was no need for clicking.) Dwell time, the amount of time spent by users on the clicked documents (calculated as the time between the click and the next seen click or query on the search engine) was lower than with other queries. [The authors, on the basis of search literature, state that longer dwell time correlates with success in finding the required information and they come to the conclusion that therefore software engineering queries are less effective than other queries. I think that this may be a hasty conclusion, as people issuing software engineering queries may be more effective and thus faster in evaluating search results than the average population, and software engineering related web pages may be more to the point, more easily understandable than other web pages. Also, the methodological issue arises that the calculation of dwell time may be questionable if software engineering people tend more to click multiple candidate results in browser tabs in rapid succession and evaluate them one-by-one, such behavior has been described (e.g., in [52], p. 3).]

The most researched search type among developers' web searches is search for code, i.e., software. They Developers look for ready-to-use software, linkable/includable code libraries, code examples, code samples, coding tips, bug fixing tips, also algorithms from which code can be written.

The most important findings on code search:

Developers prefer general-purpose search engines over any other information gathering means, specialized code search engines are not used too often ([142] [247] [129]), [245] found that subjects using Google (vs. other search engines) in a code search experiment employed a higher average number of terms per query, a higher average clickthrough rate, and more time overall, and received more hits which were perceived relevant. The researchers opined that the success of Google was probably due to the high number and variety of pages that it indexed, and also because participants in the experiment were more familiar with Google than with the other engines.

A study analyzing the log of a code search engine has come to the conclusion that "users who find code search engines usable are those who already know to a high level of specificity what to look for" ([24], p. 1).

Sometimes code searches are also performed in code repositories ([142], [245], [247]).

As far as source code search is concerned, [247] states that, in terms of ~~by~~ motivation, there are two major search archetypes. At one end of the spectrum is the search for source code for reuse as-is, when the developer does not want to modify the code at all. The other archetype is search for code as a reference example, when the developer only wants to use the knowledge behind the code. Between these archetypes are the searches which are some mixes of the two. A search process may even start at one end of the spectrum and may move toward the other end. A minor archetype has also been identified: search for information about bugs or defects, which is characterized by long queries ([226] p. 8), as full error messages are often submitted as queries.

[23] is about a lab study which graduate students searched for code to resolve programming tasks in an unfamiliar programming language. Search success was investigated. Taking only clicking to a search result into account as proof of search success is inadequate, as inspecting a result is not a guarantee for success yet, and also the correct answer to a search query may appear in the preview in the hit list, not necessitating clicking. Therefore, participants were surveyed during their search activity (when a search tab was closed and when there were signs that a query was reformulated) about their success. A few findings:

Participants most frequently searched for example code and ways to resolve bugs or errors. Learners formulated much more verbose queries than professionals did. Most participants borrowed terms from programming languages with which they were familiar, these queries were more successful than the average query. Successful searches used natural language phrases. [As to the possible reason: Gallardo-Valencia and Sim [94] opined earlier that web pages with code snippets, examples, usually contain a lot of explanations in natural language, and the general purpose search engines pick up these.] The majority of "How to" queries were successful, and "(…) the most successful searches are those that consult resources with examples, such as documentation and official/third-party tutorials." ([23], pp. 2,5,7,10).

Li et al. [169] conducted an experiment with subjects having to correct buggy Python codes with the aid of searching Stack Overflow through Google. Subjects were categorized as Novices, Non-Python-familiar-experts and Python-familiar-experts. Experts tended to click more search results than novices and had higher success rate. Python experts (the most successful of the three categories) clicked on more links that were further down the result list.

Rahman et al. [226] compared code search and other search activities of the same developers. In comparison to other searches, code searches contained more words, the number of queries per search session was much higher, query modifications occurred more often, more websites were visited in a session, more time was spent on the search. This implies that searching for code is harder, and also that developers do not give it up easily.

6.3. Exploratory searches

Martie [188], writing about code search, emphasizes the iterative nature of search, especially when the search is not for one specific piece of code, the developer is not entirely sure at the beginning what he/she is searching for.

“In such cases, search is more of an exploratory process where multiple queries are issued by the programmer so that they can discover results to learn and gain ideas from or to learn more about what code they might want in the search engine.” ([188], p. 61).

Exploratory searches may take a long time, even months ([282], p. 6) and they are hard to study.

Let me cite two search examples from my own practice as a developer. Both involved (or at least started with) exploratory searches, but while one was a relative routine task, the second one was, in my own practice as a developer, the longest and toughest search for code, taking months.

In the first case, in the project described in [296], despite the availability of vendor support for the LabVIEW graphical programming environment used in the project, web searches for concepts and for code were also performed. One of my tasks was to use a specific device, a multichannel analyzer (MCA), in the project. As my search diary (which I saved) shows, I performed multiple searches, first to understand how to handle a multichannel analyzer (which instrument I never used before) in a LabVIEW project (the LabVIEW environment was also new to me). One of my search queries was "*LabVIEW Canberra MCA*" (the firm Canberra was the manufacturer of the MCA). My queries were exploratory at first but grew more and more specific as I gained more and more knowledge, which then enabled me to handle the multichannel analyzer in the project.

The task in the second project was much more complex. I wanted a detailed, realistic virtual hand model in a particular game engine, driven with input data from a hand movement detector, for a virtual reality project. After a while it became clear that no ready-made solution existed, I had to create the hand model myself. Luckily, I found a hand mesh (geometric model of a hand) which looked suitable, so I "only" had to integrate it with an armature (a skeleton representation of the bones of the hand in the game engine) and drive the armature's bones with bone position and orientation data from the detector. Apparently, no one had done this before for that particular game engine, so I had to learn a lot about the extremely complicated armature mechanism of the game engine. Certain features worked correctly only under special and undocumented circumstances. My search process was heavily intertwined with development efforts. After each failure I searched further and tried out the new

findings. I maintained a search and development diary. The subsequent mining of this diary has revealed that I recorded 853 distinct URLs worthy of noting and these belonged to 174 different websites. 26% of the recorded hits contained code (mostly code snippets) which I thought worthy of copying or downloading. At the time of writing the diary I had no intention to make any kind of statistics from it, so the data presented here contain no such bias. Table 6.1 shows how the top 4 websites were represented among the URLs. *blenderartists.org* is an independent user site (with forums) dedicated to the *Blender* (https://scicrunch.org/resolver/RRID:SCR_008606) suite which included the game engine, *blender.org* is the official site of *Blender*, *stackexchange.com* is a network of Q&A (question-and-answer) sites on diverse topics, *leapmotion.com* is the official site of the hand movement detector. The two official sites both contain forums as well. 8 further websites held 1–2% of the URLs each. All other sites were below 1%. 119 sites were represented with only one URL. These results show how important the forums have become, and how diverse the URLs can be.

One day, after I applied a trick found in the blog of a small company, the hand model started working correctly. It was published in [37].[11]

I mostly used Google for search (Bing and Yandex did not yield additional results), set the number of hits to be displayed on one page to the maximum (to speed up evaluation). When I thought necessary, I scanned several hundred hits in the hit list. When there were several promising hits, I opened them in new tabs one-by-one, then evaluated them one-by-one, relatively quickly.

[11] See demo at https://youtu.be/uEhEZ1MIul4

Website	Among all URLs (847 total)	Among important URLS (134 total)	Among code-containing URLs (218 otal)
Blenderartists.org	30%	33%	33%
Blender.org	16%	12%	5%
Stackexchange.com	9%	15%	14%
Leapmotion.com	4%	6%	6%
All 4 sites together	59%	66%	58%

Table 6.1. How the top 4 websites were represented among the recorded URLs.

I often found useful tips in forums and blogs, and it sometimes paid off searching for other posts of authors whose posts I found informative.

My search diary was in Word format, I copied into it relevant URLs, explanatory texts, interesting pieces of code which I sometimes annotated, figures, and even screenshots of my program runs as well. I put exclamation marks before the recorded address of web pages I thought important, I also used font properties (size, color, style) to mark different types of information. When a part of the text (and the idea behind it) proved to be a dead-end street, I changed its font to a smaller one and crossed it out, thus invalidated it but still left it in the diary.

I believe that documenting the searches in the way described above highly contributed to the successful outcome of the search. Also, that I was persistent,

did not necessarily stop at 10, 20 or even 100 hits, and assessed the promising hits fast.

Why could the examination of many hits prove useful, especially with exploratory searches? According to [282], p. 15:

“Exploratory searches may be more concerned with recall (maximizing the number of possibly relevant objects that are retrieved) than precision (minimizing the number of possibly irrelevant objects that are retrieved). Thus, they are not well supported by today’s Web search engines that are highly tuned toward precision in the first page of results.”

Pariser ([217], p. 59) has put this in a simpler way:

“Google is great at helping us find what we know we want, but not at finding what we don’t know we want.”

However, it can happen that the lack of relevant results among the first hits simply means that the query is misguided, it is better to abandon or at least modify the query. This is always up to the searcher’s judgment (which usually develops with practice).

Also, we should keep in mind that Google does not display more than 1,000 results for any query ([212], p. 75; [234], p.151).

To help users in their searches (and to profit from offering products to users), search engines often create profiles of users. For this, they use their search history, geographical location and other data such as what commercial products they bought in the past, what information they have given about themselves. Thus search engines are able to provide search results more specific and

relevant to the user, but, with this, they put the user into a sort of "bubble." This feature (and also that the hits frequently clicked by other users will probably be ranked higher, with which they will get even more clicks) can be counterproductive in exploratory search, when the solution often does not lie along the well-trodden paths.

One, time-consuming, strategy to counteract this effect is the above mentioned examination of a high number of search hits. We can also try countermeasures to depersonalize, to "debubble" the search.[12]

6.4. A few more web search tips for developers

If you want to use search engines efficiently, you should know their advanced search options and characteristics. Not only Boolean operators can be put to use. For software component search, the *filetype* operator (Google, Bing, Yandex) can be of help (it recognizes the file types of several programming languages) and also Google's and Bing's wildcard (*) (e.g., when looking for functions/methods with known parameter types, wildcards can serve as substitutes for the unknown parameter names) ([212], p. 76). With the *verbatim* option Google will not look for synonyms and variations of the search terms, this can be useful e.g., when looking for specific function/method names in source code search.

You can learn a lot about the capabilities and characteristics of search engines from search experts.[13]

[12] Depersonalization tips for Google: https://nostop.net/depersonalize-google-search.html
[13] See the Introduction for such resources.

It can pay off to use search engines other than Google, you may get different results.[14]

A potentially useful type of information source is scholarly literature, both in the application domain for which you develop your software, and in the software engineering field. Scholarly articles, books, studies, research reports, dissertations may contain code snippets, algorithms, coding tips, references to ready-to-use programs and to components. Sometimes authors make their own software available to the public. Many scholarly sources can be found through general purpose search engines, but search engines specializing in scholarly literature can also be of great help. Of these scholarly search engines, Google Scholar can be recommended in the first place, because it has the widest coverage of sources. There are also topical search sites for various domains, and the sites of publishers of scholarly journals and books are also valuable resources. Publications usually contain references, some of these may be worth exploring. A publication of interest may be cited in other sources which could in turn contain further useful information for the developer. Such citation searching is supported e.g. by Google Scholar.

Social media, such as Q&A sites (notably, Stack Overflow), forums, blogs, microblogs (Twitter) collaboratory code repositories (SourceForge, GitHub etc.) can also be great sources of information. In code search, these can also help with the evaluation of code candidates, as they often show the community's opinion of the code.

[14] For alternative search engines, see e.g., https://www.searchenginejournal.com/alternative-search-engines/271409/

6.5. Discussion

On the basis of the available literature, I have listed some features of developers' web searches, and a few tips for making searches more efficient, derived from the literature and my own experiences.

Where are we headed? This greatly depends on the evolution of search engines. The current practice is to use mostly general-purpose search engines even for code search. The formerly popular Google Code Search was shut down years ago[15] and several other code search engines have become obsolete, too.[16] Perhaps the ever-increasing quantity of available code has become too much for them. It is in principle possible that a search tool specializing in code search emerges and becomes dominant, but it has not happened so far.

Possibly, general-purpose search engines might evolve into more efficient tools through addition of software-developer-friendly features. However, as mentioned above, only less than 3% of web search sessions are software engineering related, so this is probably not high on the priority list of search engine companies. Until then (and even after, in the case of exploratory searches), we must resort to search tricks to make our search efficient.

---

15 http://googleblog.blogspot.com/2011/10/fall-sweep.html

16 In 2018, Code Sample Answer (https://blogs.bing.com/searchquality-insights/2018-07/Intelligent-search-Coding-answers-at-yourfingertips/) was introduced in Bing, but now the code samples are no longer available in the search results.

7. A systematic mapping study on how developers search for source code on the web

7.1 Introduction

As a continuation of my investigations into how developers search for code, I and my co-author have conducted a systematic mapping study on how developers search for source code on the web. On the basis of the 105 relevant retrieved papers, the study identifies the search tools most frequently utilized, the types and domain names of websites most often found, the factors and cues influencing selection of search hits, the ranks of selected search results, the search strategies and developer characteristics correlating with success, and the differences in the search processes of various developer groups. Additionally, two study types not mentioned in other systematic reviews have been identified: virtual lab study and search and development diary study.

With our systematic mapping study we wished to find papers related to web search for source code, this seems a broad enough area to have quite a number of published papers. Unlike most systematic reviews on code search, we do not focus on code search tools. Instead, we concentrate on the developer, the human who searches for source code online. Our approach is somewhat related to the discipline that is called behavioral software engineering[17].

[17] [167] defines behavioral software engineering as the study of cognitive, behavioral and social aspects of software engineering performed by individuals, groups or organizations

Our mapping study is evidence-oriented, we present all the relevant evidence found in the publications, regardless of whether the particular evidence is contained in one or multiple publications pertaining to the same investigation.
Relevance has been decided on the basis of the inclusion/exclusion criteria. However, redundant publications, solely containing evidence (answering research questions) which can also be found in other publications already in the corpus are listed separately as redundant. In case a paper had multiple redundant versions, only one version was kept, and the other versions were eliminated.
The present chapter is organized as follows. Section 7.2 defines and clarifies terms and concepts pertaining to the topic. Section 7.3 describes how the mapping study was planned and executed. Section 7.4 presents the research questions and the answers to them. Section 7.5 concludes the chapter.

The Appendices to the dissertation contain supplementary material for the documentation of the systematic mapping study: the study protocol, the inclusion/exclusion criteria, the initial set of papers, the list of primary and redundant studies, the list of the used search tools, the search queries and the exception list for the duplicate eliminator.

7.2 Background

This section reviews the terminology and main concepts used throughout the paper.
By ‘developer’ we mean anyone who develops software.

7.2.1 Definition of source code

We slightly enhance the definition of source code originating from the standard [1] with emphasizing that source code is human-readable (which the standard does not explicitly state, but it can be found in other definitions, such as on the website of SUSE Linux[18]). Thus, throughout our systematic mapping study, we regard source code as *human-readable computer instructions and data definitions expressed in a form suitable for input to an assembler, compiler, or other translator*.

We also remark that the 'other translator' can be an interpreter as well.

A sub-type of source code that often appears in the literature is the *code snippet* (sometimes also called a block of code). A code snippet is a few lines of source code, often embedded in web pages such as the Q&A (question and answer) site Stack Overflow. Typically surrounded by textual explanations, a code snippet is a sort of code quotation, usually not a complete program or even a full function, cf. [269].

7.2.2 The notion of source code search/seeking

In her publication [93], Gallardo-Valencia has proposed "a five-stage model to differentiate the stages that could take place when developers look for source code on the Web." These stages include: (1) identify source code need, (2) select web resource, (3) translate need to Web resource, (4) evaluate results, and (5) use suitable results [93]. In other words: "Source code seeking on the Web involves the search, evaluation, and retrieval of source code from the Web, as

[18] https://www.suse.com/suse-defines/definition/source-code/

well as the application of the retrieved source code to solve a software development problem" [93]. Gallardo-Valencia states:

"Developers are looking for source code on the Web as a technique to write source code" [93]. This means that source code search/seeking can be regarded as a software development technique. Gallardo-Valencia emphasizes that the process is not strictly sequential, the developer may return to a previous stage or omit stages.

We adopt the above model for our literature review. We concentrate more on the second, third and fourth stages (and look for articles dealing with these), and less on the first stage, and even less on the fifth stage of the above model.

We do not include articles which investigate only the fourth and/or the fifth stage (evaluating search results / using the results) without dealing with any of the preceding stages.

We do not include articles which only indirectly investigate how developers use the retrieved code. By "indirect" investigation we mean mining software repositories and social sites for traces of copied software, etc., instead of directly investigating developers' practices. (However, we include articles on the mining of search logs, as these directly investigate the search behavior of developers, i.e. they analyze actual developer queries.)

As far as terminology is concerned, we do not differentiate so strictly between search and seeking as Sim and Gallardo-Valencia do.

7.2.3 Study types

Starting from the classification of study types found in [187] which distinguishes survey, search log analysis, field study, and lab study, we have

further refined and enhanced the classification, as outlined below, based on additional sources in the literature and our own findings and experiences.
In this mapping study, we distinguish the following study types:

• *Survey*. According to [82], surveys are used "to identify the characteristics of a broad population of individuals". We categorize surveys as questionnaire-based surveys and interview-based surveys by their (main) data collection method (see data collection methods later). Some investigations mix the two survey types, we call them combined surveys.

• *Field study*. For this type of study, we use the definition in [221]: "(...) we define field studies as observational studies of software developers, conducted during their daily work, with minimal or no interference to the developers' usual activity".

• *Lab study* and *virtual lab study*. A lab study, sometimes also called a lab experiment, is performed in vitro, i.e. in a laboratory under controlled conditions (as opposed to the in vivo method of field studies) [117]. In our systematic literature review, in addition to the 'classical' lab studies, a new type of lab study has emerged: the virtual lab study (it could also be called a remote lab study), in which the location of the participants is not restricted to the same place (the laboratory), they can be located at remote and geographically distributed places. This makes the location condition less controlled, but the other conditions usually resemble those of a classical lab study. This type of research bridges the distance gap: study participants may be located in their

home, workplace, in an internet café, etc., thousands of miles away from each other, they do not have to travel to the same lab for the study, thus, a more diverse population can be involved. This type of study has strongly come forward in the last 5 years, with the pandemic. So, we call classical lab studies lab study, and the remote ones virtual lab study. There are also mixed lab studies, in those cases we can classify them on the basis of
whether they are more classical or more virtual.

• *Focus group study*. '(...) focus groups occur when groups of people are brought together to focus on a particular issue (not just generate ideas). They also involve moderators to focus the group discussion and make sure that everyone has an opportunity to participate'. A focus group can also be virtual, i.e. members do not necessarily meet physically, see e.g. [30].

• *Log study*. By our definition, a log study (called search log analysis by [187]) is a study in which search logs are collected but users (participants) do not participate specifically in a lab study or field study, and an external search tool (such as a code search engine or a general purpose search engine or the search function of a Q&A site or repository) itself logs the search data. The search tool is accessible via a website (cf. this mapping study deals with web searches only). The purpose for which the users perform the searches is not predefined or controlled by the researchers. There are usually a high number of participants and they are at diverse locations. The participants typically cannot be identified individually (though it may be possible by IP address, or when they provide personal data themselves at registration, etc.).

We do not call it a log study when, within lab studies and field studies, another type of search log is collected, with different means: the participants' work environment (typically a browser) is instrumented with data logging capabilities (usually performed by a plugin), and collects the search data. (We note that screen recording may perform the same function, since search log info can be extracted from it, this occurs rarely [229].)

• *Search and development diary study*. Usually, diary analysis is just a part of field studies (sometimes also lab studies), but there 'diary' means 'work diary' (see in detail in Section 7.2.4) and not 'search and development diary' documenting the development process which includes web searches and usually spans a long time period. Unlike the work diary, the search and development diary is created by the developer for his/her own purposes and not because the researchers requested it. The content of this diary is not always arranged chronologically and does not necessarily record all (or perhaps any) search queries, only the usable results, and perhaps also some failures. It does not necessarily reveal minute details of the search process, but it may record URLs, code snippets, etc. Its format is not prescribed by the researchers, the developer does not necessarily know at the time of creation that the diary will later be used for research purposes. The format and content highly depend on the cognitive style of the developer, and also on the task context.

Such development diaries must be abundant in the developer community, but our mapping study has found only one study [263] where such a diary was actually analyzed. Also, in our research corpus, there are publications (related to each other) about opportunistic programming [52], [54] which mention keeping

development diaries: they state that successful opportunistic programmers document their process, instead of documenting their code, but no details are shared about these diaries.

Diary studies usually do not put any additional burden on the developer during the development process. The processing of such diaries may be enhanced with subsequent questioning of their creators.

7.2.4 Data collection methods and types of collected data

This categorization is partially based on [82]. We distinguish the following data collection methods:

• *Interview* (for surveys, lab and field studies) and discussions (in groups). In interviews, at least one researcher talks to at least one respondent [82]. The parties may meet in person or can talk with each other through an audio/video connection. An interview can be the main data collection method in an investigation, but it can also be a small part of a lab or field study (post-task/debriefing interviews are typical in such studies). The discussions in focus group studies are documented similarly. The collected data may be audio or video recordings or their transcription (there are automated tools for creating transcriptions from recordings, Otter.ai is mentioned in multiple studies), or researcher notes made during or after the interview.

• *Questionnaire*. Questionnaires are sets of questions administered in a written format. When part of a lab or field study, they are typically filled out before or after the development activity, but occasional use also occurs even during the

development process. They may be in paper form, but electronic form is more common nowadays.

• *Work diary*. “Work diaries require respondents to record various events that occur during the day. It may involve filling out a form at the end of the day, recording specific activities as they occur, or noting whatever the current task is at a pre-selected time” [82]. They may be in paper or electronic form. We remark that the work diary can be in the form of a questionnaire, we classify such questionnaires as work diary if questions fit the criteria above and the recorded data are drawn from the actual development process.

• *Search and development diary*. As already mentioned above, unlike the work diary, the search and development diary is created by the developer for his/her own purposes and may cover the whole development process. The developer usually does not know at the time of creation that the diary will later be used for research purposes. Not only the content, but also the format of these diaries can be quite diverse. In the literature, diaries made with e.g. the following tools are mentioned: Notepad or MS Word (to keep a record of visited or potentially interesting URLs etc.) [161]; files made by other note taking tools such as Workflowy, Notes, Evernote, Google Docs. Diary-like functions can also be fulfilled by bookmarks (stored by browsers), or even custom websites built by the developers [177]. The diary may also be in the form of a public blog. We believe that there is potential in processing such diaries, especially in cooperation with their authors.

• *Think-aloud protocol*. Researchers ask participants to think out loud while performing a task [82]. The data collected may be audio or video recording or its transcript, or researcher notes.

• *Observation*. The experimenter(s) observe(s) the participant(s) engaged in work, or specific experiment-related tasks [82]. The collected data are researchers' notes (electronic or paper form) which may be created simultaneously with the observation and/or afterwards.

• *Recording the search/development process on video/audio*. In our view, since the publication of [82], the video/audio recording (especially screen recording) technique has become so popular and easy to do, that it merits a separate category.

• *Instrumentation (other than video/audio recording)*. The developers' work environment is instrumented to record information automatically about the usage of tools. Logging of queries and website visits with a browser plugin is typical in this type of data collection, but more detailed recording of interaction data (such as clicks, scrolling, copy-paste) also occurs. Such data are usually collected in log files created by the plugin.

