

NOTE: This is the authors' version of the article that has been submitted to ACM Computing Surveys.

Toward Improving Binary Program Comprehension via Embodied Immersion: A Survey

DENNIS BROWN, EMILY MULDER, and SAMUEL MULDER, Auburn University Department of Computer Science and Software Engineering, USA

Binary program comprehension is critical for many use cases but is difficult, suffering from compounded uncertainty and lack of full automation. We seek methods to improve the effectiveness of the human-machine joint cognitive system performing binary PC. We survey three research areas to perform an indirect cognitive task analysis: cognitive models of the PC process, related elements of cognitive theory, and applicable affordances of virtual reality. Based on common elements in these areas, we identify three overarching themes: enhancing abductive iteration, augmenting working memory, and supporting information organization. These themes spotlight several affordances of VR to exploit in future studies of immersive tools for binary PC.

CCS Concepts: • **Software and its engineering** → **Software reverse engineering**; • **Human-centered computing** → **Virtual reality**; **Interaction techniques**; • **Security and privacy** → *Malware and its mitigation*.

Additional Key Words and Phrases: binary program comprehension, binary reverse engineering, embodied immersion, immersive analytics, virtual reality, cognitive models, cognitive systems engineering, augmented cognition

ACM Reference Format:

Dennis Brown, Emily Mulder, and Samuel Mulder. 2023. Toward Improving Binary Program Comprehension via Embodied Immersion: A Survey. 1, 1 (April 2023), 27 pages. <https://doi.org/XXXXXXXX.XXXXXXX>

1 INTRODUCTION

Understanding how an existing software program operates is critical to many tasks ranging from maintaining and improving that software to mitigating potential malicious impacts of that software. We call this process *program comprehension (PC)*: comprehending of the purpose and effects of the program. While much of the research in program comprehension primarily focuses on source code written in high-level languages, many of the same concepts and processes can be applied to understanding binary programs that lack the original source code. Comprehending non-trivial binary programs—*binary PC*¹—is quite difficult. Many tools and approaches have been developed over time to expedite the process, e.g., disassemblers, decompilers, profilers, debuggers, etc., and more ambitious projects employing artificial intelligence [43] [1] [64] [24] [53]. However, successfully understanding a complex binary program still ultimately depends on the expertise of a human in the loop, one with rare knowledge and experience.

¹This paper will use “binary PC” to refer to the process of comprehending or understanding, a binary program at the assembly language level. This process is also called “reverse engineering” or “program understanding” of binary programs in some references, and for the purposes of this work, should be considered synonymous with “binary PC.”

Authors' address: Dennis Brown, dgb0028@auburn.edu; Emily Mulder, semulder@gmail.com; Samuel Mulder, szm0211@auburn.edu, Auburn University Department of Computer Science and Software Engineering, 1301 Shelby Center, Auburn, Alabama, USA, 36849.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2023 Association for Computing Machinery.

Manuscript submitted to ACM

Manuscript submitted to ACM

1

One key to improving a human-centric process is understanding its human element and how the human interacts with the system. Cognitive Systems Engineering (CSE) considers holistically the combination of the operator and the machine—the joint cognitive system—and how to make that joint system more effective by reducing cognitive complexity for the human. In this approach, all work performed by humans is a form of cognitive work [46]. The work of binary PC includes many disparate cognitive processes. Notable prior research of the cognitive processes involved in performing binary PC casts it as a sensemaking task [13], with more nascent research studying binary PC at the level of cognitive modeling [83] [54] [30] and applying Cognitive Load Theory (CLT) [76]. Using the findings from our literature survey, we perform an indirect cognitive task analysis [21] of binary PC to identify its unique cognitive skills and demands.

We assert that if we can reduce the cognitive complexity of binary PC tasks via the employment of immersive interaction, then the overall outcomes of the joint cognitive system performing binary PC can be improved. Employing embodied cognition is posed as one method of improving cognitive processes [88], and Virtual Reality (VR) is an accessible means of providing embodied user experiences. In particular, Immersive Analytics (IA) [15] [31] [33] is an emergent and promising area of research that uses innovative display and interaction techniques to aid humans in the understanding of complex analytic problems by providing an egocentric experience. IA may provide the conceptual building blocks, applied in a CSE framework, for significantly improving the task of binary PC.