• *Query logs saved by online search tools* can also be obtained. This data collection method is distinct from the instrumentation of the developer's work environment. The logs in the category under discussion are collected by an

online search tool such as a general purpose search engine or a specialized code search engine or the query function of a Q&A site.

• *Auxiliary data collection* involves gathering the source code and / or the executable created by the developers (these are typically used to judge the success of the development activity), students' exam scores, notes or drawings made by the study participants, etc.
In principle it is also possible to collect biological parameters of the participants via biometric methods such as ECG, brain MRI etc., to characterize stress, brain activity, and so on, also eye tracking, and such methods are being used e.g. in software usability tests. However, in the relevant publications found by this mapping study we have not encountered actual use of such methods for investigation of web search for source code. (Instead, we have found instances of collecting cognitive load data by a questionnaire, the NASA-TLX survey [97] [279] which belongs to the Questionnaire category.)

7.3 The systematic mapping process

A systematic mapping study is a "broad review of primary studies in a specific topic area that aims to identify what evidence is available on the topic" [148]. Primary studies are the original research of their authors. Our goal with this systematic mapping study was systematic identification of primary studies investigating how software developers search for source code on the web to aid

their software development. We have used a broad selection of search sources and have aimed to include gray literature[19] as well.

We collected articles describing investigations into source code search practices of developers who utilize publicly available and widely used search tools on the web. We did not wish to take into account the use of experimental tools (even if they are available to the general public). This actually narrows the focus considerably: [175] states that the overwhelming majority of articles they found with their systematic review of articles on code search proposed new tools. Neither did we deal with offline code search or search in repositories not available to the general public.

Our investigations were directed at online searches by human subjects looking for source code for software development. This includes search log analyses, but does not include analyses of source code in repositories for traces of e.g. Stack Overflow usage, or for code clones. We excluded code clone detection and the search for signs of code plagiarism, as that kind of research is not directly related to web search for code for development. We included only articles in which the web searches under investigation were performed by developers (using various search tools), but we did not include articles investigating how search tools find code, if these searches are not performed by developers or the searches are not intended for software development. Neither did we deal with articles on malware detection, bug localization, code completion, and feature (concept) location.

Human-composed queries submitted to search engines/tools in order to obtain source code for development (at least when it can be assumed that the query is

[19] Gray literature in software engineering can be defined as any material about software engineering that is not formally peer reviewed nor formally published [98].

for obtaining source code for development) are within the scope of our investigation. For our research, we do not regard questions posted on Stack Overflow and similar social sites as (direct) searches for source code, even though they are often aimed at obtaining source code, indirectly, as the answers come from the community, after some time. However, queries for source code via Stack Overflow's search engine count as source code search, producing results promptly.

We have adopted a human-centered approach: our focus is on the user, i.e. the software developer searching for code, and not on creating code search tools. The practical importance of experimental code search tools is by far not proportional to their representation in scholarly literature. The overwhelming majority of such experimental tools never gain a wide user base, their code base is small and they are inadequately tested [175]. Thus, they are not subject to our literature review. Articles about new code search tools that were not well-known tools (see detailed criteria in Appendix B) were dismissed unless the use of the new tool by developers was tested together with that of well-known tools and the data thus obtained was relevant to the use of these well-known tools, or the paper contained a relevant survey or other investigation independent of the proposed new tool.

Our goal was to find as many relevant publications as (reasonably) possible, while undertaking the burden of having to eliminate a high number of irrelevant publications, i.e. we preferred recall to precision. To reach this goal, we used numerous search tools (including a general purpose search engine, in the hope of finding not only scholarly articles but also gray literature), which produce results in a wide variety of formats. We decided to create our own database

format and wrote a duplicate eliminator program which identified duplicates on the basis of their titles.

A consequence of using a high number of diverse search sources is that a lot of search tools differ from each other in their search capabilities (such as maximal number of search terms, use of wildcards, etc.) which have to be individually studied to formulate search queries tailored to the particular search tool. Tools usually restrict the length of queries and the maximal number of displayed results as well. We formulated queries in a way that no search hits were lost due to the above-mentioned limits. We do not know and have no control over the ranking of results in these tools, their ranking might not necessarily conform to the ranking which would be optimal for us. Therefore, we wanted to retrieve all the provided results, not losing any of them to the limit on the number of displayed hits. Because of this, and to stay within the query length limit, we had to issue a high number of queries which were specific (i.e. they contained several search phases instead of one or two). You might question the need for issuing so many queries (more than 100 on Google Scholar and Google) for a mapping study. However, with our queries we also wished to lay the foundations of a future systematic review, for which maximizing the number of retrieved relevant studies is a necessity.

Our queries yielded a high number of trivially irrelevant/off-topic publications. [148] recommends “maintaining a list of excluded papers, only after the totally irrelevant papers have been excluded”, because initial electronic searches result in a large number of totally irrelevant papers [148]. We have followed this recommendation.

We did not want to miss publications where the code-search-related part was not necessarily the main topic of the article. Therefore, we did not place special emphasis on article keywords.

7.3.1 Research questions

In the context of developers' source code search on the web, we have formulated the research questions (RQ-s) listed in Section 7.4. With these research questions we intended to tap knowledge of practical importance for education about source code search and for code search tool development.

7.3.2 Planning and executing the study

We have decided mainly to follow the methodology described in [148] and [147], we have used [218] as well, and we have taken into account the so-called Durham template[20] (a checklist for conducting mapping studies in software engineering, referenced by [218]).
We have divided the review protocol into phases. These are discussed in Appendix A.

7.4 Answers to the research questions

In this section, we answer the research questions on the basis of the extracted data. All research questions are in the context of web search for source code.

[20] Template for a Mapping Study Protocol, 2023
https://ebse.webspace.durham.ac.uk/wp-content/uploads/sites/49/2023/02/MappingStudyTemplateV2.pdf

Some publications are about multiple investigations. Furthermore, an investigation may be described in multiple publications that have not been dismissed as redundant because they also contain relevant data that are not present in the other publications. In the sections and tables below, we have taken care to list one investigation and its findings only once, regardless of how many relevant publications describe it. In rare cases, we list the same publication in different contexts for different research questions.

7.4.1 RQ1: How are the relevant publications distributed over the years by publication venue type?

We have established the following publication categories by publication venue and found the following number of publications for these categories (the total number of publications was 105):

- *article in journal listed in Scimago: 28*
- *article in other journal: 7*
- *conference / symposium paper published by IEEE or ACM: 37*
- *other conference / symposium paper: 3*
- *workshop paper published by IEEE / ACM: 3*
- *other workshop paper: 3*
- *dissertation / thesis (PhD, MSc, BSc): 12*
- *book chapter (here conference proceedings do not count as books): 5*
- *technical report or research note: 4*
- *other article (e.g. in a newspaper or website on technology, or never officially published paper): 3*

In establishing these categories, we have paid attention to include gray literature, as, in our experience, such items may contain scientifically sound evidence, although, for some reason, they have never been published as scientific literature. Naturally, scientific literature, especially rigorously peer-reviewed articles, has more credibility. (However, we have found one instance in which an article, published in a highly prestigious magazine ranked Q1, was later flagged as plagiarized, and we have excluded this article due to plagiarism.)

These data can be seen on a bar chart and a bubble chart (Figures 7.1. and 7.2.). The year is always the year of publication (not the date of the investigations, which is often hard to determine and might extend over years).

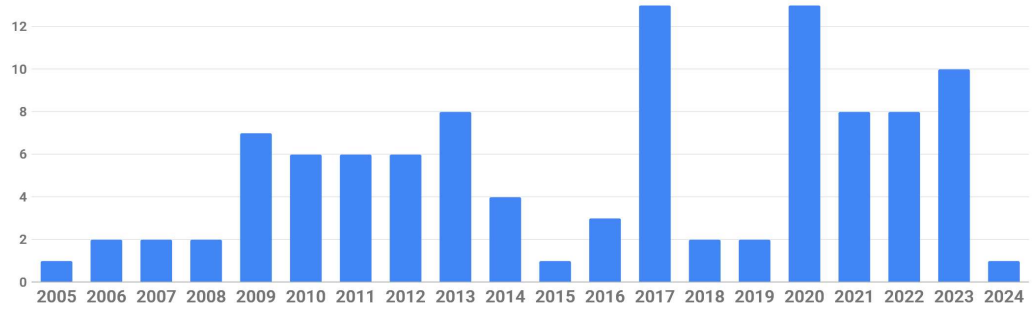

Fig. 7.1. Number of publications per publication year

As can be seen from these figures, the time range spans exactly 20 years. Even though no time limit was set for the searches, and Internet use started widely spreading in the 1990's, we have not found any relevant publication before 2005. Interest in this topic started rising in 2009. Although there are fluctuations in the distribution, the interest does not seem to decrease. It can be seen from the bubble chart that starting from 2020, the number of papers in journals listed

in Scimago, and also the number of IEEE / ACM conference publications have increased.

There are only two books which have been found pertinent: [246] with 4 chapters found relevant, and [254] with 1 relevant chapter.

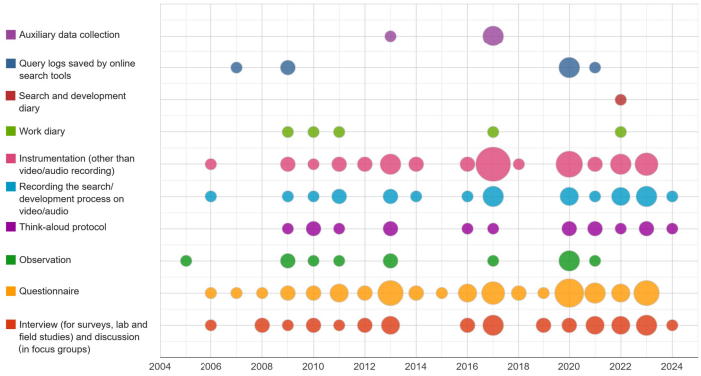

Fig. 7.2. How publications are distributed by type of publication venue and publication year. Bubble size: number of publications.

7.4.2 RQ2: How are the relevant publications distributed over the years by type of investigation?

In creating our chart, we paid attention to taking into account each unique investigation only once. This was possible because we grouped related publications and, in the data extraction of investigations, we referenced the publications that were about the same investigation.

The results are visualized on Fig. 7.3.

If we regard the three types of survey together, the distribution of surveys is rather even over the years.

Only sporadic use of the focus group study and the development diary study can be observed.

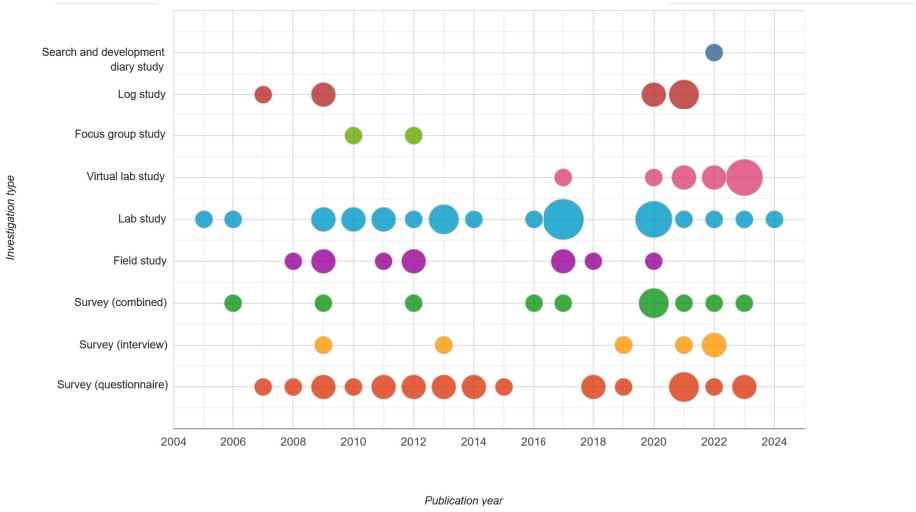

Fig. 7.3. Distribution of investigations by type and publication year.
Bubble size: number of investigations.

Notable is the appearance of an increasing trend in virtual lab studies: Before the pandemic, there was only one virtual lab study, and, since the start of the pandemic, there have been 10. In parallel with this, the number of lab and field studies decreased when the pandemic started.

It is interesting to note that within the lab study category, studies made at exams have emerged [91] [211].

7.4.3 RQ3: What data collection methods have been used and how are these distributed over the years?

Fig. 7.4 presents the yearly distribution of the various data collection methods.

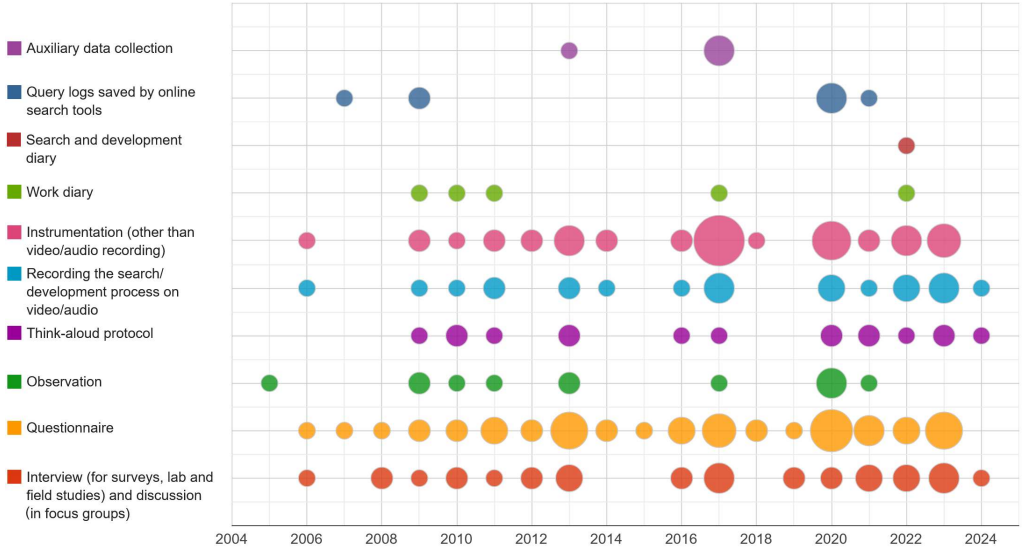

Fig. 7.4. Yearly distribution of the various data collection methods. Bubble size: number of times the method was used.

See Section 7.2.4. for the discussion of data collection methods. The most frequently used methods were: questionnaire, interview, video/audio recording, other types of instrumentation. Somewhat less frequently used: think-aloud, observation. Not too often used: obtaining query log saved by online search tools, auxiliary data collection, asking developers to write a work diary, obtaining data from an existing search and development diary.

The use of questionnaires and of instrumentation has visibly increased in the recent 8 years. The reason may be that the use of both methods has become more easy to apply than before, with the wide availability of online questionnaire tools such as Google forms, and of logging tools. Dominant

among the latter type of tools is the browser plugin (also called add-in, addon, extension) that has been created by the researchers.

7.4.4 RQ4: What non-experimental search tools are used for web search for source code, and how?

Statistics for publications in which one (or one type of) search tool is mentioned as mostly or exclusively used in the context of source code search: Google is mentioned in 3 publications as the sole search tool used. A further 14 publications mention it as the mostly used search tool. 18 other publications state that mostly general purpose search engines (Google, Yahoo, Bing) were used. 3 publications point out Stack Overflow's own search engine as the predominantly used search tool.

If we consider any mentions of tools / sites used for source code search, the statistics are in Tables 7.1 and 7.2 (some defunct / obscure / unidentifiable items have been omitted).

The dominance of Google is undoubtable among the search tools used. Few publications explicitly mention the use of search filters, advanced search options, and settings. According to recent evidence, developers sometimes use general purpose search engines (especially Google) to find content from a specific website (via the *site* option or simply specifying the name of the site), rather than visiting the site directly, even if the site has its own search function [18, 239, 240].

Searching with Google can double as a search in the official documentation as well because Google indexes many documentation sources. One of the reasons why Google is effective is that it works well even with the use of non-expert terminology [260].

Source code search engines (tools that specialize in source code search) are not used too often. Some developers are not even aware of their existence. As to why developers neglect these code search engines, the reasons can be found in [106, 178, 190].

Table 7.1. Mentions of actual search tools used for online source code search

Search tool	Mentions
Google	34
Bing (earlier called MSN search, Windows Live Search, Live Search)	6
DuckDuckGo	2
Yahoo	3
Baidu	1
Yandex	1
NPM	1
SourceForge.net	4
GitHub	5
GitLab	3
freshmeat.net	2
tigris.org	2
TheServerSide.com	2
apache.org	1
eclipse.org	1

Reddit	2
Quora	2
Stack Overflow	13
Google Image Search	1
Google Groups	2
YouTube	2
Specialized software indexes / catalogs	1
Scientific literature (and its search engines)	2

Table 7.2. Mentions of search site categories used for online source code search

Search tool	**Mentions**
General purpose search engines (without naming any)	9
Code search engines (e.g. Koders, Krugle)	9
Language-specific search engines (e.g., Hoogle)	3
Linux distributions	1
Library documentation	3
Program-specific libraries (e.g. assetstore.unity.com)	1
Application-specific (domain-specific) pages	4
Application-specific (domain-specific) pages	4
Official web pages like MSDN or Sun’s Java API	1
Internal, site-specific search engines	1
Open source repositories such as SourceForge.net	6
Domain-specific repositories	1
Public mailing lists or discussion groups (in general)	3
Forums	2
Code example web pages (often organized as forums)	1
Blogs	1

Tutorial websites (e.g. tutorials.com, cplusplus.com)	2
Specialized software indexes / catalogs	1
Scientific literature [and its search engines]	2

7.4.5 RQ5: What are the types and domain names of websites that are the most frequently found and used (clicked)?

There is a high diversity among the web pages. However, Stack Overflow is dominant among the named websites (no other site is even nearly this prominent): of the 25 publications, 16 list Stack Overflow among the visited or preferred sites. Among these 16, 6 mention it specifically in the first place. (Mention of Q&A sites among the visited ones, without naming Stack Overflow or any other, occurs 6 times.)

Other frequently mentioned types of websites are forums (in some publications Q&A sites also count as forums, in others these are two separate categories) and blogs. (It is also sometimes hard to distinguish between 'technical articles' and blogs.) Official and unofficial documentation are also important. 11 publications explicitly mention tutorials and 3 more mention videos (which are probably also tutorials). An interesting finding in two studies [194, 289] is that while the use of tutorials is frequent among students, this activity decreases as students reach a more advanced stage, and [194] also notes that, in parallel with the aforementioned decrease in the use of tutorials, the use of forums increases with the advancement of these students. [289] explicitly mentions that video tutorials are more novice-friendly than other resources.

Concerning the proportions of various websites among the total, some publications state that, after a few items there is a sharp drop in percentage, and numerous sites with a very small percentage form the remaining results ([187] and [188], [263], and somewhat [194]).

7.4.6 RQ6: Which search behaviors and characteristics correlate with success?

In this context, we regard the choice of search engines and formulation of queries as search behavior.

One thing that emerges from the heterogeneous results is that having conceptual knowledge may positively affect search success. Having pre-knowledge is an advantage, but looking for conceptual knowledge during the search process could very well be worth it, too.

There are two, seemingly contradictory findings: “Users seemed to be more successful in getting relevant results with code queries (that do not contain any natural language term but only names and symbols used in code).” [24] vs.

“Successful searches used natural language phrases, such as ‘if statement’, instead of coding shortcuts, such as ‘if’ [23]. The search engine used was Google, which is optimized for natural language queries. So, if we use a general purpose search engine, we may be better off with natural language queries, while code queries probably fit code search engines more. Another

related finding is that, in the context of working with an unfamiliar programming language, the use of borrowed terms from familiar programming languages was more successful on average than typical query [23].

These kinds of results make it clear that the context is relevant. One result was obtained from a code search engine usage log study, the other from a lab study in which a general purpose search engine was used. The difference is probably due to the two search engines serving different purposes and presumably working very differently, the general purpose engine operating better on natural language queries. (Another finding rhyming with this: In software exception-related searches, name-based queries are more successful than ID-based ones [116]. The finding has been drawn from a general purpose search engine usage log study.) Another interesting, related finding worth noting is that one publication states, "(...) over 80% of 'How to' queries successfully supplied participants with the needed information." [23].

The optimal choice of the type of search engine may depend on the code type we want to find: subsystem (library, package) or snippet (block). [247] and [245] have found that code-specific search engines worked better in searches for subsystems, but Google worked better in searches for snippets. However, in some cases general purpose search engines are not always optimal for searching for code snippets: [207] has found that, according to software practitioners, general purpose search engines, e.g., Google, when used for searching for code snippets with generic terms, sometimes do not produce the best results, the code snippets that emerge at the top of the search results list may not be the most desirable solutions. We guess that in these cases the queries were not specific enough, they did not describe the context sufficiently.

Although search success is fundamental to the search process, we found success-related findings in only 17 of the 105 relevant publications. Further research in this area could be helpful, to the benefit of teaching code search methods.

7.4.7 RQ7: Is there evidence on differences in the searches of various developer groups?

The answer to this research question is yes, some evidence has emerged on differences in the searches of developers who can be grouped along with certain characteristics, but only 12 studies have been found relevant in this respect, and the results are somewhat diverse.

Among the findings related to this research question, there is some overlap with the query success findings.

One rather trivial conclusion is that expertise and experience matter a lot, and that also shows in the searches.

There is some difference in the usage of website types between experts (or, in case of students, more advanced students) and novices. According to the findings, experts select tutorial sites somewhat less, and forum sites more than novices [194]. High-scoring students and professionals use Stack Overflow more than low-scoring students and undergraduates do [18, 211]. In addition, professionals use GitHub more than undergraduate students [18]. Novices are more likely than experts to make a search query, look at the first result, and then refine the search [169]. Novices click on the first link displayed in the search

result list, while experts are more likely to click on a link other than the first result [169]. Experts tend to click more search results than novices [169]. However, there is a divergence as to whether experts or novices issue longer queries [23, 169, 183] and who spends more time with search results [23, 169].

There are three studies [72, 239, 240] investigating gender differences, and they have really found differences. These studies were carried out at the same university, the authors overlap, and the number of participants was rather small, so this field could merit further investigation.

7.4.8 RQ8: What are the ranks of inspected / chosen search results in the result list?

12 pertinent publications have been collected. The main conclusion is that most developers do not look beyond the first (or first few) result(s) or at least the first page of results. A substantial proportion, especially novices, do not even look beyond the first result. Thus the ranking of results by the search engine is crucial. However, there is some evidence that experts are more inclined to inspect further results.

7.4.9 RQ9: How do developers keep track of / save their findings during and after the code search process?

Ten pertinent studies have been found. The findings are quite consistent. The two most frequently occurring patterns are:

• *Using the web as external memory: Instead of memorizing often-used findings such as snippets and syntax, developers keep in mind how to find the info on websites and look for it again when necessary [16, 18, 54, 95]. Thus, they memorize the way, the know-how to find the pertinent website, and use this know-how in subsequent searches. This somewhat implies a trust in the stability of the web.*

• *Keeping potentially significant search result pages open in multiple browser tabs [54, 97, 160, 260, 263, 285] or sometimes in separate browser windows [260]. One study [97] also says that developers are reluctant to close such tabs, for fear of having to look for these sites again. There is some occurrence of rearranging the tabs [97] or transferring them to bookmarks [260]. The latter behavior preserves them even after the browser is closed.*

Less frequently occurring strategies: For long search processes, saving the URLs in a diary [263]; making libraries from the retrieved code for later use [197].