Toward the improvement of cognitive processes in binary PC using immersive experiences, this work seeks to answer the following Research Questions (RQ):

- RQ1. What are common and significant characteristics of mental/cognitive models employed by binary PC practitioners?
- RQ2. What are effective techniques for understanding and improving cognition in the context of binary PC?
- RQ3. How can the affordances of immersive interaction with virtual reality be employed to improve cognition in analytic tasks?
- RQ4. How can we use these findings to effect improvements in the practice of binary PC?

This work reviews and ties together research contributions from multiple disciplines in order to answer the RQs. These disciplines include Reverse Engineering and Program Comprehension; Cognition and Cognitive Systems; Visualization; VR; and IA. The papers covered in this survey were identified through an iterative approach. We first performed exploratory ad-hoc searches of the literature using common scholarly search engines and indices looking for directly-related results in the primary topic area of using VR to improve binary PC. While we found very few close matches, we did find many adjacent works and through a clustering process, we identified the disciplines listed previously. We then iterated this search process until we filled out a set of recent and relevant research papers.

This paper proceeds as follows. Section 2 dives into what makes binary PC difficult. In Sections 3 and 4, we explore prior research in cognitive models of PC (primarily binary and adjacent forms of PC) and the cognitive theory that relates to those models in order to answer RQ1 and RQ2. In Section 5, we explore how affordances in visualization and immersion have been applied toward similar analytic problems to answering RQ3. Finally, we analyze the findings, address RQ4, and consider future work in Section 6.

2 WHY IS BINARY PC DIFFICULT?

Comprehending the actions and purpose of an arbitrary binary program is a complex task. To support our cognitive task analysis—identifying the cognitive skills and mental demands of performing a task [56]—and before we dive into

the cognitive models and related theory in upcoming sections, we should identify *why* binary PC is inherently difficult. This analysis supports the first of three principles of CSE per Hollnagel and Woods, which is to identify the problem areas of the task. [46].

First, this task is very unlikely to be fully automated. Rice’s Theorem [68] implies that deciding whether a given binary program contains any non-trivial property is formally undecidable. In practice, with intermingled code and data on modern architectures, there are many common code patterns that are unresolvable. Even with state-of-the-art tools (including Large Language Models), the process of disassembly introduces uncertainty and error. Each additional property built on top of disassembly is thus both undecidable and built on a representation including that uncertainty. Human input is required to overcome ambiguous situations and determine when a solution is good enough to satisfy the current objective, and even with human input our ability to answer questions about binary programs is severely limited.

Second, understanding what properties we care about in a program is hard; e.g., how do we know if there is a hidden trigger or a backdoor? Even given a source representation, these questions are challenging to answer, often reducing to a question of program correctness. Since programs rarely have a detailed formal specification, the *correct* behavior may be an open question. Reverse engineering a piece of software is an iterative, hypothesis-driven process, requiring the analyst to continually reformulate and ask questions relative to the current state of knowledge.

Finally, programs are arguably the most complex things ever engineered by humans [22]. In building complex software, the field of software engineering has developed many models for managing abstraction and interfaces between different components to allow many programmers to design a program as a team. Without that infrastructure in place, an entire non-trivial program is too complex for a human mind to fully understand. This is compounded by the fact that the compilation process strips helpful comments, structures, and metadata, leaving the analyst to try and recreate the layers of abstraction that the original authors used to manage the complexity.

Next, we will review mental/cognitive models used by analysts to overcome these difficulties when performing binary PC.

3 CHARACTERISTICS OF MENTAL/COGNITIVE MODELS OF BINARY PC

In this section and the next, we examine cognitive elements of binary PC in two phases. In this first phase, we present findings regarding the cognitive process of binary PC. Binary PC shares many concepts and theory with source code PC, which has been more extensively covered in the literature, so we will include some applicable work in source code PC in this section. These sections support the second of the three principles of CSE, which is to understand the circumstances of the problems identified per the first principle, focusing on potential causes and influences of those problems. [46]

3.1 Discovering Mental/Cognitive Models of Binary PC

The *mental model*, or cognitive model, concept originated with Craik [20], who asserted that humans build internal representations of the external world and use them to reason about why things happen and anticipate what might happen next. Johnson-Laird’s [48] research in experimental psychology corroborated Craik’s claims by finding that people employ mental models in their working memory to perform reasoning. [67] Cognitive models can be particularly powerful because they combine elements of both the problem domain or task area and the characteristics of human cognition. [5]

Shneiderman and Mayer [73] developed an early cognitive model of how programmers build up an internal semantic representation of a program based on evidence discovered in a series of experiments they performed. They asserted

that the model must describe common programming tasks (composition, comprehension, debugging, modification, and learning) in terms of cognitive structures that programmers form in their memories and cognitive processes to use and build that knowledge. Knowledge is categorized as semantic (general concepts independent of language) or syntactic (“precise, detailed, and arbitrary” details, primarily language-specific). They cast the task of PC as subtasks of debugging, modification, and learning in which the programmer uses syntactic knowledge to form a multi-level internal semantic representation of the program. Lower levels (e.g., sequences of operations) and higher levels (what the program does) can be formed independently, and this encoding process is similar to Miller’s chunking process [57], where smaller chunks of statements join to form larger chunks. The internal semantic representation of the program is strongly retained and widely accessible. Figure 1 illustrates their concept.

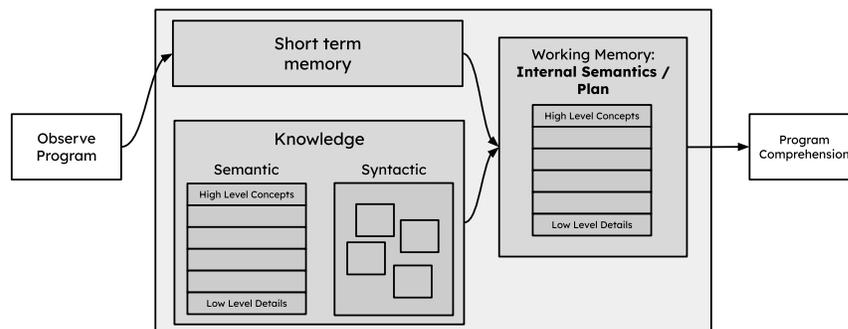


Fig. 1. PC process per Shneiderman and Mayer [73]

Introducing an iterative model, Brooks [12] envisioned the PC process as forming a mental tree that branches into increasingly complete hypotheses of how the code works and what it does. As a branch is found to be incorrect, the programmer cuts that branch and backtracks and may follow another higher-level branch or add a new one. This process is essentially abductive reasoning applied to code understanding, described by Weigand and Hartung [86] as a process of making observations, forming hypotheses, creating mental models of code that support the hypotheses, and searching for information to prove or disprove the hypotheses.

Zayour and Lethbridge [89] performed a cognitive analysis to identify the mental steps of source-code-based PC. In their analysis, the language was low-level proprietary code and not a typical high-level language on which PC is performed, increasing the applicability of their results to binary PC. They identified two primary cognitive difficulties of disorientation following recursions and disorientation in understanding the most relevant execution paths in the code. In response, they proposed two high level cognitive design requirements. The first is to minimize how many artifacts the engineer needs to keep in working memory by maintaining visual proximity between artifacts; linking new and existing information with meaningful encoding or chunking; and facilitating backtracking in execution paths. The second is to minimize fading of working memory by reducing the time artifacts need to be maintained in working memory and minimizing the number of steps between artifact acquisition. They used their findings to drive implementation of a PC assistive process in their DynaSee tool, which performs several filtering steps on program execution (remove redundancies, detect patterns, rank code routines) before visualizing trace patterns.