7.4.10 RQ10: What are the factors (criteria) used in selecting from the found source codes / programming solutions, what cues are used in this process?

The extracted data are presented below, with the mention of the pertinent publications. The list is arranged so that it goes from the abstract to the concrete, from factors / criteria (which are more associated with packages / libraries / subsystems / components) to cues (which are more associated with snippets).

Items that were considered very important in at least one publication are marked with an asterisk (*). The + sign means that the listed publications are closely related and have been evaluated together.

Functionality (features, fitness to task): [54], [93] + [219], [118, 160, 187, 200, 208, 273]
Widely adopted, recommended, reviewed, starred: [93, 95, 118, 200, 219, 240, 291]
The most relevant looking results [18]
Ranking of search results by the search engine [160, 240]
Search engine preview [249]
Images and snapshots associated with variants [160, 240]
Licensing terms / legal considerations [62, 118, 200, 213, 240, 273, 277, 291]
Readme file [240]
Availability of (appropriate) documentation [95] + [219], [118, 198]
External documentation (such as a tutorial or a page on a Q&A site) [52, 160, 239, 240]
Quality and availability of defect and feature trackers [118]
Amount of user support available [118, 273]
Availability of help] [273]
Response time on requests in forums, mailing lists [118]
* Well maintained and tested library [291]
Quality and availability of roadmaps, future plans [118]
* Title / name of the code (app / library etc.) given by its developer(s) [160, 240]
Tags given by the author(s) [240]
Unnecessary, superfluous functions (negative factor) [160]
Activity (contributors, users, updates... [95, 118, 219, 273, 291]
Number of downloads [118, 198]

A commercial actor guarantees quality of the OSS component [118]

Rating (stars, social media likes... [72, 160, 198, 200, 224, 239, 277, 289]

Maintainability [208]

Usability [208]

Portability [208]

Matches project-specific constraints [120]

* Description of the code [160, 240, 289]

* Code (overall) / Starting point for coding [72, 160, 187, 239, 240]

* Code quality (overall) [118]

* Comments in code [160]

Source code rather than pseudo code [93]

* Code status (such as obsolescence, drawn from the comments) [239]

* Modularity and complexity of the code [53, 118, 291]

Readability, clarity of code [197, 208, 240, 277, 291]

Understandability (of code) [277, 289]

Code example and its relevance [78, 187, 259, 289]

Idiomaticness / patterns [277]

* Programming language / environment [118, 289]

Quality of demo (final output of project) [240]

* Efficiency, performance of the running code [118, 200, 208, 277]

* Security, vulnerability [118, 200, 208, 277]

Reliability [208]

Reusability [208]

* Compatibility [118, 208, 277]

Completeness [208]

Few dependencies to platforms, other components, standards [118]

Architecture [118]

Scalability [121, 197]

* Stability [118, 277, 291]
Ease of implementation / integration [197, 291]
* Source of info / website authority (reputation) / reputation of the provider / community [93, 95, 118, 219, 277]
Quality of provider's website [118]
Author reputation [72, 160, 224, 239, 291]
Comments (on sites) about the source code [93, 200, 239, 289]
Recency (date of creation / posting or update) [72, 78, 160, 240]
* Textual cues (overall) [72, 160]
Text in code looks like relevant method / class / function name [164, 259]
* Title of post / question (in Stack Overflow etc.), question relevance [72, 239, 289]
* Description of question (in Stack Overflow etc.) [160, 239, 289]
Appearance of the Web page e.g. that it does not have many pop-up windows and advertisements or poor graphic design; user interface / navigation [53, 93], [95] + [219], [289]
Friendliness of forums [95]
Familiarity, e.g. variant also appears in other results / result contains elements with which the developer is otherwise familiar [54, 72, 118, 160, 239, 289]

7.4.11 RQ11: Among the search tools used for the mapping study, which found the highest number of relevant results, and how do the tools compare to each other?

In the context of literature reviews, recall means how much of the relevant literature was found. Precision is how much of the retrieved literature is relevant. In calculating recall, we considered our combined results to be the totality of the relevant literature, while, in reality, this is just an approximation.

In this mapping study, our strategy was to maximize recall (within reasonable limits) at the price of attaining only low precision. We applied two different query strategies: one was a set of a high number of queries, the other the Basic

Query. In no case could both strategies be applied to the same tool, due to the various limitations of the tools. Naturally, the number of relevant results retrieved depends not only on how good the search engines are, but also on how good the queries are.

The *via Google* notation means that the queries were executed as Google queries with the *site* specifier.

Recall is relative to all relevant items. By precision (of a particular search engine) we mean the number of relevant items by this search engine divided by the number of total hits by this search engine. In parentheses: in case of Google site queries, number of relevant items not overlapping with non-site Google search hits.

The precision for the search tools queried by the single Basic Query, and the precision for the search tools queried by a large set of queries are not really comparable, as the high number of queries (to Google and Google Scholar) yielded a high number of ‘junk results’ and duplicates.

The number of total hits without Google Books results was 30109 (with Google Books it was 30826).

The total number of hits without Google Books and after running the duplicate eliminator was 16950, these were all manually evaluated. 1166 additional

duplicates were found in the manual evaluation, subtracting these yields 15784 hits.

The number of relevant publications, without the results of the snowballing, was 84. An additional 21 articles were found by snowballing. No Google Books hits produced additional relevant articles (book chapters), so the total number of all relevant articles is 105.

5 relevant publications have been found with Google *site* queries but not found with Google without *site* queries: [16, 200, 207, 248, 276]. However, the publication date of 3 of these publications is 2023, so it is possible that these were found only by the site queries because these were executed later than the original Google queries. (Another possible explanation is that these publications were not included in the results of the original Google queries because their ípages, being recent, had no clicks in people's searches, so they were not deemed popular by the search engine and were excluded from the result list. Without knowing the internal workings of Google this is impossible to determine.) Only two of these publications [200, 207] are dated before 2023. Thus, it was not in vain to use the *site* queries, but it is not known how much of the additional gain is due to their application.

Table 3. Comparison of search tools by number of relevant results found, recall and precision

Tool	# results	# relevant	Recall	Precision
ACM via Google	433	7(1)	7%	1.61%
ArXiv via Google	1277	7(1)	7%	0.55%
CiteSeerX via Google	386	2(0)	2%	0.52%

Frontiers via Google	18	1(0)	1%	5.56%
IEEE Explore via Google	380	2(0)	2%	0.24%
MDPI via Google	159	0(0)	0%	0%
ScienceDirect via Google	412	1(1)	1%	0.24%
Semantic Scholar via Google	813	21(2)	20%	2.58%
Google	6820	59	0.86%	0.86%
Google, incl. *site* queries	10698	64	61%	0.60%
Google Scholar	9059	48	46%	0.53%
ACM, Basic Query (BQ)	852	22	21%	2.58%
CORE, BQ	1221	5	5%	0.41%
Dimensions, BQ	6006	48	46%	0.80%
IEEE Explore, BQ	21	2	2%	10.32%
Lens, BQ	237	10	10%	4.22%
ProQuest, BQ	488 (non-books)	3	3%	0.61%
Scopus, BQ	50	11	10%	22.0%
Springer, BQ	1039	8	8%	0.77%
Web of Science, BQ	41	9	9%	21.95%
Wiley, BQ	110	2	2%	1.81%
WorldCat, BQ	287	19	18%	6.69%

Google has won the contest of finding the highest number of relevant results (with *site* queries, 64 out of 105), with Google Scholar and Dimensions sharing the 2nd and 3rd places, with 48 results each. Thus, recall is 61% for Google (including *site* queries), and 46% for Dimensions and Google Scholar. (Google and Google Scholar were queried with multiple queries – the same queries for both engines –, while Dimensions were queried with the Basic Query.) The

surprises are that Google has outperformed Google Scholar, as far as number of relevant results and recall are concerned, and Dimensions queried with one complex query produced the same number of relevant results as Google Scholar driven by more than 100 queries.

The combined use of the three above-mentioned search tools found all items in the corpus. However, in the case of all three search engines, a very high number of results had to be evaluated to find the relevant ones, so their precision was low. The search tools with a precision greater than 10% were: Scopus, Web of Science, and IEEE Explore. However, they all found relatively few relevant results. All of them were queried with the Basic Query.

With the exception of three publications [3] [166] and [122], all the relevant publications are represented in Google Scholar (this can be verified by searching for these titles in Google Scholar). The three above-mentioned publications have been found by Google only, and they belong to the gray literature. Two of these are actually survey results published only on the web, the third one looks like a scientifically sound research report which was apparently never published elsewhere than on the web. However, we found only 48 of the 105 relevant publications with our Google Scholar queries (even though 102 relevant publications are represented in Google Scholar). The rest was found by Google and / or Dimensions, plus snowballing. Out of the 57 not found by Google Scholar queries, 31 were found by Google queries, and 5 additional ones by the query to Dimensions, the remaining 21 by snowballing. No other search engine used in our searches found any result that was not found by at least one of the search engines Google Scholar, Google, and Dimensions.

12 publications were found only by Google: [3, 51, 52, 97, 103, 122, 149, 166, 200, 260, 268, 276].

7 publications were found solely by Google Scholar: [30, 85, 86, 109, 112, 213, 292].

One publication, [125], was found only by Dimensions.

7.5 Conclusions

We have performed a systematic mapping study on how developers search for source code. To our knowledge, no systematic review or systematic mapping study with the user perspective has been published about this research area yet. We hope that our systematic mapping study gives a useful overview of what research has been achieved in the field so far and could guide research and practice.

Based on the most important findings (presented above) of our systematic review, we list practical guidance for users and creators of code search tools below. (Further advice for users can be found in our previous publication [263].)

Guidance for users of code search tools:

1. Having preliminary conceptual knowledge can positively affect search success, and looking for conceptual knowledge during the search process could also be fruitful.
2. If we use a general purpose search engine, we may be better off with natural language queries, while code queries seem to fit code search engines more.

3. In the context of searching for code while working with an unfamiliar programming language, using terms borrowed from familiar programming languages increases the chance of success.

4. The optimal choice of the type of search engine may depend on the code type we want to find (subsystem i.e. library / package; snippet / block). Two studies have found that code-specific search engines worked better in searches for subsystems, but Google worked better in searches for snippets. However, according to one additional study, general purpose search engines, e.g. Google, when used for searching for code snippets with too generic terms, sometimes do not produce the best results.

Guidance for creators of code search tools:

5. The dominance of Google is undoubtable among the tools used for source code search, any new tool would have to compete with Google. Source code search engines (tools that specialize in source code search) are not used too often. Some developers are not even aware of their existence.

6. Most developers do not look beyond the first (or first few) result(s) or at least the first page of results. A substantial proportion, especially novices, do not even look beyond the first result. Experts are more likely to click on a link other than the first result, and they click more search results than novices. Thus, the ranking of results by the code search engine is crucial, and profiling users (expert / novice) could be beneficial.

Finally, a methodological remark for those who wish to conduct similar systematic reviews: The optimal choice of search tools for the review depends on how much you want to maximize the number of relevant search results at the price of finding a lot of irrelevant results. The present study has found that three engines, Google, Google Scholar, and Dimensions (combined), have found all the relevant results retrieved by this study (no other engine has produced additional relevant results), but these tools also found a lot of irrelevant results. In terms of precision, the search tools with a precision greater than 10% in the present study were: Scopus, Web of Science, and IEEE Explore. However, they all found relatively few relevant results, with Scopus finding the most, 11 out of the 84 found by the searches (not counting snowballing).

Appendix A: Phases of the systematic mapping study

Phase 1: Assembling an initial set of relevant papers. We performed various pre-searches (including snowballing) and found 33 relevant papers. All of these 33 papers were subsequently found by the present systematic mapping study. (Later, a few of these papers, though found relevant, were dismissed as redundant or predecessor articles, i.e. the same data were found in other papers as well.) We call this collection the initial set.

Phase 2: Choosing search tools and formulating the queries. The search tools used in the mapping study have been selected on the basis of whether they yielded significant results to preliminary queries. Several major scholarly search sites and sites of publishers have been tested. listed in Appendix F. Google Scholar has been selected as the main search tool, as it is the most comprehensive academic search engine (according to the literature, e.g. [111, 189], but also to our own pre-tests). Therefore, we first formulated Google Scholar queries (described in Appendix G.2).

We avoided reaching the search engines' limit (typically 1000) on displaying search hits, thus preventing the loss of potentially relevant publications this way. The rationale behind this is that we have no control over the search engines' ranking scheme, which is not necessarily optimal for our purposes[21]5. Because of this and other limits of the search engines, especially query length limits, multiple queries were worked out for Google Scholar, and exactly the same queries were executed in Google as well. These queries proved to be too

[21] Our view is supported not only by our own experiences but also by [10].

complex for the other search tools, so we formulated for them the Basic Query described below.

Google Scholar queries were formed in such a way that, if initially a result list to a query exceeded the 1000-result limit, the query had to be split into several queries, and/or filtering by year had to be applied. Our queries contain all sorts of synonyms and word variations (which are necessary because Google Scholar does not perform stemming on members of OR-ed terms), and also additional terms, and, in some cases, exclusions, to make the queries more specific. The resulting query set consists of more than 100 queries.

For the other tools that do not possess all the sophisticated search features of Google / Google Scholar (such as handling wildcards and OR-ed terms within phrases), we have developed a simpler query which we call ’Basic Query’ (see Appendix G.1).

For search tools where no phrase search (e.g., "source code”) was possible at all, or even the Basic Query was too complex for them (concerning maximal number of terms or other limitations), but information on the articles is publicly accessible on their site, the search was performed via Google, with the *site* specifier (e.g. *site:frontiersin.org*), though this method searches only in the publicly available abstracts and the open access texts. (We used these queries in the hope that we would find hits omitted by Google’s ranking algorithm.)

Google searches (except queries with the *site* specifier) were always performed with the option “with the omitted results included”, and all provided results (not just the first few) had to be processed because we did not know how Google ranked the displayed results. Google often initially indicated a high number of

results, but when the result list was actually viewed, the number became much smaller, and the number of displayed results never reached 1000.

Comparing query results with and without the *site* specifier showed that the listed results for queries without the *site* specifier did not contain all relevant results obtained with the *site* specifier, so Google did not necessarily display all possible results. Therefore, the use of the *site* specifier was justified, but it was at the price of having to inspect a lot of additional search results.

Search sites not offering phrase search and not searchable by Google, either, have been dismissed because they produced mostly irrelevant results, often in an unmanageably high number, when searched with the members of a phrase as separate terms.

Two databases (ACM, IEEE Explore) have been searched in both ways (with their own search engine and via Google). The general purpose search engine Bing was dismissed because of its heavy limitation of 10 terms, which made it impossible to use either the Basic Query or the Google Scholar / Google queries.

The search engines were set to display English results only, where such an option was available. No date limit was set.

For the search in ProQuest, we excluded books because preliminary book searches had produced a high number of totally irrelevant results. Otherwise we included books in the queries to the other search engines, and even searched Google Books. (The Google Books searches were performed via Google, as our preliminary searches on https://books.google.com/, trying the same queries as on Google, produced too many irrelevant results. So, standard Google was used

instead, with exactly the same queries used in the normal Google searches and then clicking to *Books*.)

Phase 3: Executing the queries and collecting the results.
This phase was executed between 17 March 2023 and 2 July 2023.
30109 items have been collected, not counting the Google Books results (717), which were processed separately. With the latter, the total number of results was 30826.

Phase 4: Programmatic elimination of duplicates. A self-developed duplicate eliminator program processed the database of articles and eliminated duplicate items on the basis of the titles.
This phase reduced the number of results to 16,950, that is, almost by half.

Phase 5: Manual processing of articles, acceptance / rejection, grouping related articles.
For articles that did not have PDF (or other full text) links but looked promising, the PDF was obtained. There remained a few articles that could not be obtained in any way.
Then the abstract, the preview (if available), and, in case these looked promising, the full text of the articles were inspected more closely. The acceptance or rejection decision was taken according to the predefined inclusion / exclusion criteria (see Appendix B), and in the case of rejection, the reason was documented if the article was not trivially irrelevant. (We remark that the acceptance / rejection decision was taken solely on the basis of the

above-mentioned criteria and not on whether the article answered any research question. Thus, if further research questions arise in the future, the same collection of articles, the relevant and the redundant ones together, can be used to answer the new research questions, without performing new, time-consuming searches on the web for pertinent publications.)

The related articles were grouped together. Articles which were predecessor versions of later articles (examples: ArXiv version, conference paper version of a later journal article) and not containing any additional information relative to the later article were dismissed as predecessor articles. Articles that were redundant were excluded: The rule of thumb was that any publication worthy of keeping had to have at least some unique findings (worthy of data extraction) not present in any other publication. In case of multiple papers presenting the same evidence, the more elaborate or the later publication was kept (in that order). In case of a tie, that publication was kept, which was more likely to have been rigorously reviewed (by order of importance: published in a journal present in Scimago, IEEE or ACM conference paper, other journal, other conference, IEEE or ACM workshop paper, other workshop paper, thesis, research report, other). In some papers on the same research, certain findings were discussed in one publication and other findings in the other publication. In such cases, both publications were kept.
The difference between treatment of predecessor and redundant articles was that the latter type participated in the snowballing (typically, dissertations were found redundant, but they often had longer reference lists than the related articles).

Phase 6: Snowballing.

The first author performed backward and forward snowballing on all the articles which were deemed relevant, even on those ones which were dismissed as redundant.

The forward snowballing was performed with the aid of Google Scholar, as suggested by [287].

After dealing with all relevant articles, the snowballing continued on the newly found articles, and this was repeated iteratively, until no more new articles were found. This required three rounds. The snowballing was performed between 20-Dec-2023 and 12-Feb-2024.

Snowballing has yielded 21 new articles:

[7] [14] [72] [118] [120] [121] [142] [150] [171] [174] [183] [184] [185] [194] [211] [230] [239] [240] [279] [280] [289].

At this point, the collection of the 105 relevant (and non-redundant) articles was complete.

Phase 7: Data extraction. Data were extracted out of each relevant article in the collection.

The extracted data were the following:

- *Title, author(s), year of publication, category by publication venue (see categories in Section 4.1). (RQ1)*
- *Aim of study (only the part relevant for our review), context.*
- *Main findings and conclusions (only the part relevant for our review).*
- *Type of study, i.e. investigation type. (RQ2)*

• Data acquisition methods, collected data. (RQ3)

• Demographics of participants in study. In case multiple investigations were presented in a publication, demographics were listed separately for each investigation. In case the same investigation was presented in multiple publications, demographics were listed for one publication, and a reference was inserted for the related publications.

• Listing of the (non-experimental) search tools used for source code search by the participants, how these tools were used (filters, operators). (RQ4)

• Found / used(clicked) website types in source code search and list of actual sites / domains. (RQ5)

• Ranks of inspected and selected search results. (RQ8)

• How developers kept track of / saved their findings during and after the search process. (RQ9)

• Differences in search style / strategies among participant groups during the source code search process. (RQ7)

• Factors / criteria / cues having influence on selecting code or webpage. (RQ10)

• How various behaviors, participant characteristics correlated with success. (RQ6)

Phase 8: Answering the research questions. Taking the research questions one by one, the pertinent data extracted from the findings were grouped and summarized. From the data, an answer to the question was constructed. These answers are presented in Section 7.4.

Appendix B: Inclusion / exclusion criteria

This section describes how the result lists of the searches were filtered in order to find the relevant results.
In this list, exclusion criteria have precedence over inclusion criteria, i.e. if an article matches an exclusion criterion, it should be excluded, regardless of whether it matches any inclusion criterion.

Inclusion: The article provides data on how developers search for source code. The data have been obtained by scientifically acceptable methods, such as experiments, field or lab studies, surveys, questionnaires, interviews, observation of developers, analysis of logs and diaries. The search may be for source code of any size, including code snippets.

Inclusion: The article describes any of the following activities of source code search, performed by one or more developers, for software development (classification based on [93]): selecting Web resource (after identifying source code need); translating need to Web resource (interacting with the resource, typically querying, browsing); evaluating results.

Inclusion: The searches were aimed at finding source code (not other type of code). API-related web searches, in the absence of information to the contrary, count as searches for source code. (Explanation: Many articles in the literature categorize developers' searches by intent and not by the type of information sought. In such cases, it is often difficult to distinguish between developers' source code search and their non-source-code search. In the cases where the searches were API-related and there is no additional information on whether the search was for source code, we assume that it was, so we treat these searches as searches for source code [in the absence of information to the contrary]. Searches for API can be considered source code searches because the calling interface is source code – and the search is often for examples of calling the API – even if the source code of the API itself is not available.)

Exclusion: The article does not present relevant evidence which is the result of investigations by at least one of the authors of the article. The article only quotes others' evidence, which is typical in reviews, mapping studies, and the literature review sections of articles.

Exclusion: The article mentions evidence, but it is not described how the evidence was obtained.

Exclusion: The purpose (if identifiable) of the source code searches under investigation is not software development (or at least a simulation of it). The construction of queries by authors for testing their own tool does not count as query construction for software development.

Exclusion: The article describes how a particular tool searches / finds code on the web, but does not describe how a developer (a human) who uses this search tool or other search tools seeks source code on the web.

Exclusion: If the developer searching for code uses an experimental code search tool (even if it is publicly available) and not a commonly known code search tool, unless tests are performed with other, commonly known tools as well and the data thus obtained characterize the source code seeking behavior (using the commonly known tools) of human test subjects. We consider the following as commonly known code search tools:
• The code search engines Google Code Search (and the related ParseWeb), Koders, Ohloh/Black Duck Open Hub, Krugle, Merobase (and the related Code Conjurer), Askyourcode, Searchcode, OpenGrok. We recognize that this selection of code search engines is somewhat arbitrary, it is based on their mention in academic papers as code search engines usually used by developers. We note that certain code search engines mentioned above are already defunct or merged with others.
• The search services provided by public repositories (such as GitHub, SourceForge, snipplr). Here we note that general purpose and academic search engines as well as the search functions of social / Q&A sites
(such as Stack Overflow) can be (and often are) also used for code search, even though they are not tools specifically aimed at code search. Their use is within our scope of investigation.

Exclusion: The article investigates only the fifth stage of source code seeking (described in Section 7.2.2), i.e. the use of the results of the search, without dealing with any of the preceding stages (identify source code
need, select web resource, translate need to web resource, evaluate results).

Exclusion: The paper deals with how developers rate search results but not with how developers obtain those results, or the results were not the outcome of searches where the queries were formulated and executed by the developers themselves.

Exclusion: The paper deals with copy-and-paste behavior but not with how developers find the code to copy.

Exclusion: A description of software download which is not preceded by search, i.e. the developer received a direct ownload link (e.g. in e-mail), and this was not preceded by any search, does not count as source code search. However, searching for articles, web pages in the problem domain and finding a download link to source code in one of the retrieved articles / pages is “in”.

Exclusion: The search for code was not performed online using a publicly accessible search tool or collection / repository. (The code repository of a firm, when it is not accessible to the general public, is not a publicly accessible collection.)

Exclusion: The article does not give evidence on how developers look for code, e.g. it gives just a list of sites where software or search tools can be found.

Exclusion: The article gives advice, guidance, but does not actually show (by proof) how developers really retrieve code using the recommended method(s).
Exclusion: Code search is discussed informally, no scientific evidence can be drawn from it (forum posts, web articles about search engines may fall into this category).

Exclusion: The authors mined software repositories or social sites for traces of source code usage, but did not directly investigate source code search by developers. Note: However, mining search logs (and similar datasets storing user queries) is regarded as direct investigation, as these queries come directly from the developers.