The Data-Frame Theory was established by Klein et al. [50] in which a frame—an explanatory structure such as a story, map, script, or plan that relates entities to other entities—is simultaneously fitted to discovered data and

also drives the discovery of further data. The frame provides a foundation for understanding until flaws are detected. *Sensemaking* happens when the frame is re-fitted to new evidence in response to those flaws, and it is enabled by abductive reasoning [29]. The approach to modeling taken by Bryant et al. [13] is to frame binary PC as a sensemaking task, in which the engineer develops a mental model or hypothesis about the situation and its constituent elements, and adapts that model through iterations of goal-directed information seeking. Through a process of cognitive task analysis and verbal protocol analysis, they identified nine sensemaking functions in binary PC and developed a state machine representing the process, represented in Figure 2. Within their model, Bryant et al. included both procedural and declarative knowledge. Procedural knowledge consists of stored patterns of interaction. They categorized the declarative, or factual, knowledge into twelve subdomains covering the fundamental training and experience needed by reverse engineers (programming, debugging, program loading and execution, instruction sets, etc.). Additional declarative knowledge comes from abstract and concrete causal relationships within the data, e.g., constraints on prior knowledge or predicates applied to constants. They explored codifying declarative and procedural knowledge as an implied state machine via production rules implemented in the ACT-R [69] cognitive architecture.

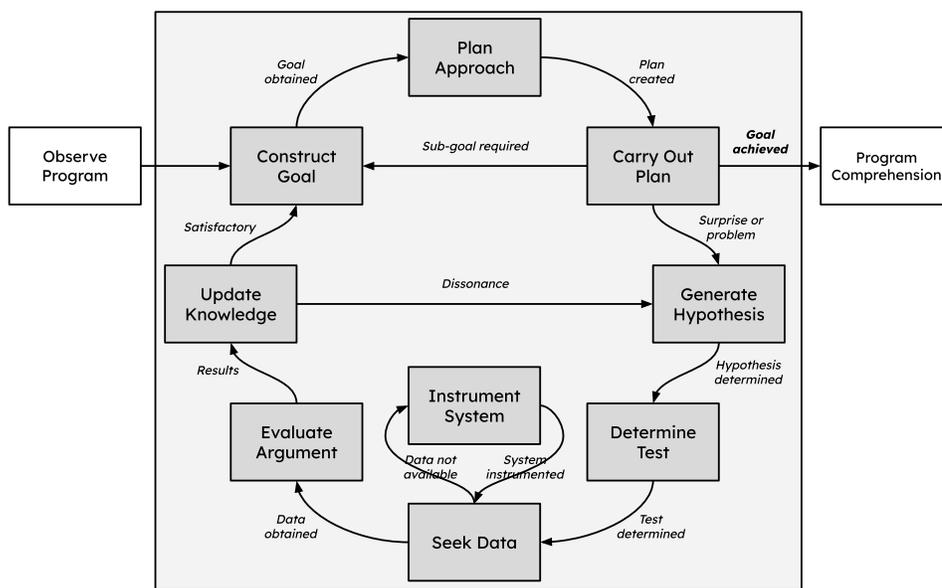


Fig. 2. PC process based on sensemaking; adapted from Bryant et al. [13]

To facilitate and encourage collaboration through communication of information in analysts' mental models, Tenor [80] surveyed and analyzed cognitive aspects of binary PC and proposed that the binary reverse engineering community build vocabularies and ontologies. In this vein, Sisco et al. [75] sought to mathematically formalize Bryant et al.'s concept of binary PC as a sensemaking task and developed a supporting ontology. They asserted that reverse engineers analyze programs using four foundational patterns: navigation (searching for items, e.g., beacons²); translation (determining how the code would be implemented in a higher-level language); experimentation (deducing how program

²The *beacon* is one important concept in the cognitive process identified by Brooks, which is a "set of features that typically indicate the occurrence of certain structures or operations within the code." [12]. An example of a beacon is a block of code that interchanges values within an array inside of a loop, which strongly indicates a sorting function. Programmers seek beacons to confirm their hypotheses in the abductive reasoning process.

values change over different flows); and elaboration (identifying and explaining the major components and properties of the program). Reverse engineers use these patterns to build knowledge as interrelated mental objects with various constraints on their relationships. They proposed an ontology for representing this knowledge composed of *ologs*: category-based types, aspects, and facts; this representation allows commutation and a higher level of expressiveness than is supported by common alternatives such as Web Ontology Language (OWL) and Resource Description Framework (RDF). The team generated ologs for fundamental assembly instructions, program data, control flow, and operating system events, and used those ologs to formalize information flow in the experimentation pattern. The team theorized that this work provides the basis for a computational framework to model cognitive tasks in binary PC, and that model can further inform the actions of an analyst or agent encoded in ACT-R.