Exclusion: From paper on how developers obtain information, no data specifically on source code search can be
extracted, unless the searches were either API-related searches (in which cases we assume that they were for source

code, see above) or for open source software.

Exclusion: The code search was performed in an integrated development environment (IDE), but the code search function in the IDE did not use the web or a commonly known search tool (see list above) available on the web, or it used an experimental code search tool.

Exclusion: The code search was not for source code, e.g. it was for ready-to-run software (unless interpreted source code) or object code (including object libraries).

Exclusion: The search was for bug localization (within one's own code or in another particular software project).

Exclusion: The article deals with any of the following: feature (concept) location, finding code clones, detecting
malware, code smell detection, code completion, uncovering code plagiarism and it does not deal with code search for
software development.

Exclusion: The article deals with code comprehension, but it does not deal with the search for the code.

Exclusion: The searches for code were not performed by developers (e.g. the queries came from a dataset), or not by the same developers who evaluated / used the results.

Exclusion: The paper carries a plagiarism notice, shows signs of plagiarism, is highly suspect otherwise, or has been withdrawn by the author(s).

Exclusion: The paper is not available online.

Exclusion: The paper is an early variant of a later article which has been identified as relevant.

Exclusion: The item is not written text but video or audio.

Exclusion: The item is a presentation consisting of slides.

Exclusion: The search result is a full book, without naming any relevant chapter in it, the book title does not make it likely that the book is relevant and there is no meaningful preview making it likely that at least one chapter is relevant.

Exclusion: The search result is a full volume of a journal or proceedings, without naming any relevant article in it, and there is no meaningful preview making it likely that at least one article is relevant.

Exclusion: From the article, only very slight information can be obtained about how developers search for source code (example: 'The participants sometimes used Google to solve the programming task' but nothing more is revealed).

Appendix C: The initial set

[23, 24, 26, 27, 30, 32, 33, 34, 52, 54, 94, 95, 124, 127, 129, 142, 169, 176, 187, 188, 219, 226, 245, 247, 257, 258, 259, 263, 268, 273, 290, 291, 292]

Some of these papers, though relevant, were later eliminated because of redundancy.

Appendix D: Primary studies

Studies are generally called primary studies when they are original research of their authors. They are usually published in peer-reviewed journals. Here we list those primary studies which we have found relevant.

[3, 6, 7, 14, 16, 18, 21, 23, 24, 30, 31, 33, 35, 51–54, 62, 72, 75, 78, 85–87, 91, 92, 95–97, 103, 109, 112, 115, 116, 118, 120–122, 124–127, 129, 131, 139, 142, 149, 149, 150, 156, 160, 164, 166, 169–171, 174, 176, 178, 183–185, 187, 188, 191, 192, 194, 197, 198, 200, 201, 203, 207, 208, 211, 213, 219, 224, 226, 227,

229, 230, 239, 240, 242, 245, 247, 249, 253, 256, 258–260, 263, 268, 273, 274, 277, 279, 280, 285, 289–292]

We note that a newer, conference paper version of [16] exists ([17]) but it was not found by the systematic mapping process, perhaps because the publication was too recent.

Appendix E: Relevant but redundant studies

This is the list of publications found relevant but redundant. See the description of Phase 5 of the systematic mapping study process (in Appendix A) for details on the criteria for declaring a publication redundant.

[5, 15, 25, 26. 32, 34, 34, 36, 50, 61, 76, 94, 99, 107, 108, 110, 119, 132, 172, 182, 190, 202, 225, 228, 284, 288]

Appendix F: The used search tools

Table 4 lists the used search tools with their URLs and remarks on the use of these search tools. The prefix *https://* has been omitted from the non-*site* queries. The Basic Query is described in Appendix G.1.

Table 4. The used search tools, with URL or *site* query

Search tool	**URL (or Google *site* query parameter)**	**Remark, settings**
ACM, version 1	dl.acm.org/search/	In: The ACM Guide to

	advanced	Computing Literature. Basic Query
ACM, version 2	site:dl.acm.org/doi/pdf	Queried via Google
ArXiv	site:arxiv.org/	Queried via Google
CiteSeerX	site:citeseerx.ist.psu.edu	Queried via Google
CORE	core.ac.uk/	Basic Query
Dimensions	app.dimensions.ai/ discover/publication	Free version. Search in: Full data. Filtering by year.
Frontiers	site:frontiersin.org	Queried via Google
Google	Firefox default search engine page	Limits: displayed results limited to 1000, query max. 32 words
Google Scholar	scholar.google.hu/	Filtering by year. Limits: query: 256 characters, results: 1000.
IEEE Explore, version 1	ieeexplore.ieee.org/ search/advanced	Basic Query
IEEE Explore, version 2	site:ieee.org	Queried via Google
Lens	www.lens.org/	Scholarly Works tab. Basic Query
MDPI	site:mdpi.com	Queried via Google
ProQuest	www.proquest.com/	Anything but books (books largely irrelevant). Basic Query
ScienceDirect	site:sciencedirect.com	Queried via Google

Scopus	www.scopus.com/ home.uri	Search within: Article title, abstract, keywords. Basic Query
Semantic Scholar	site:semanticscholar.org	Queried via Google
Springer	link.springer.com/ advanced-search	Interface language set to English (because of stemming)
Web of Science	www.webofscience.com/ wos/alldb/advanced-search	Search in: All Databases. Collections: All. Basic Query
Wiley	onlinelibrary.wiley.com	Basic Query
WorldCat	www.worldcat.org/	Basic Query

Appendix G: The search queries

G.1 The Basic Query

The Basic Query, used in multiple search tools:

("code search" OR "search for code" OR "search code" OR "software search" OR "search for software" OR "search software" OR "code retrieval" OR "retrieve code" OR "look for code" OR "look for software") AND ("source code" OR "code snippet" OR "code snippets" OR "example code" OR "sample code" OR "code example" OR "code examples" OR "code sample" OR "code samples") AND (survey OR study OR experiment OR questionnaire OR interview OR log) AND (programmer OR programmers OR developer OR developers OR "software engineer" OR "software engineers" OR student OR students) AND (web OR online OR net OR internet OR google)

This Basic Query has been adapted to the following tools:

ACM:
AllField:("code search" OR "search for code" OR "search code" OR "software search" OR "search for software" OR "search software" OR "code retrieval" OR "retrieve code" OR "look for code" OR "look for software") AND ("source code" OR "code snippet" OR "code snippets" OR "example code" OR "sample code" OR "code example" OR "code examples" OR "code sample" OR "code samples") AND (survey OR study OR experiment OR questionnaire OR interview OR log) AND (programmer OR programmers OR developer OR developers OR "software engineer" OR "software engineers" OR student OR students) AND (web OR online OR net OR internet OR google

Web of Science:
TS=(("code search" OR "search for code" OR "search code" OR "software search" OR "search for software" OR "search software" OR "code retrieval" OR "retrieve code" OR "look for code" OR "look for software") AND ("source code" OR "code snippet" OR "code snippets" OR "example code" OR "sample code" OR "code example" OR "code examples" OR "code sample" OR "code samples") AND (survey OR study OR experiment OR questionnaire OR interview OR log) AND (programmer OR programmers OR developer OR developers OR "software engineer" OR "software engineers" OR student OR students) AND (web OR online OR net OR internet OR google))

G.2 The queries for Google Scholar and Google

The queries were constructed for Google Scholar, but Google was driven by exactly the same queries. It was always checked whether the query remained within the query length limit (256 characters for Google Scholar). In all cases, all results offered by the search engine were obtained.

Google Scholar:

Settings: 20 hits per page, English-language results only, show BibTex info. No restriction on date.

In a few cases, additional search terms and then they as exceptions were added to the query, to split results below the 1000-item limit. Example: a query which we denote with *query*, could be split with the term *input* as
query input
query -input -inputs

In some cases, the hits were also filtered by year to get all results.

Google:
Same queries as on Google Scholar. 100 results per page have been set. The "with the omitted results included" option was always chosen in the hit lists. When, in the results, a missing term was indicated, such as
Missing: experiment|" | Must include: experiment|",
the "must include" option was chosen (if present, otherwise the item was left out). If multiple occurrences of "missing"
were present, we tried to eliminate all of them. In the absence of the possibility of obtaining a result without something "missing", the result was ignored.

The search queries:

"find|finding|finds|found * code snippets|snippet" web|online|internet|net experiment|"lab| laboratory|field study"|survey|observation|interview| questionnaire|log programmer|programmers| developer|developers|"software engineer|engineers"|student|students

"retrieve|retrieving * code snippets|snippet" web|online|internet| net experiment|"lab|laboratory|field study"|survey|observation|interview| questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"retrieves|retrieved * code snippets|snippet" web|online|internet|net experiment|
"lab|laboratory|field study"|survey|observation|interview|questionnaire|log
programmer|programmers|developer|developers|"software engineer|engineers"|
student|students

"find|finding|finds|found code snippets|snippet" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|developers|"software
engineer|engineers"|student|students

"retrieve|retrieves|retrieved code snippets|snippet"
web|online|internet|net experiment|"lab|laboratory|field study"|survey|
observation|interview|questionnaire|log programmer|programmers|developer|
developers|"software engineer|engineers"| student|students

"retrieving code snippets|snippet" web|online|internet|net experiment|"lab|
laboratory|
field study"|survey|observation|interview|questionnaire|log programmer|
programmers|developer|
developers|"software engineer|engineers"|student|students

"hunt|hunting|hunts|hunted ** code snippets|snippet" web|online|internet|
net experiment|"lab|laboratory|field study"|survey|observation|
interview|questionnaire|log programmer|programmers|developer|
developers|"software engineer|engineers"|student|students

"seek|seeking|seeks|sought * code snippets|snippet" web|online|internet|
net experiment|"lab|laboratory|field study"|survey|
observation|interview|questionnaire|log programmer|programmers|
developer|developers|"software engineer|engineers"| student|students

"hunt|hunting|hunts|hunted * code snippets|snippet" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log
programmer|programmers|
developer|developers|"software engineer|engineers"|student|students

"code snippet search"|"code snippet hunt" web|online|internet|net experiment|
"lab|laboratory| field study"|survey|observation|interview|questionnaire|log
programmer|programmers|developer|developers|"software engineer|engineers"|
student|students

"search|searching|searches|searched for code snippets" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer| programmers|developer|developers|"software
engineer|engineers"| student|students

"look|looking|looks|looked for code snippets" web|online|internet|net
experiment| "lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|developers|"software
engineer|engineers"| student|students

"searching|searches|searched for * code snippets" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|developers|"software
engineer|engineers"|student|students

"search|look|looking|looks|looked for * code snippets" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|developers| "software
engineer|engineers"|student|students

"search|searching for ** code snippets" web|online|internet|net experiment|"lab|
laboratory|field study"|survey|observation|interview|questionnaire|log
programmer|programmers|developer|developers|"software engineer|engineers"|
student|students

"searches|searched for ** code snippets" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|
developer|developers|"software engineer|engineers"|student|students

"look|looking|looks|looked for ** code snippets" web|online|internet|net

experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|developers|"software
engineer|engineers"| student|students

"google|googling|googles|googled for * code snippets" experiment|"lab|
laboratory| field study"|survey|observation|interview|questionnaire|log
programmer|programmers|developer|developers|"software engineer|engineers"|
student|students

"google|googling for code snippets" experiment|"lab|laboratory|field study"|
survey|observation|interview|questionnaire|log programmer|programmers|
developer|developers|"software engineer|engineers"|student|students

"googles|googled for code snippets" experiment|"lab|laboratory|field study"
|survey|observation|interview|questionnaire|log programmer|programmers|
developer|developers|"software engineer|engineers"|student|students

"google|googling for ** code snippets" experiment|"lab|laboratory|field study"|
survey|observation|interview|questionnaire|log programmer|programmers|
developer|developers|"software engineer|engineers"|student|students

"googles|googled for ** code snippets" experiment|"lab|laboratory|field study"|
survey|observation|interview|questionnaire|log programmer|programmers|
developer|developers|"software engineer|engineers"|student|students

"source code search" web|online|internet|net experiment|"lab|laboratory|
field study"|survey|observation|interview|questionnaire|analysis|log
programmer|programmers| developer|developers|"software engineer|engineers"|
student|students

"example code search"|"example code hunt" web|online|internet|net experiment|
"lab|laboratory|field study"|survey|observation|interview|questionnaire|log
programmer|programmers|developer|developers|"software engineer|engineers"|
student|students

"hunt for * source|example|sample code" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|

developer|developers|"software engineer|engineers"|student|students

"hunts|hunted for * source|example|sample code" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire| log programmer|programmers|developer|developers|"software
engineer|engineers"| student|students

"search|searching for * source|example|sample code" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|developers|"software
engineer|engineers"|student|students

"searches|searched for * source|example|sample code" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|developers|"software
engineer|engineers"|student|students

"looking|looks for * source|example|sample code" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|track|
questionnaire|log programmer|programmers|developer|developers|"software
engineer|engineers"|student|students

"look|looked for * source|example|sample code" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|developers|"software
engineer|engineers"|student|students

"hunting for * source|example|sample code" web|online|internet|net
experiment| "lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|developers|"software
engineer|engineers"|student|students

"seek|seeks|sought|seeking for * source|example code" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|developers|"software
engineer|engineers"|student|students

"google|googling|googles|googled for * source|example|sample code"

"google|googling|googles|googled for source|example|sample code"

"google|googling|googles|googled for ** source|example|sample code"

"hunt|hunting for source|example|sample code" web|online|internet|net experiment|"lab|laboratory| field study"|survey|observation|interview| questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"hunts|hunted for source|example|sample code" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview| questionnaire|log programmer|programmers|developer|developers| "software engineer|engineers"|student|students

"search|searching for source|example|sample code" web|online|internet|net experiment| "lab|laboratory| field study"|survey|observation|interview| questionnaire|log programmer|programmers|developer| developers| "software engineer|engineers"|student|students

"searches|searched for source|example|sample code" web|online|internet|net experiment|
"lab|laboratory|field study"|survey|observation|interview|questionnaire| log programmer|programmers|developer|developers|"software engineer|engineers" | student|students

"looking|looks|looked for source|example|sample code" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview| questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"look|seek for source|example|sample code" web|online|internet|net experiment| "lab|laboratory|field study"|survey|observation|interview|questionnaire|log programmer|programmers| developer|developers|"software engineer|engineers"| student|students

"seeking|seeks|sought for source|example|sample code" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer
|programmers|developer|developers|"software engineer|engineers"|student|
students

"hunt|hunting for ** source|example|sample code" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|developers|"software
engineer|engineers"|student|students

"hunts|hunted for ** source|example|sample code" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|developers|"software
engineer|engineers"|student|students

"search|searching for ** source|example|sample code" web|online|internet|net
experiment|
"lab|laboratory| field study"|survey|observation|interview| questionnaire|
log programmer|programmers|developer|developers|"software engineer|
engineers"|
student|students

"searches|searched for ** source|example|sample code" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer| programmers|developer|developers|"software
engineer|engineers"|student|students

"look|looking|looks for ** source|example|sample code" web|online|internet|
net experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|developers|"software
engineer|engineers"|student|students

"seek|seeking|seeks for ** source|example|sample code" web|online|internet|
net experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|developers|"software
engineer|engineers"|student|students

"looked|sought for ** source|example|sample code" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview| questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"find|finding|finds|found example|sample code" web|online|internet| net experiment|"lab|laboratory|field study"|survey|observation|interview questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"retrieve|retrieves|retrieved example|sample code" web|online|internet|net experiment|"lab|laboratory| field study"|survey|observation|interview| questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"retrieving example|sample code" web|online|internet|net experiment|"lab| laboratory|field study"|survey|observation|interview|questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"| student|students

"find|finding|finds|found * example|sample code" web|online|internet|net experiment|"lab|laboratory| field study"|survey|observation|interview| questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"retrieve|retrieves|retrieved * example|sample code" web|online|internet|net experiment|"lab|laboratory| field study"|survey|observation|interview| questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"retrieving * example|sample code" web|online|internet|net experiment|"lab| laboratory|field study"|survey| observation|interview|questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"| student|students

"find|finding|finds|found ** example|sample code" web|online|internet|net experiment| "lab|laboratory| field study"|survey|observation|interview|

questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"retrieve|retrieves|retrieved ** example|sample code" web|online|internet|net experiment|"lab|laboratory| field study"|survey|observation|interview| questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"retrieving ** example|sample code" web|online|internet|net experiment|"lab| laboratory|field study"|survey|observation|interview|questionnaire|log programmer|programmers|developer|developers|"software engineer| engineers"|student|students

"find * source code" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview|questionnaire|log programmerprogrammers| developer|developers|"software engineer|engineers"|student|students

"find|finds|found source code" web|online|internet|net experiment|"lab| laboratory|field study"|survey|observation|interview|track|questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"| student|students

"find ** source code" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview|questionnaire|log programmer| programmers|developer|developers|"software engineer|engineers"|student| students

"finds|found * source code" web|online|internet|net experiment|"lab|laboratory| field study"|survey|observation|interview|questionnaire|log programmer| programmers|developer|developers|"software engineer|engineers"|student| students

"find|finding|finds|found source code" web|online|internet|net experiment|"lab| laboratory|field study"|survey|observation|interview|questionnaire|log programmer|programmers|developer| developers|"software engineer|engineers"|student|students

"finding * source code"|"finding ** source code" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|developers|"software
engineer|engineers"|student|students

"seek|seeking|seeks|sought * source code" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|developers|"software
engineer|engineers"|student|students

"seek|seeking|seeks|sought source code" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|developers|"software
engineer|engineers"|student|students

"seek|seeking|seeks|sought ** source code" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|
developers|"software engineer|engineers"|student|students

"retrieve|retrieves|retrieved * source code" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|
developers|"software engineer|engineers"|student|students

"retrieving * source code" web|online|internet|net experiment| "lab|laboratory|
field study"|survey|observation|interview|questionnaire|log programmer|
programmers|developer|developers|"software engineer|engineers"|student|
students

"retrieve|retrieving|retrieves|retrieved source code" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|developers|"software
engineer|engineers"|student|students

"retrieve|retrieves|retrieved ** source code" web|online|internet|net experiment|

"lab|laboratory|field study"|survey|observation|interview|questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"retrieving ** source code" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview|questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"hunt|hunting|hunts|hunted * example code" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview|questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"finding|finds|found|retrieve * example code" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview|questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"retrieving|retrieves|retrieved * example code" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview|questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"finding|finds|found|retrieve * sample code" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview|questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"retrieving|retrieves|retrieved * sample code" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview|questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"seek|seeking|seeks|sought example code" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview|questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"hunt|hunting|hunts|hunted example code" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|
developers|"software engineer|engineers"|student|students

"seek|seeking|seeks|sought sample code" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|
developers|"software engineer|engineers"|student|students

"hunt|hunting|hunts|hunted sample code" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|
developers|"software engineer|engineers"|student|students

"find|finding|finds|found|retrieve example code" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|developers|"software
engineer|engineers"|student|students

"retrieving|retrieves|retrieved example code" web|online|internet|net experiment|
"lab|laboratory|field study"|survey|observation|interview|questionnaire|log
programmer|programmers|developer|developers|"software engineer|engineers"|
student|students

"find|finding|finds|found|retrieve sample code" web|online|internet|net
experiment| "lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|developers|"software
engineer|engineers"|student|students

"retrieving|retrieves|retrieved sample code" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|
developers|"software engineer|engineers"|student|students

"seek|seeking|seeks|sought ** example|sample code" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|

questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"hunt|hunting|hunts|hunted|find ** example|sample code" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview|questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"find ** example|sample code" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview|questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"finding|finds|found|retrieve ** example|sample code" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview|questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"retrieving|retrieves ** example|sample code" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview|questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"retrieved ** example|sample code" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview|questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"seek|seeking|seeks|sought * code examples|samples" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview|questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"hunt|hunting|hunts|hunted * code examples|samples" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview|questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"find * code examples|samples" web|online|internet|net experiment|"lab|
laboratory|field study"|survey|observation|interview|questionnaire|log
programmer|programmers|developer|developers|
"software engineer|engineers"|student|students

"finding|finds|found|retrieve * code examples|samples" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|developers|"software
engineer|engineers"|student|students

"retrieves|retrieved * code examples|samples" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|developers|"software
engineer|engineers"|student|students

"retrieving * code examples|samples" web|online|internet|net experiment|"lab|
laboratory|field study"|survey|observation|interview|questionnaire|log
programmer|programmers|developer|developers|
"software engineer|engineers"|student|students

"seek|seeking|seeks|sought code examples|samples" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|developer|developers|"software
engineer|engineers"|student|students

"hunt|hunting|hunts|hunted code examples|samples" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|
questionnaire|log programmer|programmers|
developer|developers|"software engineer|engineers"|student|students

"find code examples|samples" web|online|internet|net experiment|"lab|
laboratory|field study"|survey|observation|interview|questionnaire|log
programmer|programmers|developer|developers|
"software engineer|engineers"|student|students

"finding|finds|found|retrieving code examples|samples" web|online|internet|net
experiment|"lab|laboratory|field study"|survey|observation|interview|

questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"retrieve|retrieves|retrieved code examples|samples" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview| questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"find ** code examples|samples" web|online|internet|net experiment|"lab| laboratory|field study"|survey|observation|interview|questionnaire|log programmer|programmers|developer|developers| "software engineer|engineers"|student|students

"seek|seeking|seeks|sought ** code examples|samples" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview| questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"hunt|hunting|hunts|hunted ** code examples|samples" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview| questionnaire|log programmer|programmers| developer|developers|"software engineer|engineers"|student|students

"finding|finds|found ** code examples|samples" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview| questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"retrieve|retrieving ** code examples|samples" web|online|internet|net experiment| "lab|laboratory|field study"|survey|observation|interview|questionnaire|log programmer| programmers|developer|developers|"software engineer|engineers"|student| students

"retrieves|retrieved ** code examples|samples" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview|

questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"source code retrieval" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview|questionnaire|log programmerprogrammers|developer|developers|"software engineer|engineers"|student|students

"example|sample code retrieval" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview|questionnaire|log programmer|programmers|developer|developers|"software engineer|engineers"|student|students

"code example|sample retrieval" web|online|internet|net experiment|"lab|laboratory|field study"|survey|observation|interview|questionnaire|log programmer|programmers|developer|developers|
"software engineer|engineers"|student|students

"code|software search skill|skills" "code|software search" "source code" survey student -plagiarism web|online|net|internet|google "search engine|engines" "code|software search" "source code" survey developer|developers|programmer|programmers|"software engineer|engineers" -plagiarism web|online|net|internet|google "search engine|engines" -student -students

"code reuse"|"reuse code" "source code" search survey developer|developers|programmer|programmers|"software engineer|engineers"|student|students -plagiarism

web|online|net|internet|google "search engine|engines" "code search" "source code"|"example|sample code"|"code snippet|snippets|sample|example" programmer|programmers|developer|developers|"software engineer|engineers"|student|students

web|online|net|internet|google -automatic -automated "search|searching for code" "source code"|"example|sample code"|"code snippet|snippets|sample|example" programmer|programmers|developer|developers|"software engineer|engineers"|student|students

web|online|net|internet|google -automatic -automated "search|searching for * code" "source code"|"example|sample code"|"code snippet|snippets|sample|example" programmer|programmers|developer|developers|"software engineer|engineers"|student|students

web|online|net|internet|google -automatic -automated "search|searching for ** code" "source code"|"example|sample code"|"code snippet|snippets|sample|example" programmer|programmers|developer|developers|"software engineer|engineers"|student|students

web|online|net|internet|google -automatic -automated "search|searching" "relevant code" "source code"|"example|sample code"|"code snippet|snippets|sample|example" programmer|programmers|developer|developers|"software engineer|engineers"|student|students

web|online|net|internet|google "lab|field study" "software search"|"search for software" "source code"|"example|sample code"|"code snippet|snippets|sample|example" programmer|programmers|developer|developers|"software engineer|engineers"|student|students web|online|net|internet|google -automatic -automated

Appendix H: The exception list for the duplicate eliminator

The following exceptions were set:

AN ABSTRACT OF THE DISSERTATION OF
Association for Computing Machinery
CEC 322
Code Search
communications - ACM Digital Library
computer
dissertation
Doctoral Thesis
Example-centric programming
For Research Only
information
Institute of Electrical and Electronics Engineers

Introduction
Merobase
Papers/Articles (General) - ASCL.net
Reverse engineering
Technical reviews
THESIS
THESIS/THÈSE
THÈSE
This thesis has been submitted in fulfilment of the ... - ERA
UC Irvine
UC Irvine - eScholarship
UCLA Electronic Theses and Dissertations
UNCORRECTED PROOFS
Understanding the impact of support for iteration on code search
university of california
university of california - eScholarship
Untitled
Web Information Systems and Applications

REFERENCES

[1] [n. d.]. ISO/IEC/IEEE International Standard - Systems and software engineering–Vocabulary. Technical Report. IEEE. https://doi.org/10.1109/IEEESTD.2017.8016712 ISBN: 9781504441186.

[2] 2010. Immersive virtual reality technology. In Communication in the age of virtual reality (transferred to digital print ed.), Frank Biocca (Ed.). Routledge, New York, NY.

[3] 2014. Code Samples Survey 2014 — Results. Survey results. Mozilla Developer Network. https://wiki.mozilla.org/images/a/ae/Code-samples-survey-results.pdf

[4] Niran B. Abbas. 2004. Narrative as Virtual Reality: Immersion and Interactivity in Literature and Electronic Media by Marie-Laure Ryan (review). The Yearbook of English Studies 34, 1 (2004), 277–278. https://doi.org/10.1353/yes.2004.0066

[5] Shamsa Abid. 2021. Feature-driven API Usage-based Code Example Recommendation for Opportunistic Reuse. Ph. D. thesis. Lahore University of Management Sciences. https://shamsa-abid.github.io/files/ShamsaPhDThesis2021.pdf

[6] Shamsa Abid, Shafay Shamail, Hamid Abdul Basit, and Sarah Nadi. 2021. FACER: An API usage-based code-example recommender for opportunistic reuse. Empirical Software Engineering 26, 6 (Nov. 2021), 110. https://doi.org/10.1007/s10664-021-10000-w

[7] Yasemin Acar, Michael Backes, Sascha Fahl, Doowon Kim, Michelle L. Mazurek, and Christian Stransky. 2016. You Get Where You're Looking for: The Impact of Information Sources on Code Security. In 2016 IEEE Symposium on Security and Privacy (SP). IEEE, San Jose, CA, 289–305. https://doi.org/10.1109/SP.2016.25

[8] Ernest Adams and Andrew Rollings. 2007. Fundamentals of game design. Pearson Prentice Hall, Upper Saddle River, N.J.

[9] S. I. Agafonov and A. I. Bobenko. 2003. Hexagonal circle patterns with constant intersection angles and discrete Painlevé and Riccati equations. J. Math. Phys. 44, 8 (Aug. 2003), 3455–3469. https://doi.org/10.1063/1.1586966

[10] Ritu Agarwal and Elena Karahanna. 2000. Time Flies When You're Having Fun: Cognitive Absorption and Beliefs about Information Technology Usage. MIS Quarterly 24, 4 (Dec. 2000), 665. https://doi.org/10.2307/3250951

[11] Sarvesh Agreval, Adèle Simon, Bech Søren, Klaus B. Bærentsen, and Søren Forchammer. 2019. Defining Immersion: Literature Review and Implications for Research on Immersive Audiovisual Experiences. In 147th Audio Engineering Society Convention. Audio Engineering Society. https://doi.org/10.17743/aesconv.2019.978-1-942220-31-2

[12] Az Aktaş and E Orçun. 2016. A survey of computer game development. The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology 13, 2 (April 2016), 239–251. https://doi.org/10.1177/1548512914554405

[13] Mazin Al-Adawi and Mika Luimula. 2019. Demo Paper: Virtual Reality in Fire Safety – Electric Cabin Fire Simulation. In 2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Naples, Italy, 551–552. https://doi.org/10.1109/CogInfoCom47531.2019.9089938

[14] Mareh Fakhir Al-Sammarraie. 2017. An Empirical Investigation of Collaborative Web Search Tool on Novice's Query Behavior. Master's thesis. University of North Florida. https://digitalcommons.unf.edu/cgi/viewcontent.cgi?article=1810&context=etd

[15] Omar Alghamdi. 2023. An investigation of web use during programming. Ph. D. thesis. The University of Manchester, UK. https://pure.manchester.ac.uk/ws/portalfiles/portal/270979802/FULL_TEXT.PDF

[16] Omar Alghamdi, Sarah Clinch, Mohammad Alhamadi, and Caroline Jay. 2023. Novice programmers strategies for online resource use and their impact on source code. (2023). https://doi.org/10.1109/CHASE58964.2023.00018 Publisher: arXiv Version Number: 1.

[17] Omar Alghamdi, Sarah Clinch, Mohammad Alhamadi, and Caroline Jay. 2023. Novice Programmers Strategies for Online Resource Use and Their Impact on Source Code. In 2023 IEEE/ACM 16th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE). IEEE, Melbourne, Australia, 92–104. https://doi.org/10.1109/CHASE58964.2023.00018

[18] Omar Alghamdi, Sarah Clinch, Rigina Skeva, and Caroline Jay. 2023. How are websites used during development and what are the implications for the coding process? Journal of Systems and Software 205 (Nov. 2023), 111803. https://doi.org/10.1016/j.jss.2023.111803

[19] Rutger J. Allan, Irene J. F. De Jong, and Casper C. De Jonge. 2017. From Enargeia to Immersion: The Ancient Roots of a Modern Concept. Style 51,1 (2017), 34–51. https://doi.org/10.1353/sty.2017.0003

[20] Anna Carolina Muller Queiroz, Alexandre Moreira Nascimento, Thomas Brashear Alejandro, Romero Tori, Vinicius Veloso de Melo, Fernando de Souza Meirelles, and Maria Isabel da Silva Leme. 2018. Virtual Reality in Marketing: Technological and Psychological Immersion. In Proceedings of the 24th Americas Conference on Information Systems. https://www.researchgate.net/publication/326957110_Virtual_Reality_in_Marketing_Technological_and_Psychological_immersion

[21] Claudia Ayala, Xavier Franch, Reidar Conradi, Jingyue Li, and Daniela Cruzes. 2013. Developing Software with Open Source Software Components. In Finding Source Code on the Web for Remix and Reuse, Susan Elliott Sim and Rosalva E. Gallardo-Valencia (Eds.). Springer New York, New York, NY, 167–186. https://doi.org/10.1007/978-1-4614-6596-6_9

[22] Daniel Bachmann, Frank Weichert, and Gerhard Rinkenauer. 2018. Review of Three-Dimensional Human-Computer Interaction with Focus on the Leap Motion Controller. Sensors 18, 7 (July 2018), 2194. https://doi.org/10.3390/s18072194

[23] Gina R. Bai, Joshua Kayani, and Kathryn T. Stolee. 2020. How Graduate Computing Students Search When Using an Unfamiliar Programming Language. In Proceedings of the 28th International Conference on Program Comprehension. ACM, Seoul Republic of Korea, 160–171. https://doi.org/10.1145/3387904.3389274

[24] Sushil Bajracharya and Cristina Lopes. 2009. Mining search topics from a code search engine usage log. In 2009 6th IEEE International Working Conference on Mining Software Repositories. IEEE, Vancouver, BC, Canada, 111–120. https://doi.org/10.1109/MSR.2009.5069489

[25] Sushil Krishna Bajracharya. 2010. Facilitating Internet-Scale Code Retrieval. Ph. D. Dissertation. University of California, Irvine. OCLC: 911604762.

[26] Sushil Krishna Bajracharya and Cristina Videira Lopes. 2012. Analyzing and mining a code search engine usage log. Empirical Software Engineering 17, 4-5 (Aug. 2012), 424–466. https://doi.org/10.1007/s10664-010-9144-6

[27] Chetan Bansal, Thomas Zimmermann, Ahmed Hassan Awadallah, and Nachiappan Nagappan. 2019. The Usage of Web Search for Software Engineering. https://arxiv.org/abs/1912.09519v1

[28] Peter Baranyi and Adam Csapo. 2010. Cognitive infocommunications: CogInfoCom. In 2010 11th International Symposium on Computational Intelligence and Informatics (CINTI). IEEE, Budapest, Hungary, 141–146. https://doi.org/10.1109/CINTI.2010.5672257

[29] Péter Baranyi, Adam Csapo, and Gyula Sallai. 2015. Cognitive Infocommunications (CogInfoCom). Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-19608-4

[30] Ohad Barzilay. 2011. Example embedding: On the diversity of example usage in professional software development. Ph. D. thesis. Tel Aviv University, Tel Aviv, Israel.

[31] Ohad Barzilay, Orit Hazzan, and Amiram Yehudai. 2009. Characterizing Example Embedding as a software activity. In 2009 ICSE Workshop on Search-Driven Development-Users, Infrastructure, Tools and Evaluation. IEEE, Vancouver, BC, Canada, 5–8. https://doi.org/10.1109/SUITE.2009.5070011

[32] Ohad Barzilay and Cathy Urquhart. 2014. Understanding reuse of software examples: A case study of prejudice in a community of practice. Information and Software Technology 56, 12 (Dec. 2014), 1613–1628. https://doi.org/10.1016/j.infsof.2014.02.013

[33] Ohad Barzilay, Amiram Yehudai, and Orit Hazzan. 2010. Developers attentiveness to example usage. In Human Aspects of Software Engineering. ACM, Reno Nevada, 1–8. https://doi.org/10.1145/1938595.1938599

[34] Veronika Bauer, Jonas Eckhardt, Benedikt Hauptmann, and Manuel Klimek. 2014. An exploratory study on reuse at google. In Proceedings of the 1st International Workshop on Software Engineering Research and Industrial Practices. ACM, Hyderabad India, 14–23. https://doi.org/10.1145/2593850.2593854

[35] Veronika Bauer and Antonio Vetro'. 2016. Comparing reuse practices in two large software-producing companies. Journal of Systems and Software 117 (July 2016), 545–582. https://doi.org/10.1016/j.jss.2016.03.067

[36] Veronika Maria Bauer. 2016. Analysing and supporting software reuse in practice. Ph.D. thesis. Technische Universität München. https://mediatum.ub.tum.de/doc/1306629/517630.pdf

[37] Beatrix Katalin Szabó. 2019. Rigged hand model for the Blender Game Engine. Recent Innovations in Mechatronics (RIiM) Vol. 6, No. 1 (2019).https://doi.org/10.17667/riim.2019.1/5.

[38] Fahmi Bellalouna. 2019. Virtual-Reality-based Approach for Cognitive Design-Review and FMEA in the Industrial and Manufacturing Engineering. In 2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Naples, Italy, 41–46. https://doi.org/10.1109/CogInfoCom47531.2019.9090002

[39] Joakim D. Bergroth, Hanna M. K. Koskinen, and Jari O. Laarni. 2018. Use of Immersive 3-D Virtual Reality Environments in Control Room Validations. Nuclear Technology 202, 2-3 (June 2018), 278–289. https://doi.org/10.1080/00295450.2017.1420335

[40] Borbála Berki. 2018. 2D Advertising in 3D Virtual Spaces. Acta Polytechnica Hungarica 15, 3 (March 2018). https://doi.org/10.12700/APH.15.3.2018.3.10

[41] Borbála Berki. 2019. Does Effective Use of MaxWhere VR Relate to the Individual Spatial Memory and Mental Rotation Skills? Acta Polytechnica Hungarica 16, 5 (Aug. 2019). https://doi.org/10.12700/APH.16.6.2019.6.4

[42] Borbala Berki. 2019. Sense of Presence in MaxWhere Virtual Reality. In 2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Naples, Italy, 91–94. https://doi.org/10.1109/CogInfoCom47531.2019.9089976

[43] Kornel Bertok and Attila Fazekas. 2014. Recognizing human activities based on head movement trajectories. In 2014 5th IEEE Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Vietri sul Mare, Italy, 273–278. https://doi.org/10.1109/CogInfoCom.2014.7020460

[44] Frank Biocca and Mark R. Levy (Eds.). 1995. Communication in the age of virtual reality. L. Erlbaum Associates, Hillsdale, N.J.

[45] John M. Blain. 2018. The complete guide to Blender graphics: computer modeling & animation (fourth edition ed.). Taylor & Francis, a CRC title, part of the Taylor & Francis imprint, a member of the Taylor & Francis Group, the academic division of T&F Informa, plc, Boca Raton.

[46] Joshua Blake. 2011. Natural user interfaces in .NET: WPF 4, Surface 2, and Kinect. Manning ; Pearson Education [distributor], Greenwich, Conn., London. OCLC: 751755005.

[47] Karen Blakeman. 2013. Finding Research Information on the Web: How to Make the Most of Google and Other Free Search Tools. Science Progress 96, 1 (March 2013), 61–84. https://doi.org/10.3184/003685013X13617253047438

[48] Kata Boros, Gyongyi Bujdoso, Cornelia Mihaela Novac, and Ovidiu Constantin Novac. 2019. E-Health Promotion virtual reality services in MaxWhere VR spaces - design and development. In 2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Naples, Italy, 389–390. https://doi.org/10.1109/CogInfoCom47531.2019.9089975

[49] Wichor M. Bramer, Dean Giustini, and Bianca M. R. Kramer. 2016. Comparing the coverage, recall, and precision of searches for 120 systematic reviews in Embase, MEDLINE, and Google Scholar: a prospective study. Systematic Reviews 5, 1 (Dec. 2016), 39. https://doi.org/10.1186/s13643-016-0215-7

[50] Joel Brandt, Mira Dontcheva, Marcos Weskamp, and Scott R. Klemmer. 2010. Example-centric programming: integrating web search into the development environment. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, Atlanta Georgia USA, 513–522. https://doi.org/10.1145/1753326.1753402

[51] J. Brandt, P.J. Guo, J. Lewenstein, S.R. Klemmer, and M. Dontcheva. 2009. Writing Code to Prototype, Ideate, and Discover. IEEE Software 26, 5 (Sept. 2009), 18–24. https://doi.org/10.1109/MS.2009.147

[52] Joel Brandt, Philip J. Guo, Joel Lewenstein, Mira Dontcheva, and Scott R. Klemmer. 2009. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, Boston MA USA, 1589–1598. https://doi.org/10.1145/1518701.1518944

[53] Joel Brandt, Philip J. Guo, Joel Lewenstein, and Scott R. Klemmer. 2008. Opportunistic programming: how rapid ideation and prototyping occur in practice. In Proceedings of the 4th international workshop on End-user software engineering. ACM, Leipzig Germany, 1–5. https://doi.org/10.1145/1370847.1370848

[54] Joel Richard Brandt. 2010. Example-centric programming: Integrating web search into the development process. Ph. D. thesis. https://purl.stanford.edu/zs049bj7942

[55] Emily Brown and Paul Cairns. 2004. A grounded investigation of game immersion. In CHI '04 Extended Abstracts on Human Factors in Computing Systems. ACM, Vienna Austria, 1297–1300. https://doi.org/10.1145/985921.986048

[56] Tamás Budai and Miklos Kuczmann. 2018. Towards a Modern, Integrated Virtual Laboratory System. Acta Polytechnica Hungarica 15, 3 (March 2018). https://doi.org/10.12700/APH.15.3.2018.3.11

[57] Tamas Budai and Miklos Kuczmann. 2019. A multi-purpose virtual laboratory with interactive knowledge integration. In 2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Naples, Italy, 529–532. https://doi.org/10.1109/CogInfoCom47531.2019.9089896

[58] Gyongyi Bujdoso, Kata Boros, Cornelia Mihaela Novac, and Ovidiu Constantin Novac. 2019. Developing cognitive processes as a major goal in designing e-health information provider VR environment in information science education. In 2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Naples, Italy, 187–192. https://doi.org/10.1109/CogInfoCom47531.2019.9089958

[59] Cecília Sik-Lányi, David J. Brown, Penny Standen, Jacqueline Lewis, Lewis, and Vilma Butkute. 2012. Results of user interface evaluation of serious games for students with intellectual disability. Acta Polytechnica Hungarica 9, 1 (2012), 225–245. https://acta.uni-obuda.hu//Sik-Lanyi_Brown_Standen_Lewis_Butkute_33.pdf

[60] Zenon Chaczko, Ryszard Klempous, Jerzy Rozenblit, Tosiron Adegbija, Christopher Chiu, Konrad Kluwak, and Czeslaw Smutnick. 2020. Biomimetic Middleware Design Principles for IoT Infrastructures. Acta Polytechnica Hungarica 17, 5 (2020), 135–150. https://doi.org/10.12700/APH.17.5.2020.5.7

[61] Weibing Chen, Jingyue Li, Jianqiang Ma, Reidar Conradi, Junzhong Ji, and Chunnian Liu. 2007. A Survey of Software Development with Open Source Components in Chinese Software Industry. In Software Process Dynamics and Agility, Qing Wang, Dietmar Pfahl, and David M. Raffo (Eds.). Vol. 4470. Springer Berlin Heidelberg, Berlin, Heidelberg, 208–220. https://doi.org/10.1007/978-3-540-72426-1_18 Series Title: Lecture Notes in Computer Science.

[62] Weibing Chen, Jingyue Li, Jianqiang Ma, Reidar Conradi, Junzhong Ji, and Chunnian Liu. 2008. An empirical study on software development with open source components in the chinese software industry. Software Process: Improvement and Practice 13, 1 (Jan. 2008), 89–100. https://doi.org/10.1002/spip.361

[63] Christian Stößel and Lucienne Blessing. 2009. Is gesture-based interaction a way to make interfaces more intuitive and accessible?. In HCI2009 Electronic Proceedings: WS4 - Prior Experience. Cambridge. https://www.joernhurtienne.com/iuui/Prior_Experience/Position_Papers_files/StoesselBlessing.pdf

[64] Adam B. Csapo, Hunor Nagy, Arni Kristjansson, and Gyorgy Wersenyi. 2016. Evaluation of human-Myo gesture control capabilities in continuous search and select operations. In 2016 7th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Wroclaw, Poland, 415–420. https://doi.org/10.1109/CogInfoCom.2016.7804585

[65] Fabrizio Cutolo and Vincenzo Ferrari. 2018. The Role of Camera Convergence in Stereoscopic Video See-through Augmented Reality Displays. International Journal of Advanced Computer Science and Applications 9, 8 (2018). https://doi.org/10.14569/IJACSA.2018.090803

[66] David McNeill. 1994. Hand and Mind. What Gestures Reveal About Thought. University of Chicago Press. https://www.researchgate.net/publication/37688404_Hand_and_Mind_What_Gestures_Reveal_About_Thought

[67] Modar Dergham and Attila Gilanyi. 2019. Application of Virtual Reality in Kinematics Education. In 2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Naples, Italy, 107–112. https://doi.org/10.1109/CogInfoCom47531.2019.9089971

[68] Modar Dergham and Attila Gilanyi. 2019. On a System of Virtual Spaces for Teaching Kinematics. In 2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Naples, Italy, 411–414. https://doi.org/10.1109/CogInfoCom47531.2019.9089935

[69] Modar Dergham and Attila Gilanyi. 2019. On a System of Virtual Spaces for Teaching Kinematics. In 2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Naples, Italy, 411–414. https://doi.org/10.1109/CogInfoCom47531.2019.9089935

[70] Luca Di Grazia and Michael Pradel. 2023. Code Search: A Survey of Techniques for Finding Code. Comput. Surveys 55, 11 (Nov. 2023), 1–31. https://doi.org/10.1145/3565971

[71] Julia Diemer, Georg W. Alpers, Henrik M. Peperkorn, Youssef Shiban, and Andreas MÃ¼hlberger. 2015. The impact of perception and presence on emotional reactions: a review of research in virtual reality. Frontiers in Psychology 6 (Jan. 2015). https://doi.org/10.3389/fpsyg.2015.00026

[72] Vaishvi Diwanji, Abim Sedhain, Grey Bodi, and Sandeep Kaur Kuttal. 2022. Developers' Foraging Behavior on Stack Overflow. In 2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). IEEE, Roma, Italy, 1–3. https://doi.org/10.1109/VL/HCC53370.2022.9833123

[73] Dominic Arsenault. 2005. Dark Waters: Spotlight on Immersion. Ghent, Belgium. https://www.researchgate.net/publication/241678065_Dark_waters_Spotlight_on_immersion

[74] Donchyts, Gennadii, Baart, Fedor, van Dam, Arthur, and Jagers, Bert. 2014. Benefits of the use of natural user interfaces in water simulation. In Proceedings of the 7th International Congress on Environmental Modelling and Software (iEMSs). San Diego, California, USA. https://www.researchgate.net/publication/263655945_Benefits_of_the_use_of_natural_user_interfaces_in_water_simulations

[75] Brian Dorn, Adam Stankiewicz, and Chris Roggi. 2013. Lost while searching: Difficulties in information seeking among end-user programmers. Proceedings of the American Society for Information Science and Technology 50, 1 (Jan. 2013), 1–10. https://doi.org/10.1002/meet.14505001059

[76] Brian James Dorn. 2010. A case-based approach for supporting the informal computing education of end-user programmers. Ph. D. Dissertation. Georgia Institute of Technology, Atlanta, GA. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=fe570f2772403a4b5e606aeab0400509ec45f4cc

[77] Rael Dornfest, Paul Bausch, and Tara Calishain. 2006. Google hacks (3rd ed ed.). O'Reilly, Sebastopol, Calif.