In a push toward building automated agents to assist with binary PC, Dudenhofer [29] proposed the Cognitive Understanding of Reverse Engineering (CURE) model to capture sensemaking steps in binary PC. The model, shown in Figure 3, is founded on the iterative cycle of abductive reasoning and experimentation: form a hypothesis; explore to find information to prove or disprove a hypothesis; recognize cues in the code or related artifacts, e.g., beacons, and store them in working memory; use that information to refine the hypothesis or form the next hypothesis; and repeat the cycle. The model inspired an application, the CURE Assistant, which works with an existing binary reverse engineering framework to identify code snippets matching those in an extensible catalog of recipes and present possible program behaviors to the analyst.

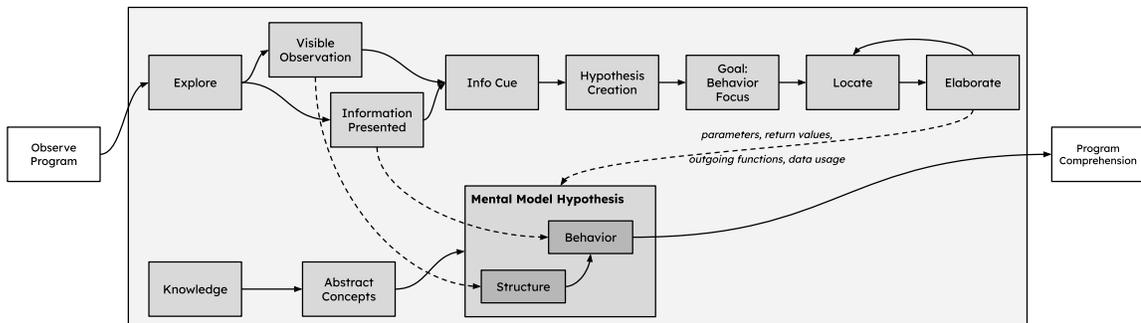


Fig. 3. PC process based on abductive iteration; adapted from Dudenhofer [29]

Votipka et al. [83] presented a three-phase process model for binary PC: (1) Examining and executing the full program for an overview, then choosing focus areas; (2) Reviewing specific program slices chosen in the first phase, scanning for beacons and data and control flows, and generating specific hypotheses; and (3) Inspecting lines of assembly code and traces to test the hypotheses, as shown in Figure 4. They also identified several categories of beacons for use across the phases: APIs and strings across all phases; UI elements in the first phase; and constants, variable names, control flow structures, compiler optimizations, function prototypes, and program flow in the second and third phases. In developing this process, they identified one significant similarity between binary PC and source-code-based PC: mental simulation of code execution; and two primary differences between binary PC and source-code-based PC: binary PC involves more overview while source-code-based PC is more focused and pinpointed, and binary PC uses a more diverse set of beacons, common recognizable schema or patterns. Their work culminated in five guidelines for designing RE tools: (1) Match interaction with analysis phases; (2) Present input and output in the context of code; (3) Allow data transfer between static and dynamic contexts; (4) Allow selection of analysis methods; and (5) Support readability improvements.