[78] Ekwa Duala-Ekoko and Martin P. Robillard. 2010. The information gathering strategies of API learners. Technical report TR-2010.6,. School ofComputer Science, McGill University. https://www.cs.mcgill.ca/~eduala/papers/information-gathering-strategies-ekwa-duala-ekoko.pdf

[79] Eva Dulau, Chrisna R Botha-Ravyse, and Mika Luimula. 2019. Virtual reality for physical rehabilitation: A Pilot study How will virtual reality change physical therapy?. In 2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Naples, Italy, 277–282. https://doi.org/10.1109/CogInfoCom47531.2019.9089980

[80] Eva Dulau, Chrisna R Botha-Ravyse, Mika Luimula, Panagiotis Markopoulos, Evangelos Markopoulos, and Kimmo Tarkkanen. 2019. A virtual reality game for cognitive impairment screening in the elderly: a user

perspective. In 2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Naples, Italy, 403–410. https://doi.org/10.1109/CogInfoCom47531.2019.9089973

[81] Tim Dwyer, Kim Marriott, Tobias Isenberg, Karsten Klein, Nathalie Riche, Falk Schreiber, Wolfgang Stuerzlinger, and Bruce H. Thomas. 2018. Immersive Analytics: An Introduction. In Immersive Analytics, Kim Marriott, Falk Schreiber, Tim Dwyer, Karsten Klein, Nathalie Henry Riche, Takayuki Itoh, Wolfgang Stuerzlinger, and Bruce H. Thomas (Eds.). Vol. 11190. Springer International Publishing, Cham, 1–23. https://doi.org/10.1007/978-3-030-01388-2_1

[82] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. 2008. Selecting Empirical Methods for Software Engineering Research. In Guide to Advanced Empirical Software Engineering, Forrest Shull, Janice Singer, and Dag I. K. Sjøberg (Eds.). Springer London, London, 285–311. https://doi.org/10.1007/978-1-84800-044-5_11

[83] Erik Harpstead, Christopher J. MacLellan, Robert P. Marinier III, and Kenneth R. Koedinger. 2018. Towards natural cognitive system training interactions: a preliminary framework. Technical report. 388–393 pages. https://cdn.aaai.org/Symposia/Spring/2018/SS-2018_Technical_Report_SS- 18.pdf

[84] Laura Ermi and Frans Mäyrä. 2005. Fundamental Components of the Gameplay Experience: Analysing Immersion. In Digital Games Research Conference 2005, Changing Views: Worlds in Play, June 16-20, 2005, Vancouver, British Columbia, Canada. https://dl.digra.org/index.php/dl/article/view/119

[85] Johannes Feiner and Elmar Krajnc. 2009. Copy & Paste Education - Solving Programming Problems with Web Code Snippets. In Proceedings of the Interactive Computer Aided Learning (ICL 2009) conference. Villach, Austria, 81–88. https://www.researchgate.net/profile/Elmar-Krainz/publication/200538971_Copy_Paste_Education_-_Solving_Programming_Problems_with_Web_Code_Snippets/links/00b4953ccba2929a77000000/Copy-Paste-Education-Solving-Programming-Problems-with-Web-Code-Snippets.pdf

[86] Juan M. Fernández-Luna, Juan F. Huete, Ramiro Pérez-Vázquez, and Julio C. Rodríguez-Cano. 2009. Improving search–driven development with collaborative information retrieval techniques. Catholic University of America School of Library and Information Science (SLIS), Washington, D.C.

[87] Felix Fischer, Yannick Stachelscheid, and Jens Grossklags. 2021. The Effect of Google Search on Software Security: Unobtrusive Security Interventions via Content Re-ranking. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. ACM, Virtual Event Republic of Korea, 3070–3084. https://doi.org/10.1145/3460120.3484763

[88] Vilém Flusser. 2014. Gestures. University of Minnesota Press, Minneapolis.

[89] Giuseppe Andrea Fontanelli1, Mario Selvaggio1, Marco Ferro, Fanny Ficuciello, Marilena Vendittelli, and Bruno Siciliano. 2019. Portable dVRK: an augmented V-REP simulator of the da Vinci Research Kit. Acta Polytechnica Hungarica 16, 8 (Sept. 2019). https://doi.org/10.12700/APH.16.8.2019.8.6

[90] Tord Hettervik Froland, Elisabeth Ersvar, Gry Sjeholt, Ilona Heldal, Anne Hjellbrekke Freyen, Shangavi Logeswaran, Attila Kovari, Jozsef Katona, Cristina Costescu, Adrian Rosan, and Andrea Hathazi. 2019. mStikk - A Mobile Application for Learning Phlebotomy. In 2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Naples, Italy, 499–506. https://doi.org/10.1109/CogInfoCom47531.2019.9089979

[91] Markus Fuchs, Markus Heckner, Felix Raab, and Christian Wolff. 2014. Monitoring students' mobile app coding behavior data analysis based on IDE and browser interaction logs. In 2014 IEEE Global Engineering Education Conference (EDUCON). IEEE, Istanbul, 892–899. https://doi.org/10.1109/EDUCON.2014.6826202

[92] Rosalva Eulogia Gallardo Valencia. 2012. How Software Developers Solve Problems by Searching for Source Code on the Web : Studies on Judgments in Evaluation of Results and Information Use. Ph. D dissertation. University of California, Irvine. https://renati.sunedu.gob.pe/bitstream/sunedu/3286980/1/GallardoValenciaRE.pdf

[93] Rosalva E. Gallardo-Valencia. 2012. Source Code Seeking on the Web: A Survey of Empirical Studies and Tools. lulu.com. https://d1wqtxts1xzle7.cloudfront.net/31037275/gallardo-survey.pdf

[94] Rosalva E. Gallardo-Valencia and Susan Elliott Sim. 2009. Internet-Scale Code Search. In 2009 ICSE Workshop on Search-Driven Development-Users, Infrastructure, Tools and Evaluation. IEEE, Vancouver, BC, Canada, 49–52. https://doi.org/10.1109/SUITE.2009.5070022

[95] Rosalva E. Gallardo-Valencia and Susan Elliott Sim. 2011. What kinds of development problems can be solved by searching the web?: a field study. In Proceedings of the 3rd International Workshop on Search-Driven

Development: Users, Infrastructure, Tools, and Evaluation. ACM, Waikiki, Honolulu HI USA, 41–44. https://doi.org/10.1145/1985429.1985440

[96] Rosalva E. Gallardo-Valencia and Susan Elliott Sim. 2013. Software Problems That Motivate Web Searches. In Finding Source Code on the Web for Remix and Reuse, Susan Elliott Sim and Rosalva E. Gallardo-Valencia (Eds.). Springer New York, New York, NY, 253–270. https://doi.org/10.1007/978-1-4614-6596-6_13

[97] Gao Gao, Finn Voichick, Michelle Ichinco, and Caitlin Kelleher. 2020. Exploring Programmers' API Learning Processes: Collecting Web Resources as External Memory. In 2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). IEEE, Dunedin, New Zealand, 1–10. https://doi.org/10.1109/VL/HCC50065.2020.9127274

[98] Vahid Garousi, Michael Felderer, Mika V. Mäntylä, and Austen Rainer. 2020. Benefitting from the Grey Literature in Software Engineering Research. In Contemporary Empirical Methods in Software Engineering, Michael Felderer and Guilherme Horta Travassos (Eds.). Springer International Publishing, Cham, 385–413. https://doi.org/10.1007/978-3-030-32489-6_14

[99] Marina Marinela Gerea. 2007. Selection of Open Source Components - A Qualitative Survey in Norwegian IT Industry. Master's thesis. Norwegian University of Science and Technology. https://ntnuopen.ntnu.no/ntnu-xmlui/bitstream/handle/11250/251181/348488_FULLTEXT01.pdf?sequence=2&isAllowed=y

[100] Alexander Matthias Gerner and Michele Guerra. 2014. On the Cinematic Self. Cinematic Experience as "Out-of-Body" Experience? In Altered self and altered self-experience. BoD - Books on Demand, Norderstedt. https://www.researchgate.net/publication/323582129_Gerner_A_Guerra_M_2014_On_the_Cinematic_Self_Cinematic_experience_as_Out-of-Body_Experience_In_A_Gerner_J_Goncalves_8eds_Altered_Self_and_Altered_Self_Experience_Norderstedt_Bod_p85-106

[101] Dalma Geszten, Anita Komlodi, Karoly Hercegfi, and Wayne G. Lutters. 2018. A Content-Analysis Approach for Exploring Usability Problems in a Collaborative Virtual Environment. Acta Polytechnica Hungarica 15, 5 (Nov. 2018). https://doi.org/10.12700/APH.15.5.2018.5.5

[102] Albert Glinsky and Bob Moog. 2005. Theremin: ether music and espionage. University of Illinois Press, Urbana Chicago Springfield.

[103] Max Goldman and Robert C. Miller. 2009. Codetrail: Connecting source code and web resources. Journal of Visual Languages & Computing 20, 4 (Aug. 2009), 223–235. https://doi.org/10.1016/j.jvlc.2009.04.003

[104] Oliver Grau. 2002. Virtual Art: From Illusion to Immersion. The MIT Press. https://doi.org/10.7551/mitpress/7104.001.0001

[105] Grzegorz Glonek and Maria Pietruszka. 2012. Natural User Interfaces (NUI): Review. Journal of Applied Computer Science 20, 2 (2012), 27–45. https://core.ac.uk/outputs/53096348/?utm_source=pdf&utm_medium=banner&utm_campaign=pdf-decoration-v1

[106] Adam Grzywaczewski. 2013. Consistency of Human Information Behaviour and its Impact on Personalised Contextualised Information Retrieval Systems. Ph. D thesis. Coventry University. https://pureportal.coventry.ac.uk/en/studentTheses/consistency-of-human-information-behaviour-and-its-impact-on-pers

[107] Adam Grzywaczewski and Rahat Iqbal. 2011. Software engineers information behaviour and implicit relevance indicators. International Journal of Knowledge and Web Intelligence 2, 2/3 (2011), 185. https://doi.org/10.1504/IJKWI.2011.044123

[108] Adam Grzywaczewski and Rahat Iqbal. 2012. Task-specific information retrieval systems for software engineers. J. Comput. System Sci. 78, 4 (July 2012), 1204–1218. https://doi.org/10.1016/j.jcss.2011.10.009

[109] Adam Grzywaczewski, Rahat Iqbal, John Halloran, Kashif Iqbal, and Anne James. 2012. Copy and paste as an indicator of rele-vance: Towards the development of intelligent recommender systems for software engineers. In B. Larsen, C. Lioma, and A. P. de Vries, editors: Proceedings of the Task-Based and Aggregated Search (TBAS2012) Workshop, ECIR 2012, Barcelona, Spain, 01-April-2012. Barcelona, Spain, 14–18. https://www.researchgate.net/profile/Diana-Sorensen/publication/256461917_An_exploration_of_retrieval-enhancing_methods_for_integrated_search_in_a_digital_library/links/02e7e522dcf5e06151000000/An-exploration-of-retrieval-enhancing-methods-for-integrated-search-in-a-digital-library.pdf#page=17

[110] Adam Grzywaczewski, Rahat Iqbal, Anne James, and John Halloran. 2011. Software Developers' Information Needs: Towards the Development of Intelligent Recommender Systems. https://doi.org/10.14236/ewic/IUBICOM2011.8

[111] Michael Gusenbauer. 2019. Google Scholar to overshadow them all? Comparing the sizes of 12 academic search engines and bibliographic databases. Scientometrics 118, 1 (Jan. 2019), 177–214. https://doi.org/10.1007/s11192-018-2958-5

[112] Micaela Gustafsson and Rasmus Flomén. 2020. Game developer experience: A cognitive task analysis with different game engines. B. Sc. thesis. Blekinge Institute of Technology. https://www.diva-portal.org/smash/get/diva2:1437636/FULLTEXT01.pdf

[113] Tibor Guzsvinecz, Csaba Kovacs, Dominik Reich, Veronika Szucs, and Cecilia Sik-Lanyi. 2018. Developing a virtual reality application for the improvement of depth perception. In 2018 9th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Budapest, Hungary, 17–22. https://doi.org/10.1109/CogInfoCom.2018.8639935

[114] Gábor Házi and József Páles. 2013. Virtuális vezénylő a paksi teljesléptékű szimulátorhoz (Virtual control room for the full-scope simulator of the Paks nuclear power plant). Nukleon VI (Dec. 2013). http://nuklearis.hu/sites/default/files/nukleon/6_4_146

[115] Lei Han, Tianwa Chen, Gianluca Demartini, Marta Indulska, and Shazia Sadiq. 2020. On Understanding Data Worker Interaction Behaviors. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, Virtual Event China, 269–278. https://doi.org/10.1145/3397271.3401059

[116] Foyzul Hassan, Chetan Bansal, Nachiappan Nagappan, Thomas Zimmermann, and Ahmed Hassan Awadallah. 2020. An Empirical Study of Software Exceptions in the Field using Search Logs. In Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). ACM, Bari Italy, 1–12. https://doi.org/10.1145/3382494.3410692

[117] Wilhelm Hasselbring (Ed.). 2006. Research methods in software engineering. GITO, Berlin.

[118] Oyvind Hauge. 2007. Open Source Software in Software Intensive Industry - A Survey. Master's thesis. Norwegian University of Science and Technology. https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/251241

[119] Oyvind Hauge. 2010. Adoption of Open Source Software in Software-Intensive Industry. Doctoral Thesis. Norwegian University of Science and Technology. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=a5affdab80485941b7a782a2319915b4f0aa65a2

[120] Oyvind Hauge, Thomas Osterlie, Carl-Fredrik Sorensen, and Marinela Gerea. 2009. An empirical study on selection of Open Source Software - Preliminary results. In 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development. IEEE, Vancouver, BC, Canada, 42–47. https://doi.org/10.1109/FLOSS.2009.5071359

[121] Oyvind Hauge and Andreas Røsdal. 2006. A Survey of Industrial Involvement in Open Source. Thesis. Norwegian University of Science and Technology, Trondheim. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=0f2a307de6c1e406aacdd87b2010ea0b4a03fb37

[122] Elise Hein. 2017. Google-Driven Development: A Situated Study of Web Use in Programming. HCI-E MSc Final Project Report. UCL Interaction Centre, University College London. https://elisehe.in/assets/post-assets/googledrivendevelopment.pdf

[123] Roland Hess. 2008. The essential Blender: guide to 3D creation with the open source suite Blender. No Starch Press, San Francisco. ISBN: 1593271662

[124] Raphael Hoffmann, James Fogarty, and Daniel S. Weld. 2007. Assieme: finding and leveraging implicit references in a web search interface for programmers. In Proceedings of the 20th annual ACM symposium on User interface software and technology. ACM, Newport Rhode Island USA,13–22. https://doi.org/10.1145/1294211.1294216

[125] Reid Holmes, Rylan Cottrell, Robert J. Walker, and Jorg Denzinger. 2009. The end-to-end use of source code examples: An exploratory study. In 2009 IEEE International Conference on Software Maintenance. IEEE, Edmonton, AB, Canada, 555–558. https://doi.org/10.1109/ICSM.2009.5306387

[126] Andre Hora. 2021. APISonar: Mining API usage examples. Software: Practice and Experience 51, 2 (Feb. 2021), 319–352. https://doi.org/10.1002/spe.2906

[127] Andre Hora. 2021. Googling for Software Development: What Developers Search For and What They Find. In 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR). IEEE, Madrid, Spain, 317–328. https://doi.org/10.1109/MSR52588.2021.00044

[128] Ildiko Horvath. 2018. Evolution of teaching roles and tasks in VR / AR-based education. In 2018 9th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Budapest, Hungary, 355–360. https://doi.org/10.1109/CogInfoCom.2018.8639907

[129] M. Hucka and M.J. Graham. 2018. Software search is not a science, even among scientists: A survey of how scientists and engineers find software.

Journal of Systems and Software 141 (July 2018), 171–191. https://doi.org/10.1016/j.jss.2018.03.047

[130] Oliver Hummel, Werner Janjic, and Colin Atkinson. 2008. Code Conjurer: Pulling Reusable Software out of Thin Air. IEEE Software 25, 5 (Sept. 2008), 45–52. https://doi.org/10.1109/MS.2008.110

[131] Rahat Iqbal, Adam Grzywaczewski, John Halloran, Faiyaz Doctor, and Kashif Iqbal. 2017. Design implications for task-specific search utilities for retrieval and re-engineering of code. Enterprise Information Systems 11, 5 (May 2017), 738–757. https://doi.org/10.1080/17517575.2015.1086494

[132] Rahat Iqbal, Adam Grzywaczewski, Anne James, Faiyaz Doctor, and John Halloran. 2012. Investigating the value of retention actions as a source of relevance information in the software development environment. In Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD). IEEE, Wuhan, China, 121–127. https://doi.org/10.1109/CSCWD.2012.6221807

[133] Roman Jakobson. 1972. Motor signs for 'Yes' and 'No'. Language in Society 1, 1 (April 1972), 91–96. https://doi.org/10.1017/S0047404500006564

[134] Janos Sebestyen Janosy. 2007. Simulator-Aided Instrumentation and Control System Refurbishment at Paks Nuclear Power Plant. In First Asia International Conference on Modelling & Simulation (AMS'07). IEEE, Phyket, Thailand, 53–58. https://doi.org/10.1109/AMS.2007.94

[135] Janos Sebestyen Janosy, Andras Kereszturi, Gabor Hazi, Jozsef Pales, and Endre Vegh. 2010. Real-Time 3D Simulation of a Pressurized Water Nuclear Reactor. In 2010 12th International Conference on Computer Modelling and Simulation. IEEE, Cambridge, United Kingdom, 414–419. https://doi.org/10.1109/UKSIM.2010.83

[136] Daniela Janssen, Christian Tummel, Anja Richert, and Ingrid Isenhardt. 2016. Virtual Environments in Higher Education – Immersion as a Key Construct for Learning 4.0. International Journal of Advanced Corporate Learning (iJAC) 9, 2 (Aug. 2016), 20. https://doi.org/10.3991/ijac.v9i2.6000

[137] JECRC University, Jaipur, Dr. Manju Kaushik, and Rashmi Jain. 2014. Gesture Based Interaction NUI: An Overview. International Journal of Engineering Trends and Technology 9, 12 (March 2014), 633–636. https://doi.org/10.14445/22315381/IJETT-V9P319

[138] Charlene Jennett, Anna L. Cox, Paul Cairns, Samira Dhoparee, Andrew Epps, Tim Tijs, and Alison Walton. 2008. Measuring and defining the experience of immersion in games. International Journal of Human-Computer Studies 66, 9 (Sept. 2008), 641–661. https://doi.org/10.1016/j.ijhcs.2008.04.004

[139] Xiaoyu Jin. 2017. A Social Information Foraging Approach to Improving End-User Developers ' Productivity. Doctoral dissertation. University of Cincinnati, Cincinnati, OH. http://rave.ohiolink.edu/etdc/view?acc_num=ucin1512039659764376

[140] János Sebestyén Jánosy, Pál Iváncsics, Zoltán Hózer, and Endre Végh. 1993. Upgrading of the Paks Full Scale Simulator. In Proceedings of the 1993 Simulation Multi Conference, Vol. Simulation Series Vol. 25. No. 4. 59–64.

[141] János Sebestyén Jánosy, Tamás Bogdán, Gábor Házi, and Beatrix Katalin Szabó. 1995. Generalising and Upgrading a Set of Simulation Tools to a Full-Graphic Environment. In Proceedings of the 1995 Simulation Multi Conference. SCS, Phoenix, Arizona, 298–303.

[142] George Kakarontzas, Panagiotis Katsaros, and Ioannis Stamelos. 2010. Component Certification as a Prerequisite forWidespread OSS Reuse. Electronic Communications of the EASST (Dec. 2010), Volume 33: Foundations and Techniques for Open Source Software Certification 2010. https://doi.org/10.14279/TUJ.ECEASST.33.449 Publisher: European Association of Software Science and Technology.

[143] Anna Karampela and Carl Vogel. 2019. Nouns and Verbs in Professional Reporting of Extreme Events. In 2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Naples, Italy, 253–258. https://doi.org/10.1109/CogInfoCom47531.2019.9089964

[144] Kevin Brooks. 2003. There is Nothing Virtual About Immersion: Narrative Immersion for VR and Other Interfaces. http://alumni.media.mit.edu/brooks/storybiz/immersiveNotVirtual.pdf

[145] Kisub Kim. 2021. Steps Towards Semantic Code Search. Dissertation. Université du Luxembourg. https://orbilu.uni.lu/bitstream/10993/47704/1/Thesis_Kisub_Kim.pdf

[146] Kisub Kim, Sankalp Ghatpande, Dongsun Kim, Xin Zhou, Kui Liu, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. 2024. Big Code Search: A Bibliography. Comput. Surveys 56, 1 (Jan. 2024), 1–49. https://doi.org/10.1145/3604905

[147] Barbara Kitchenham and Pearl Brereton. 2013. A systematic review of systematic review process research in software engineering. Information and Software Technology 55, 12 (Dec. 2013), 2049–2075. https://doi.org/10.1016/j.infsof.2013.07.010

[148] Barbara Kitchenham and Stuart Charters. 2007. Guidelines for performing Systematic Literature Reviews in Software Engineering, Version 2.3. Technical report EBSE-2007-01. https://www.researchgate.net/profile/Barbara-Kitchenham/publication/302924724_Guidelines_for_performing_Systematic_Literature_Reviews_in_Software_Engineering/links/61712932766c4a211c03a6f7/Guidelines-for-performing-Systematic-Literature-Reviews-in-Software-Engineering.pdf

[149] Amy J. Ko and Yann Riche. 2011. The role of conceptual knowledge in API usability. In 2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). IEEE, Pittsburgh, PA, USA, 173–176. https://doi.org/10.1109/VLHCC.2011.6070395

[150] Andreas P. Koenzen, Neil A. Ernst, and Margaret-Anne D. Storey. 2020. Code Duplication and Reuse in Jupyter Notebooks. In 2020 IEEE SymposiumVisual Languages and Human-Centric Computing (VL/HCC). IEEE, Dunedin, New Zealand, 1–9. https://doi.org/10.1109/VL/HCC50065.2020.9127202

[151] Robert Konrad. 2015. What is the vergence-accommodation conflict and how do we fix it? XRDS: Crossroads, The ACM Magazine for Students 22, 1 (Sept. 2015), 52–55. https://doi.org/10.1145/2810048

[152] Stefan Korecko, Marian Hudak, Branislav Sobota, Martin Marko, Barbora Cimrova, Igor Farkas, and Roman Rosipal. 2018. Assessment and training of visuospatial cognitive functions in virtual reality: proposal and perspective. In 2018 9th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Budapest, Hungary, 39–44. https://doi.org/10.1109/CogInfoCom.2018.8639958

[153] Štefan Korečko, Marián Hudák, and Branislav Sobota. 2019. LIRKIS CAVE: Architecture, Performance and Applications. Acta Polytechnica Hungarica 16, 2 (May 2019). https://doi.org/10.12700/APH.16.2.2019.2.12

[154] Gregory Kramida. 2016. Resolving the Vergence-Accommodation Conflict in Head-Mounted Displays. IEEE Transactions on Visualization and Computer Graphics 22, 7 (July 2016), 1912–1931. https://doi.org/10.1109/TVCG.2015.2473855

[155] Andreas Kratky. 2012. Playing Nature – A Short History of Our Mediated Relationship to Nature. In Human-Computer Interaction, Tourism and Cultural Heritage, David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Francisco Cipolla-Ficarra, Kim Veltman, Huang Chih-Fang, Miguel Cipolla-Ficarra, and Andreas Kratky (Eds.). Vol. 7546. Springer Berlin Heidelberg, Berlin, Heidelberg, 89–98. https://doi.org/10.1007/978-3-642-33944-8_8 Series Title: Lecture Notes in Computer Science.

[156] Apostolos Kritikos, George Kakarontzas, and Ioannis Stamelos. 2010. A SEMI-AUTOMATED PROCESS FOR OPEN SOURCE CODE REUSE:. In Proceedings of the Fifth International Conference on Evaluation of Novel Approaches to Software Engineering. SciTePress - Science and and Technology Publications, University of Piraeus, Greece, 179–185. https://doi.org/10.5220/0002999401790185

[157] M. Kuczmann and P. Baranyi. 2019. State Space Model Based Control in Virtual Laboratory. In 2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Naples, Italy, 507–510. https://doi.org/10.1109/CogInfoCom47531.2019.9090003

[158] Miklós Kuczmann and Tamás Budai. 2019. Linear State Space Modeling and Control Teaching in MaxWhere Virtual Laboratory. Acta Polytechnica Hungarica 16, 5 (Aug. 2019). https://doi.org/10.12700/APH.16.6.2019.6.3

[159] Logan Kugler. 2021. The state of virtual reality hardware. Commun. ACM 64, 2 (Jan. 2021), 15–16. https://doi.org/10.1145/3441290

[160] Sandeep Kaur Kuttal, Se Yeon Kim, Carlos Martos, and Alexandra Bejarano. 2021. How end-user programmers forage in online repositories? An information foraging perspective. Journal of Computer Languages 62 (Feb. 2021), 101010. https://doi.org/10.1016/j.cola.2020.101010

[161] Sandeep Kaur Kuttal, Anita Sarma, and Gregg Rothermel. 2013. Predator behavior in the wild web world of bugs: An information foraging theory perspective. In 2013 IEEE Symposium on Visual Languages and Human Centric Computing. IEEE, San Jose, CA, USA, 59–66. https://doi.org/10.1109/VLHCC.2013.6645244

[162] Jari Laarni, Marja Liinasuo, Satu Pakarinen, Kristian Lukander, and Tomi Passi. 2021. Control Room Operators' Cognitive Strategies in Complex Troubleshooting. In Advances in Neuroergonomics and Cognitive Engineering, Hasan Ayaz, Umer Asgher, and Lucas Paletta (Eds.). Vol. 259. Springer International Publishing, Cham, 238–245. https://doi.org/10.1007/978-3-030-80285-1_29 Series Title: Lecture Notes in Networks and Systems.

[163] Jari Laarni, Marja Liinasuo, Satu Pakarinen, Kristian Lukander, Tomi Passi, Ville Pitkänen, and Leena Salo. 2020. Building Cognitive Readiness and Resilience Skills for Situation Assessment and Diagnostic Reasoning in a VR CR. In HCI International 2020 - Posters, Constantine Stephanidis and Margherita Antona (Eds.). Vol. 1225. Springer International Publishing, Cham, 77–84. https://doi.org/10.1007/978-3-030-50729-9_11 Series Title: Communications in Computer and Information Science.

[164] Sam Lau, Sruti Srinivasa Srinivasa Ragavan, Ken Milne, Titus Barik, and Advait Sarkar. 2021. TweakIt: Supporting End-User Programmers Who Transmogrify Code. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems. ACM, Yokohama Japan, 1–12. https://doi.org/10.1145/3411764.3445265

[165] Brenda Laurel and S. Joy Mountford (Eds.). 1990. The Art of human-computer interface design. Addison-Wesley Pub. Co, Reading, Mass.

[166] JaYoung Lee. 2020. What we learned from the Flutter Q2 2020 survey. Survey report. Flutter Team. https://medium.com/flutter/what-we-learned-from-the-flutter-q2-2020-survey-a4f1fc8faac9

[167] Per Lenberg, Robert Feldt, and Lars Göran Wallgren. 2015. Behavioral software engineering: A definition and systematic literature review. Journal of Systems and Software 107 (Sept. 2015), 15–37. https://doi.org/10.1016/j.jss.2015.04.084

[168] Giovanna Leone, Stefano Migliorisi, and Isora Sessa. 2016. Detecting social signals of honesty and fear of appearing deceitful: A methodological proposal. In 2016 7th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Wroclaw, Poland, 289–294. https://doi.org/10.1109/CogInfoCom.2016.7804563

[169] Annie Li, Madeline Endres, and Westley Weimer. 2022. Debugging with stack overflow: web search behavior in novice and expert programmers. In Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Software Engineering Education and Training. ACM, Pittsburgh Pennsylvania, 69–81. https://doi.org/10.1145/3510456.3514147

[170] Hongwei Li, Zhenchang Xing, Xin Peng, and Wenyun Zhao. 2013. What help do developers seek, when and how?. In 2013 20th Working Conference n Reverse Engineering (WCRE). IEEE, Koblenz, Germany, 142–151. https://doi.org/10.1109/WCRE.2013.6671289

[171] Jing Li, Lingfeng Bao, Zhenchang Xing, Xinyu Wang, and Bo Zhou. 2016. BPMiner: mining developers' behavior patterns from screen-captured task videos. In Proceedings of the 31st Annual ACM Symposium on Applied Computing. ACM, Pisa Italy, 1371–1377. https://doi.org/10.1145/2851613.2851771

[172] Shuangyi Li. 2022. An Investigation into Code Search Engines: The State of the Art Versus Developer Expectations. Diss. Virginia Tech. https://vtechworks.lib.vt.edu/items/2665bfc8-c1b3-4f62-baf5-d177b9718f84

[173] Zheng Li. 2021. Stop Building Castles on a Swamp! The Crisis of Reproducing Automatic Search in Evidence-Based Software Engineering. In 2021 IEEE/ACM 43rd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER). IEEE, Madrid, ES, 16–20. https://doi.org/10.1109/ICSE-NIER52604.2021.00012

[174] Chun Jiann Lim and Moon Ting Su. 2023. RECOMMENDING JAVA API METHODS BASED ON PROGRAMMING TASK DESCRIPTIONS BY NOVICE PROGRAMMERS. Malaysian Journal of Computer Science 36, 2 (April 2023), 148–172. https://doi.org/10.22452/mjcs.vol36no2.3

[175] Chao Liu, Xin Xia, David Lo, Cuiyun Gao, Xiaohu Yang, and John Grundy. 2022. Opportunities and Challenges in Code Search Tools. Comput. Surveys 54, 9 (Dec. 2022), 1–40. https://doi.org/10.1145/3480027

[176] Jiakun Liu, Sebastian Baltes, Christoph Treude, David Lo, Yun Zhang, and Xin Xia. 2021. Characterizing search activities on stack overflow. In Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ACM, Athens Greece, 919–931. https://doi.org/10.1145/3468264.3468582

[177] Michael Xieyang Liu, Shaun Burley, Emily Deng, Angelina Zhou, Aniket Kittur, and Brad A. Myers. 2018. Supporting Knowledge Acceleration for Programming from a Sensemaking Perspective. In Sensemaking Workshop at CHI Conference on Human Factors in Computing Systems, 2018. Montreal, QC, Canada.

[178] Yin Liu, Shuangyi Li, and Eli Tilevich. 2022. Toward a Better Alignment Between the Research and Practice of Code Search Engines. In 2022 29th Asia-Pacific Software Engineering Conference (APSEC). IEEE, Japan, 219–228. https://doi.org/10.1109/APSEC57359.2022.00034

[179] Zhiyao Liu, Qichao Zhao, Liming Zhang, Xuegang Zhang, Jieyun Fan, Qingju Wang, and Ping Wu. 2021. Quantitative Evaluation on the Effect of Experience Under Emergency Situations in NPP Main Control Room Based on Multimodal Data. Nuclear Technology 207, 4 (April 2021), 575–581. https://doi.org/10.1080/00295450.2020.1784683

[180] Matthew Lombard and Theresa Ditton. 2006. At the Heart of It All: The Concept of Presence. Journal of Computer-Mediated Communication 3, 2 (June 2006), 0–0. https://doi.org/10.1111/j.1083-6101.1997.tb00072.x

[181] Jack M. Loomis, James J. Blascovich, and Andrew C. Beall. 1999. Immersive virtual environment technology as a basic research tool in psychology. Behavior Research Methods, Instruments, & Computers 31, 4 (Dec. 1999), 557–564. https://doi.org/10.3758/BF03200735

[182] Yihan Lu and I-Han Hsiao. 2016. Seeking Programming-related Information from Large Scaled Discussion Forums, Help or Harm? International Educational Data Mining Society. https://files.eric.ed.gov/fulltext/ED592724.pdf

[183] Yihan Lu and I-Han Hsiao. 2017. Personalized Information Seeking Assistant (PiSA): from programming information seeking to learning. Information Retrieval Journal 20, 5 (Oct. 2017), 433–455. https://doi.org/10.1007/s10791-017-9305-y

[184] Yihan Lu and I-Han Hsiao. 2017. Toward understanding novices' search process in programming problem solving. In 2017 IEEE Frontiers in Education Conference (FIE). IEEE, Indianapolis, IN, 1–7. https://doi.org/10.1109/FIE.2017.8190706

[185] Yihan Lu, I-Han Hsiao, and Qi Li. 2016. Exploring Online Programming-Related Information Seeking Behaviors via Discussion Forums. In 2016 IEEE 16th International Conference on Advanced Learning Technologies (ICALT). IEEE, Austin, TX, USA, 283–287. https://doi.org/10.1109/ICALT.2016.63

[186] Evangelos Markopoulos, Jenny Lauronen, Mika Luimula, Pihla Lehto, and Sami Laukkanen. 2019. Maritime Safety Education with VR Technology (MarSEVR). In 2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Naples, Italy, 283–288. https://doi.org/10.1109/CogInfoCom47531.2019.9089997

[187] Lee Martie, André Van Der Hoek, and Thomas Kwak. 2017. Understanding the impact of support for iteration on code search. In Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. ACM, Paderborn Germany, 774–785. https://doi.org/10.1145/3106237.3106293

[188] Lee Thomas Martie. 2017. Understanding the Impact of Support for Iteration on Code Search. Ph. D. dissertation. University of California, Irvine. https://escholarship.org/uc/item/7692s82b

[189] Alberto Martín-Martín, Mike Thelwall, Enrique Orduna-Malea, and Emilio Delgado López-Cózar. 2021. Google Scholar, Microsoft Academic, Scopus, Dimensions, Web of Science, and OpenCitations' COCI: a multidisciplinary comparison of coverage via citations. Scientometrics 126, 1 (Jan. 2021), 871–906. https://doi.org/10.1007/s11192-020-03690-4

[190] Collin McMillan. 2012. Searching, Selecting, And Synthesizing Source Code Components. (2012). https://doi.org/10.21220/S2-2VAG-BY15 Publisher: College of William and Mary - Arts and Sciences.

[191] Collin McMillan, Mark Grechanik, Denys Poshyvanyk, Chen Fu, and Qing Xie. 2011. Exemplar: A Source Code Search Engine for Finding Highly Relevant Applications. IEEE Transactions on Software Engineering 38, 5 (2011), 1069–1087. https://doi.org/10.1109/TSE.2011.84

[192] Collin Mcmillan, Denys Poshyvanyk, Mark Grechanik, Qing Xie, and Chen Fu. 2013. Portfolio: Searching for relevant functions and their usages in millions of lines of code. ACM Transactions on Software Engineering and Methodology 22, 4 (Oct. 2013), 1–30. https://doi.org/10.1145/2522920.2522930

[193] Mel Slater. 2003. A Note on the Presence Terminology. http://www0.cs.ucl.ac.uk/research/vr/Projects/Presencia/ConsortiumPublications/ucl_cs_papers/presence-terminology.htm

[194] Ryan Michaels, Naveen Tula, Susie Ramisetty-Mikler, Dmitry Nurmuradov, and Renee Bryce. 2020. An empirical study of how novice programmers search the web for help. The Journal of Computing Sciences in Colleges 36, 2 (2020), 42–52. https://doi.org/10.5555/3447065.3447071

[195] Juraj Mihaľov, Emília Pietriková, Anton Baláž, Branislav Madoš, and Norbert Ádám. 2018. Potential of Low Cost Motion Sensors Compared to Programming Environments. Acta Polytechnica Hungarica 15, 6 (2018), 155–177. https://doi.org/10.12700/APH.15.6.2018.6.9

[196] David Mioduser. 2014. From real virtuality in Lascaux to virtual reality today: Cognitive processes with cognitive technologies. In From orthography to pedagogy: Essays in honor of Richard L. Venezky, (0 ed.), Thomas R. Trabasso, John P. Sabatini, Dominic W. Massaro, and Robert Calfee (Eds.). Psychology Press. https://doi.org/10.4324/9781410613028

[197] Alfaroq O.M. Mohammed, Ziad A. Abdelnabi, and Abdalmunam Abdalla. 2021. The Influence of Code Retrieval from the Web on Programmer's Skills, Meth-odologies, and Coding Behaviors. AL-MUKHTAR JOURNAL OF SCIENCES 36, 2 (June 2021), 160–166. https://doi.org/10.54172/mjsc.v36i2.66

[198] Suhaib Mujahid, Rabe Abdalkareem, and Emad Shihab. 2023. What are the characteristics of highly-selected packages? A case study on the npm ecosystem. Journal of Systems and Software 198 (April 2023), 111588. https://doi.org/10.1016/j.jss.2022.111588

[199] Janet Horowitz Murray. 1997. Hamlet on the holodeck: the future of narrative in cyberspace. Free Press, New York.

[200] Niko Mäkitalo, Antero Taivalsaari, Arto Kiviluoto, Tommi Mikkonen, and Rafael Capilla. 2020. On opportunistic software reuse. Computing 102, 11 (Nov. 2020), 2385–2408. https://doi.org/10.1007/s00607-020-00833-6

[201] Luana Müller, Milene Selbach Silveira, and Clarisse S. De Souza. 2019. Source Code Comprehension and Appropriation by Novice Programmers: Understanding Novice Programmers' Perception about Source Code Reuse. Journal of Interactive Systems 10 (Dec. 2019), 96. https://doi.org/10.5753/jis.2019.556

[202] Keitaro Nakasai, Masateru Tsunoda, and Hideaki Hata. 2016. Web search behaviors for software development. In Proceedings of the 9th International Workshop on Cooperative and Human Aspects of Software Engineering. ACM, Austin Texas, 125–128. https://doi.org/10.1145/2897586.2897614

[203] Keitaro Nakasai, Masateru Tsunoda, and Kenichi Matsumoto. 2022. Analyzing Web Search Strategy of Software Developers to Modify Source Codes. IEICE Transactions on Information and Systems E105.D, 1 (Jan. 2022), 31–36. https://doi.org/10.1587/transinf.2021MPL0004

[204] Costanza Navarretta. 2015. Pauses delimiting semantic boundaries. In 2015 6th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Gyor, Hungary, 533–537. https://doi.org/10.1109/CogInfoCom.2015.7390650

[205] Costanza Navarretta. 2016. Predicting an individual's gestures from the interlocutor's co-occurring gestures and related speech. In 2016 7th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Wroclaw, Poland, 233–238. https://doi.org/10.1109/ CogInfoCom.2016.7804554

[206] Costanza Navarretta. 2018. Automatic Gender and Identity Recognition in Annotated Multimodal Face-to-face Conversations. In 2018 9th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Budapest, Hungary, 87–92. https://doi.org/10.1109/ CogInfoCom.2018.8639905

[207] Ifeanyi G. Ndukwe, Sherlock A. Licorish, and Stephen G. MacDonell. 2022. Perceptions on the Utility of Community Question and Answer Websites Like Stack Overflow to Software Developers. IEEE Transactions on Software Engineering 49, 4 (2022), 2413–2425. https://doi.org/10.1109/TSE.2022.3220236

[208] Ifeanyi G. Ndukwe, Sherlock A. Licorish, Amjed Tahir, and Stephen G. MacDonell. 2022. How have views on Software Quality differed over time? Research and practice viewpoints. Journal of Systems and Software 195 (2022), 111524. https://doi.org/10.1016/j.jss.2022.111524

[209] Duc-Canh Nguyen, Gerard Bailly, and Frederic Elisei. 2016. Conducting neuropsychological tests with a humanoid robot: Design and evaluation. In 2016 7th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Wroclaw, Poland, 337–342. https://doi.org/10.1109/CogInfoCom.2016.7804572

[210] Donald A. Norman. 2010. Natural user interfaces are not natural. Interactions 17, 3 (May 2010), 6–10. https://doi.org/10.1145/1744161.1744163

[211] Henrik Nygren, Juho Leinonen, and Arto Hellas. 2017. Tracking Students' Internet Browsing in a Machine Exam. In Proceedings of the 6th Computer Science Education Research Conference. ACM, Helsinki Finland, 91–95. https://doi.org/10.1145/3162087.3162103

[212] Oliver Hummel. 2008. Semantic Component Retrieval in Software Engineering. PhD thesis. Universität Mannheim, Mannheim. https://madoc.bib.uni-mannheim.de/1883/1/Dissertation_OliverHummel.pdf

[213] Maryam Omer. 2022. Using Stack Overflow for Professional Learning and Practices at Work. Master's thesis. University of Gothenburg. https://gupea.ub.gu.se/bitstream/handle/2077/74189/ITLGU%20Thesis_Maryam%20Omer.pdf?sequence=1

[214] Stanislav Ondas, Jozef Juhar, Matus Pleva, Peter Fercak, and Rastislav Husovsky. 2017. Multimodal dialogue system with NAO and VoiceXML dialogue manager. In 2017 8th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Debrecen, 439–444. https://doi.org/10.1109/CogInfoCom.2017.8268286

[215] Thomas Ousterhout. 2015. Cross-form facilitation effects from simultaneous gesture/word combinations with ERP analysis. In 2015 6th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Gyor, Hungary, 493–497. https://doi.org/10.1109/CogInfoCom.2015. 7390643

[216] Thomas Ousterhout and Costanza Navarretta. 2015. Reaction time for two types of semantically related gesture and sentence pairs. In 2015 6th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Gyor, Hungary, 499–503. https://doi.org/10.1109/CogInfoCom.2015.7390644

[217] Eli Pariser. 2011. The filter bubble: what the Internet is hiding from you (1. publ ed.). Viking, London.

[218] Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. Information and Software Technology 64 (Aug. 2015), 1–18. https://doi.org/10.1016/j.infsof.2015.03.007

[219] Kavita Philip, Medha Umarji, Megha Agarwala, Susan Elliott Sim, Rosalva Gallardo-Valencia, Cristina V. Lopes, and Sukanya Ratanotayanon. 2012. Software reuse through methodical component reuse and amethodical snippet remixing. In Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work. ACM, Seattle Washington USA, 1361–1370. https://doi.org/10.1145/2145204.2145407

[220] Sakari Pieskä, Mika Luimula, and Taisto Suominen. 2019. Fast Experimentations with Virtual Technologies Pave the Way for Experience Economy. Acta Polytechnica Hungarica 16, 5 (Aug. 2019). https://doi.org/10.12700/APH.16.6.2019.6.2

[221] Lori Pollock. 2016. Experiences in scaling field studies of software developer behavior: keynote for the software engineering research & industrial practice workshop. In Proceedings of the 3rd International Workshop on Software Engineering Research and Industrial Practice. ACM, Austin Texas, 1–2. https://doi.org/10.1145/2897022.2897838

[222] Péter Baranyi and Ádám Csapó. 2012. Definition and synergies of cognitive infocommunications. Acta Polytechnica Hungarica 9(1) (2012), 67–83. https://acta.uni-obuda.hu//Baranyi_Csapo_33.pdf

[223] Péter Galambos. 2012. Vibrotactile feedback for haptics and telemanipulation: Survey, concept and experiment. Acta Polytechnica Hungarica 9(1) (2012), 41–65. https://acta.uni-obuda.hu/Galambos_33.pdf

[224] Chaiyong Ragkhitwetsagul, Jens Krinke, and Rocco Oliveto. 2018. Awareness and Experience of Developers to Outdated and License-Violating Code on Stack Overflow: An Online Survey. (2018). https://doi.org/10.48550/ARXIV.1806.08149 Publisher: arXiv Version Number: 1.

[225] Chaiyong Ragkhitwetsagul, Jens Krinke, Matheus Paixao, Giuseppe Bianco, and Rocco Oliveto. 2021. Toxic Code Snippets on Stack Overflow. IEEE Transactions on Software Engineering 47, 3 (March 2021), 560–581. https://doi.org/10.1109/TSE.2019.2900307

[226] Md Masudur Rahman, Jed Barson, Sydney Paul, Joshua Kayani, Federico Andrés Lois, Sebastián Fernandez Quezada, Christopher Parnin, Kathryn T. Stolee, and Baishakhi Ray. 2018. Evaluating how developers use general-purpose web-search for code retrieval. In Proceedings of the 15th International Conference on Mining Software Repositories. ACM, Gothenburg Sweden, 465–475. https://doi.org/10.1145/3196398.3196425

[227] Nikitha Rao, Chetan Bansal, Thomas Zimmermann, Ahmed Hassan Awadallah, and Nachiappan Nagappan. 2019. Analyzing Web Search Behavior for Software Engineering Tasks. (2019). https://doi.org/10.48550/ARXIV.1912.09519 Publisher: [object Object] Version Number: 3.

[228] Brittany Reid. 2023. Improving Developer Efficiency through Code Reuse. Ph. D. thesis. The University of Adelaide. https://hdl.handle.net/2440/139921

[229] Brittany Reid, Marcelo d'Amorim, Markus Wagner, and Christoph Treude. 2023. NCQ: Code Reuse Support for Node.js Developers. IEEE Transactions on Software Engineering 49, 5 (May 2023), 3205–3225. https://doi.org/10.1109/TSE.2023.3248113

[230] V. Smrithi Rekha and S. Venkatapathy. 2015. Understanding the Usage of Online Forums as Learning Platforms. Procedia Computer Science 46 (2015), 499–506. https://doi.org/10.1016/j.procs.2015.02.074

[231] Ton Roosendaal, Roland Hess, and Blender Foundation (Eds.). 2007. The essential Blender: guide to 3D creation with the open source suite Blender. No Starch Press, San Francisco, CA. OCLC: ocn162126897.

[232] Frank Rose. 2012. The art of immersion: how the digital generation is remaking Hollywood, Madison Avenue, and the way we tell stories. W. W. Norton & Company, New York, N.Y. London.

[233] Daniel M. Russell. 2019. The Joy of Search: A Google Insider's Guide to Going Beyond the Basics. The MIT Press. https://doi.org/10.7551/mitpress/11920.001.0001

[234] Tony Russell-Rose and Tyler Tate (Eds.). 2013. Designing the search experience: the information architecture of discovery. Elsevier, Morgan Kaufmann, Amsterdam.

[235] Marie-Laure Ryan. 1999. Immersion vs. Interactivity: Virtual Reality and Literary Theory. SubStance 28, 2 (1999), 110–137. https://doi.org/10.1353/sub.1999.0015

[236] Sandra Weniger and Claudia Loebbecke. 2011. Cognitive Absorption: Literature Review and Suitability in the Context of Hedonic IS Usage. https://mtm.uni-koeln.de/team-loebbecke-publications-conf-proceedings/Conf-144-2011-CognitiveAbsorptionAndTheUseOfHedonicIS.pdf

[237] Mustafa Tuncay Sarıtaş. 2015. The Emergent Technological and Theoretical Paradigms in Education: The Interrelations of Cloud Computing (CC), Connectivism and Internet of Things (IoT). Acta Polytechnica Hungarica 12, 6 (June 2015). https://doi.org/10.12700/APH.12.6.2015.6.10

[238] Carsten Schwede and Thomas Hermann. 2015. HoloR: Interactive mixed-reality rooms. In 2015 6th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Gyor, Hungary, 517–522. https://doi.org/10.1109/CogInfoCom.2015.7390647

[239] Abim Sedhain, Vaishvi Diwanji, Helen Solomon, Shahnewaz Leon, and Sandeep Kaur Kuttal. 2024. Developers' information seeking in Question & Answer websites through a gender lens. Journal of Computer Languages 79 (June 2024), 101267. https://doi.org/10.1016/j.cola.2024.101267

[240] Abim Sedhain, Shahnewaz Leon, Riley Raasch, and Sandeep Kaur Kuttal. 2023. Developers Foraging Behavior in Code Hosting Sites: A Gender Perspective. In Human Interface and the Management of Information, Hirohiko Mori and Yumi Asahi (Eds.). Vol. 14016. Springer Nature Switzerland, Cham, 575–593. https://doi.org/10.1007/978-3-031-35129-7_42 Series Title: Lecture Notes in Computer Science.

[241] William R. Sherman and Alan B. Craig. 2019. Understanding Virtual Reality. Elsevier. https://doi.org/10.1016/C2013-0-18583-2

[242] Shailesh Kumar Shivakumar. 2021. A Survey and Taxonomy of Intent-Based Code Search:. International Journal of Software Innovation 9, 1 (Jan. 2021), 69–110. https://doi.org/10.4018/IJSI.2021010106

[243] Cecilia Sik-Lanyi, Shervin Shirmohammmadi, Tibor Guzsvinecz, Boris Abersek, Veronika Szucs, Karel Van Isacker, Andrean Lazarov, Petya Grudeva, and Baris Boru. 2017. How to develop serious games for social and cognitive competence of children with learning difficulties. In 2017 8th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Debrecen, 000321–000326. https://doi.org/10.1109/CogInfoCom.2017.8268264

[244] Cecilia Sik-Lányi. 2014. Styles or Cultural Background does Influence the Colors of Virtual Reality Games? Acta Polytechnica Hungarica 11, 01 (Jan. 2014). https://doi.org/10.12700/APH.11.01.2014.01.7

[245] Susan Elliott Sim, Megha Agarwala, and Medha Umarji. 2013. A Controlled Experiment on the Process Used by Developers During Internet-Scale Code Search. In Finding Source Code on the Web for Remix and Reuse, Susan Elliott Sim and Rosalva E. Gallardo-Valencia (Eds.). Springer New York, New York, NY, 53–77. https://doi.org/10.1007/978-1-4614-6596-6_4

[246] Susan Elliott Sim and Rosalva E. Gallardo-Valencia (Eds.). 2013. Finding Source Code on the Web for Remix and Reuse. Springer New York, New York, NY. https://doi.org/10.1007/978-1-4614-6596-6

[247] Susan Elliott Sim, Medha Umarji, Sukanya Ratanotayanon, and Cristina V. Lopes. 2011. How Well Do Search Engines Support Code Retrieval on the Web? ACM Transactions on Software Engineering and Methodology 21, 1 (Dec. 2011), 1–25. https://doi.org/10.1145/2063239.2063243

[248] Richard Skarbez, Frederick P. Brooks, Jr., and Mary C. Whitton. 2018. A Survey of Presence and Related Concepts. Comput. Surveys 50, 6 (Nov. 2018), 1–39. https://doi.org/10.1145/3134301

[249] James Skripchuk, Neil Bennett, Jeffrey Zhang, Eric Li, and Thomas Price. 2023. Analysis of Novices' Web-Based Help-Seeking Behavior While Programming. In Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1. ACM, Toronto ON Canada, 945–951. https://doi.org/10.1145/3545945.3569852

[250] Mel Slater. 1999. Measuring Presence: A Response to the Witmer and Singer Presence Questionnaire. Presence: Teleoperators and Virtual Environments 8, 5 (Oct. 1999), 560–565. https://doi.org/10.1162/105474699566477

[251] Mel Slater, Vasilis Linakis, Martin Usoh, and Rob Kooper. 1996. Immersion, presence and performance in virtual environments: an experiment with tri-dimensional chess. In Proceedings of the ACM Symposium on Virtual Reality Software and Technology - VRST '96. ACM Press, Hong Kong, 163–172. https://doi.org/10.1145/3304181.3304216

[252] B. Sobota, Š. Korečko, P. Pastornický, and L. Jacho. 2016. Virtual-reality technologies in the process of handicapped school children education. In 2016 International Conference on Emerging eLearning Technologies and Applications (ICETA). IEEE, Stary Smokovec, Slovakia, 321–326. https://doi.org/10.1109/ICETA.2016.7802077

[253] Manuel Sojer. 2011. Open source software developers' perspectives on code reuse. In Reusing Open Source Code. Gabler, Wiesbaden, 20–130. https://doi.org/10.1007/978-3-8349-6390-1_3

[254] Manuel Sojer. 2011. Reusing Open Source Code. Gabler, Wiesbaden. https://doi.org/10.1007/978-3-8349-6390-1

[255] Raffaele Sperandeo, Lucia Luciana Mosca, Yari Mirko Alfano, Valeria Cioffi, Alfonso Davide Di Sarno, Anastasiya Galchenko, Daniela Iennaco, Teresa Longobardi, Enrico Moretto, Silvia Dell'Orco, Benedetta Muzii, and Nelson Mauro Maldonato. 2019. Complexity in the narration of the self A new theoretical and methodological perspective of diagnosis in psychopathology based on the computational method. In 2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Naples, Italy, 445–450. https://doi.org/10.1109/CogInfoCom47531.2019.9089986

[256] Kathryn T. Stolee. 2013. Solving the Search for Source Code. Ph. D. dissertation. University of Nebraska, Lincoln, Nebraska. https://digitalcommons. unl.edu/computerscidiss/64

[257] Kathryn T. Stolee and Sebastian Elbaum. 2012. Solving the search for suitable code: An initial implementation. Technical report 126. University of Nebraska-Lincoln. https://digitalcommons.unl.edu/cgi/viewcontent.cgi?article=1131&context=csetechreports

[258] Kathryn T. Stolee, Sebastian Elbaum, and Daniel Dobos. 2014. Solving the Search for Source Code. ACM Transactions on Software Engineering and Methodology 23, 3 (May 2014), 1–45. https://doi.org/10.1145/2581377

[259] J. Stylos and B.A. Myers. 2006. Mica: A Web-Search Tool for Finding API Components and Examples. In Visual Languages and Human-Centric Computing (VL/HCC'06). IEEE, Brighton, UK, 195–202. https://doi.org/10.1109/VLHCC.2006.32

[260] Jeffrey Stylos and Brad A. Myers. 2005. How Programmers Use Internet Resources to Aid Programming. (2005). http://www.cs.cmu.edu/~jsstylos/stylos-2005.pdf

[261] Weisong Sun, Chunrong Fang, Yifei Ge, Yuling Hu, Yuchen Chen, Quanjun Zhang, Xiuting Ge, Yang Liu, and Zhenyu Chen. 2023. A Survey of Source Code Search: A 3-Dimensional Perspective. (2023). https://doi.org/10.48550/ARXIV.2311.07107 Publisher: arXiv Version Number: 1.

[262] B. Katalin Szabo. 2019. Interaction in an immersive virtual reality application. In 2019 10th IEEE International Conference on Cognitive Infocommuni-cations (CogInfoCom). IEEE, Naples, Italy, 35–40. https://doi.org/10.1109/CogInfoCom47531.2019.9089957

[263] B. Katalin Szabó. 2022. Web search of software developers—Characteristics and tips. Frontiers in Education 7 (Aug. 2022), 908712. https://doi.org/10.3389/feduc.2022.908712

[264] Pawel Szczesny, Jan Nikodem, and Konrad Kluwak. 2016. Visual communication in expanding of human-computer interactions. In 2016 7th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Wroclaw, Poland, 391–396. https://doi.org/10.1109/CogInfoCom.2016.7804581

[265] Szilvia Szeghalmy, Marianna Zichar, and Attila Fazekas. 2013. Comfortable mouse control using 3D depth sensor. In 2013 IEEE 4th International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Budapest, Hungary, 219–222. https://doi.org/10.1109/CogInfoCom.2013.6719244

[266] Gergely Sziladi, Tibor Ujbanyi, and Jozsef Katona. 2016. Cost-effective hand gesture computer control interface. In 2016 7th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Wroclaw, Poland, 239–244. https://doi.org/10.1109/CogInfoCom.2016.7804555

[267] Gergely Sziladi, Tibor Ujbanyi, Jozsef Katona, and Attila Kovari. 2017. The analysis of hand gesture based cursor position control during solve an IT related task. In 2017 8th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Debrecen, 413–418. https://doi.org/10.1109/CogInfoCom.2017.8268281

[268] Philippe Tamla, Thilo Böhm, Kerstin Gaisbachgrabner, Jana Mertens, Matthias Hemmje, and Michael Fuchs. 2019. Survey: Software Search in Serious Games Development. In Proceedings of the 5th Collaborative European Research Conference (CERC 2019). Darmstadt, Germany, 155–166.

[269] Phitchayaphong Tantikul, C. Albert Thompson, Rosalva E. Gallardo-Valencia, and Susan Elliott Sim. 2013. Novel and Applied Algorithms in a Search Engine for Java Code Snippets. In Finding Source Code on the Web for Remix and Reuse, Susan Elliott Sim and Rosalva E. Gallardo-Valencia (Eds.). Springer New York, New York, NY, 271–287. https://doi.org/10.1007/978-1-4614-6596-6_14

[270] Suresh Thummalapenta and Tao Xie. 2007. Parseweb: a programmer assistant for reusing open source code on the web. In Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering. ACM, Atlanta Georgia USA, 204–213. https://doi.org/10.1145/1321631.1321663

[271] Janos Toth, Laszlo Kovacs, Balazs Harangi, Csaba Kiss, Andras Mohacsi, Zoltan Orosz, and Andras Hajdu. 2014. An online benchmark system for image processing algorithms. In 2014 5th IEEE Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Vietri sul Mare, Italy, 377–382. https://doi.org/10.1109/CogInfoCom.2014.7020482

[272] Tibor Ujbanyi. 2018. Examination of eye-hand coordination using computer mouse and hand tracking cursor control. In 2018 9th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Budapest, Hungary, 000353–000354. https://doi.org/10.1109/CogInfoCom.2018.8639882

[273] Medha Umarji and Susan Elliott Sim. 2013. Archetypal Internet-Scale Source Code Searching. In Finding Source Code on the Web for Remix and Reuse, Susan Elliott Sim and Rosalva E. Gallardo-Valencia (Eds.). Springer New York, New York, NY, 35–52. https://doi.org/10.1007/978-1-4614-6596-6_3

[274] Jolán Velencei, Agnes Szeghegyi, and Viktoria Szoboszlai. 2014. Informal Post-Experiential Learning. Acta Polytechnica Hungarica 11, 4 (April 2014). https://doi.org/10.12700/APH.25.04.2014.04.16

[275] Ville Pitkänen. [n. d.]. Utilization of virtual reality in Loviisa nuclear power plant. Master's thesis. Lappeenranta-Lahti University. https://lutpub.lut.fi/bitstream/handle/10024/161260/diplomityo_pitkanen_ville.pdf?sequence=1&isAllowed=y

[276] Finn Voichick. 2020. Exploring Usage of Web Resources Through a Model of API Learning. Master's thesis. Washington University in St. Louis. https://openscholarship.wustl.edu/eng_etds/513 Publisher: Washington University in St. Louis.

[277] Chong Wang, Xin Peng, Zhenchang Xing, Yue Zhang, Mingwei Liu, Rong Luo, and Xiujie Meng. 2023. XCoS: Explainable Code Search Based on Query Scoping and Knowledge Graph. ACM Transactions on Software Engineering and Methodology 32, 6 (Nov. 2023), 1–28. https://doi.org/10.1145/3593800

[278] Xian Wang, Ana M. Bernardos, Juan A. Besada, Eduardo Metola, and José R. Casar. 2015. A gesture-based method for natural interaction in smart spaces. Journal of Ambient Intelligence and Smart Environments 7, 4 (July 2015), 535–562. https://doi.org/10.3233/AIS-150325

[279] Yi Wang. 2017. Characterizing Developer Behavior in Cloud Based IDEs. In 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). IEEE, Toronto, ON, 48–57. https://doi.org/10.1109/ESEM.2017.27

[280] Ko Watanabe, Hacer Kübra Kücük, Sarah Gonzales, Andrew Vargo, Koichi Kise, and Shoya Ishimaru. 2023. Combining the Knowledge of Experienced Programmers to Extract Useful Web Resources for Solving Programming Tasks. In Asian HCI Symposium'23. ACM, Online Indonesia, 2–27. https://doi.org/10.1145/3604571.3604575

[281] David Weibel and Bartholomäus Wissmath. 2011. Immersion in Computer Games: The Role of Spatial Presence and Flow. International Journal of Computer Games Technology 2011 (2011), 1–14. https://doi.org/10.1155/2011/282345

[282] Ryen W. White and Resa A. Roth. 2009. Exploratory Search: Beyond the Query—Response Paradigm. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-031-02260-9

[283] Daniel Wigdor and Dennis Wixon. 2011. Brave NUI world: designing natural user interfaces for touch and gesture. Morgan Kaufmann, Burlington, Mass.

[284] Douglas Wightman. 2013. Search interfaces for integrating crowdsourced code snippets within development environments. Ph. D. Dissertation. Queen's University, Kingston, Ontario, Canada. https://static1.squarespace.com/static/519d10a2e4b090350a2b66a0/t/51ab859ce4b0be9ceada9488/1370195356614/Doug+Wightman+PhD+Thesis+As+Submitted.pdf

[285] Doug Wightman, Zi Ye, Joel Brandt, and Roel Vertegaal. 2012. SnipMatch: using source code context to enhance snippet retrieval and parameteri-zation. In Proceedings of the 25th annual ACM symposium on User interface software and technology. ACM, Cambridge Massachusetts USA, 219–228. https://doi.org/10.1145/2380116.2380145

[286] Bob G. Witmer and Michael J. Singer. 1998. Measuring Presence in Virtual Environments: A Presence Questionnaire. Presence: Teleoperators and Virtual Environments 7, 3 (June 1998), 225–240. https://doi.org/10.1162/105474698565686

[287] Claes Wohlin. 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering. ACM, London England United Kingdom, 1–10. https://doi.org/10.1145/2601248.2601268

[288] David Wong-Aitken. 2021. Investigating students' preferences and perceptions of online resources in an emergency-remote introductory programming course. Master's thesis. Simon Fraser University. https://summit.sfu.ca/_flysystem/fedora/2022-08/input_data/22439/etd21499.pdf

[289] David Wong-Aitken, Diana Cukierman, and Parmit K. Chilana. 2022. "It Depends on Whether or Not I'm Lucky" How Students in an Introductory Programming Course Discover, Select, and Assess the Utility of Web-Based Resources. In Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol. 1. ACM, Dublin Ireland, 512–518. https://doi.org/10.1145/3502718.3524751

[290] Xin Xia, Lingfeng Bao, David Lo, Pavneet Singh Kochhar, Ahmed E. Hassan, and Zhenchang Xing. 2017. What do developers search for on the web? Empirical Software Engineering 22, 6 (Dec. 2017), 3149–3185. https://doi.org/10.1007/s10664-017-9514-4

[291] Bowen Xu, Le An, Ferdian Thung, Foutse Khomh, and David Lo. 2020. Why reinventing the wheels? An empirical study on library reuse and re-implementation. Empirical Software Engineering 25, 1 (Jan. 2020), 755–789. https://doi.org/10.1007/s10664-019-09771-0

[292] Alexey Zagalsky. 2013. Investigating Opportunistic Software Development Using Social Media Recommendation System. Master's thesis. Tel-Aviv University. https://alexeyza.com/pdf/Zagalsky.Alexey-MSc.pdf

[293] Éva Orbán-Mihálykó, László Koltay, Ferenc Szabó, Péter Csuti, Renáta Kéri, and János Schanda. 2015. A New Statistical Method for Ranking of Light Sources based on Subjective Points of View. Acta Polytechnica Hungarica 12, 08 (Dec. 2015). https://doi.org/10.12700/APH.12.8.2015.8.11

[294] B. Katalin Szabó. 2022. Using a virtual reality headset in the simulation of the control room of a nuclear power plant. In 2022 IEEE 2nd Conference on Information Technology and Data Science (CITDS). IEEE, Debrecen, Hungary, 249–252. https://doi.org/10.1109/CITDS54976.2022.9914190

[295] B. Katalin Szabó and Attila Gilányi. 2020. The notion of immersion in virtual reality literature and related sources. In 2020 11th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Mariehamn, Finland, 371–378. https://doi.org/10.1109/CogInfoCom50765.2020.9237875

[296] Felicián Gergely, János Osán, B. Katalin Szabó and Szabina Török. Analytical performance of a versatile laboratory microscopic X-ray fluorescence system for metal uptake studies on argillaceous rocks. Spectrochimica Acta Part B: Atomic Spectroscopy 116, 75–84. https://doi.org/10.1016/j.sab.2015.12.007

[297] B. Katalin Szabó and Attila Gilányi. 2025. Definitions of immersion in virtual reality literature: a tertiary review (Manuscript)

[298] Jan-Noël Thon. 2008. Immersion Revisited: On the Value of a Contested Concept. In: Amyris Fernandez/Olli Leino/Hanna Wirman (Hg.): Extending Experiences. Structure, Analysis and Design of Computer Game Player Experience. Rovaniemi: Lapland University Press, 29–43. ISBN 9524841975

[299] Noirin Curran. Factors of immersion. 2017. The Wiley Handbook of Human Computer Interaction, https://doi.org/10.1002/9781118976005.ch13

[300] Zhang Chenyan.

Publications of the author, relevant to the dissertation

[Y1] **B. Katalin Szabó**. 2022. Web search of software developers—Characteristics and tips. Frontiers in Education 7 (Aug. 2022), 908712. https://doi.org/10.3389/feduc.2022.908712. IF: 2.3 SJR: Q2

Felicián Gergely, János Osán, **B. Katalin Szabó,** Szabina Török. 2016. Analytical performance of a versatile laboratory microscopic X-ray fluorescence system for metal uptake studies on argillaceous rocks. Spectrochimica Acta Part B: Atomic Spectroscopy Volume 116 75–84. https://doi.org/10.1016/j.sab.2015.12.007
SJR: D1. Quotations: according to Scopus: 10 Web of Science: 10

[Y2] **B. Katalin Szabó** and Attila Gilányi. 2025. How developers search for source code on the web -- a systematic mapping study. (manuscript, submitted to a journal)

[Y3] **B. Katalin Szabó** and Attila Gilányi. 2025. Definitions of immersion in virtual reality literature: a

[Y4] tertiary review. (Kézirat, folyóirathoz benyújtva)

Beatrix Katalin Szabó. 2019. Rigged hand model for the Blender Game Engine. Recent Innovations

[Y5] in Mechatronics (RIiM) Vol. 6, No. 1 (2019).https://doi.org/10.17667/riim.2019.1/5.

B. Katalin Szabó. 2022. Using a virtual reality headset in the simulation of the control room of a nuclear power plant. In 2022 IEEE 2nd Conference on Information Technology and Data Science

[Y6] (CITDS). IEEE, Debrecen, Hungary, 249–252. https://doi.org/10.1109/CITDS54976.2022.9914190

B. Katalin Szabo. 2019. Interaction in an immersive virtual reality application. In 2019 10th IEEE International Conference on Cognitive Infocommuni-cations (CogInfoCom). IEEE, Naples, Italy, 35–40. https://doi.org/10.1109/CogInfoCom47531.2019.9089957

[Y7] Quotations: according to Scopus: 9 Web of Science: 5

B. Katalin Szabó and Attila Gilányi. 2020. The notion of immersion in virtual reality literature and related sources. In 2020 11th IEEE International Conference on Cognitive Infocommunications (CogInfoCom). IEEE, Mariehamn, Finland, 371–378. https://doi.org/10.1109/CogInfoCom50765.2020.9237875
Quotations: according to Scopus: 19 Web of Science: 12

Other publications of the author

[Y7] Házi Gábor, Jánosy János Sebestyén, Seregi László, **Szabó B. Katalin**. 1997. A GRASS szimulációs környezet és alkalmazása a Paksi Atomerőműben (The GRASS simulation environment and its application in the Paks Nuclear Power Plant), Elektron. technol., microtech. (1-2) 38-43. 1997. ISSN 0236-8676

[Y8] A. Jávor and B. **K. Szabó**. 1985. Switch Level Extension of a Logic Simulator. In Simulation in Research and Development, Proceedings of the IMACS European Simulation Meeting on Simulation in Research and Development, Eger, Hungary, 27–30 August, 1984. Elsevier Science Publishers, , ISBN 0-444-87747-9

[Y9] **B. K. Szabó**. 1988. Part-Task Simulator for a WWER-440 Nuclear Power Plant Subsystem, KFKI-1988-34/G,M Preprint
Online: https://inis.iaea.org/search/search.aspx?orig_q=RN:20019725

[Y10] F. Adorján, L. Bürger, V.V. Ivanov, I. Lux, L. Meskó, A.A. Mozhaev, **K. Szabó,** J. Végh, V.V. Yakovlev. 1990. EMERIS, an advanced information system for a materials testing reactor, KFKI-1990-24/G Preprint, Online: https://inis.iaea.org/search/search.aspx?orig_q=RN:22016833 HU ISSN 0368 5330

[Y11] F. Adorján, L. Bürger, V. V. Ivanov, I. Lux, L. Meskó, A. A. Mozhaev, **K. Szabó**, J. Végh, V. V. Yakovlev, 1990. EMERIS, an advanced information system for a materials testing reactor, Proceedings of the IAEA/OECD International Symposium on Balancing Automation and Human Action in Nuclear Power Plants, IAEA-SM-315, pp. 223–234, Munich, FRG (9–13 July, 1990) Online: https://inis.iaea.org/collection/NCLCollectionStore/_Public/22/063/22063655.pdf?r=1&r=1

[Y12] J. S. Janosy, Zs. Kiss, **K. Szabo**. 1991. Neutron Flux Controller Model for Compact Simulators, International Atomic Agency Working Group on Nuclear Power Plant Control and Instrumentation, Specialists' Meeting, Balatonfüred, Hungary, September 24–27, 1991. pp. 329-338

[Y13] F. Adorján, L. Bürger, V. V. Ivanov, I. Lux, L. Meskó, A. A. Mozhaev, **K. Szabó**, J. Végh, V. V. Yakovlev: Advanced operator support system (EMERIS), including automatic disturbance analysis for a materials testing reactor, Proceedings of the NEACRP Specialists' Mtg. on In-Core Instrumentation and Reactor Core Assessment, Pittsburg, USA (1-4 October, 1991). pp. 285-295 https://www.oecd-nea.org/upload/docs/application/pdf/2019-12/nea1832-ic91.pdf sts' Meeting, Balatonfüred, Hungary, September 24–27, 1991. pp. 329-338

[Y14] **B. K. Szabó**: Human Factors/Ergonomics in Software Development. 1994. Proceedings of the ERŐFI II Autumn School on Reactor Physics, 7–10 November, 1994, Lillafüred, KFKI-1995-11/G,

[Y15] pp. 245–258. https://inis.iaea.org/collection/NCLCollectionStore/_Public/27/002/27002197.pdf
J. S. Jánosy, T. Bogdán, G. Házi, **B. K. Szabó**: Generalizing and Upgrading a Set of Simulation Tools to a Full-Graphic Environment, Proceedings of the 1995 Simulation Multi Conference, April 9–13, Phoenix, Arizona, 1995. SCS Publication, ISBN 1-56555-049-8

[Y26] J. S. Jánosy, G. Házi, L. Seregi, **B. K. Szabó**: GRASS - the Graphic Simulation System. 1997. Proceedings of the 2nd CSNI Meeting on Simulators and Plant Analyzers Hanasaari, Espoo, Finland,, pp. 103–112 Online: https://www.oecd-nea.org/nsd/docs/1997/csni-r1997-37.pdf

[Y27] E. Végh, G. Házi, J. S. Jánosy, G. Mayer, L. Seregi, **B. K. Szabó**. 2001. Graphical Simulator of the ETTR-1 Research Reactor, Modelling And Simulation 2001, 15th European Simulation Multiconference, Prague, June 2001. In: Kerckhoffs, EJH; Snorek, M (ed.) Modelling and simulation 2001: 15th European Simulation Multiconference, ESM 2001, Delft, The Netherlands. Society for Computer Simulation. pp. 373-380.