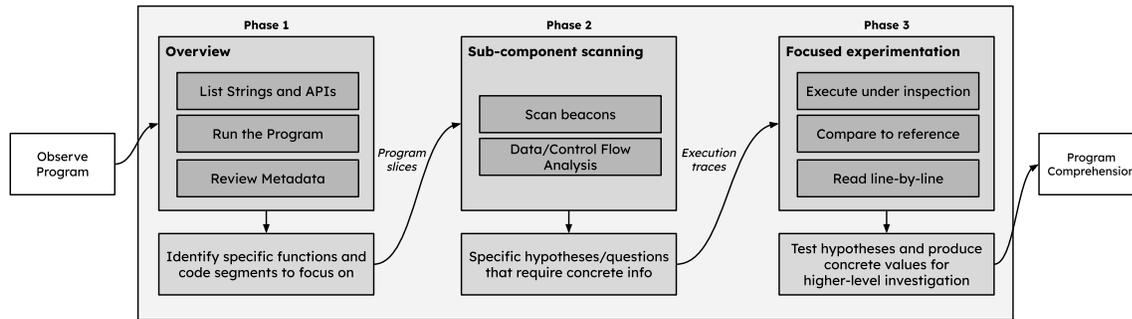


Fig. 4. PC process in three phases, adapted from Votipka et al. [83]

Nyre-Yu et al. [61] conducted a task analysis of static binary PC. They found evidence reinforcing previous findings that binary PC and source-code-based PC share similarities in cognitive processes: identifying goals and plans, creating hypotheses, and exploring to gather information. They asserted that analysts very commonly ask “where is this used in the code?”, “where is the method being called?”, “how can I get calling information?”, “where does this information/data go?”, “where is the data coming from?” and “what is the context of this vulnerability/code?” [61]. They also observed the revisiting of past states reported by Votipka et al. [83], however, they did not find common patterns across participants. The team also identified repetitive actions that are targets for automation: accessing library function documentation and accessing the task definition.

3.2 Differences between Novices and Experts in Performing Binary PC

Besides the mental models associated with binary PC, it is important to understand differences between novices and experts. While both classes of analysts use many of the same elements of the mental model, the approaches they take are different.

Looking toward source-code-based PC again as a close analog of binary PC, Vessey [82] studied experts and novices in a code comprehension and debugging task, observing that while both experts and novices employ breadth-first approaches, experts apply a system view that novices do not; novices also employ depth-first approaches while experts do not. Additionally, experts perform chunking effectively to proceed steadily through a program, while novices perform chunking much less effectively, resulting in jumping around within a program to understand it. Storey’s survey [77] found that experts use more external memory aids [27], and that novices focus mainly on objects while experts also consider functional relationships and algorithms. Siegmund et al. [74] asserted there are two basic models of PC: top-down and bottom-up. Rather than distinguishing between “expert” and “novice,” they focused on level of domain knowledge. Analysts who have domain knowledge they can apply to the program use the top-down process and use beacons to form hypotheses. Otherwise, without domain knowledge, they use the bottom-up process to analyze the program line-by-line.

Cowley [19] conducted a job analysis to identify performance predictors for binary PC practitioners. In addition to situational (team and organizational) predictors, Cowley identified individual predictors for novices and experts. From those predictors, the author determined five milestones along the progression from novice to expert. Milestones to the intermediate level include (1) proficiency in relevant tools; (2) significant reduction in assistance needed to complete tasks; and (3) parity with experts in identification and employment of binary PC strategies. The final two

milestones complete the transition to a true expert, including (4) organizational recognition through promotion; and (5) establishing a track record of solving binary PC problems without assistance.

Mantovani et al. [54] studied nearly 300 hours of activity performed by 72 novice and expert reverse engineers performing a static analysis task. They observed that most often, novices move forward from `main()` and jump around between code blocks, visiting some more than once, while experts move both forward and backward from `main()` and move more linearly through code blocks. Additionally, experts are quicker to identify what they can ignore.

3.3 Section Summary

In this section, we surveyed evidence in support of answering RQ1, “What are common and significant characteristics of mental/cognitive models employed by binary PC practitioners?” We examined a progression of research contributions in this area, starting with the basic conceptualization of a mental or cognitive model. Early research in mental models for software development proposed an internal semantic representation and chunking process, then progressed to include iterative hypotheses with abductive reasoning. Researchers then formalized the sensemaking process of goal-oriented iteration and adaptation of the analyst’s emergent mental model, and further developed the field by creating an ontology and introducing automation.

This progression of research yielded a set of common concepts. While researchers found that the process of binary PC is crafted uniquely by each individual, especially at the expert level, we identify the following themes and salient points regarding modeling the cognitive activities involved in binary PC:

- Uses multi-level internal semantic representation of the program [73]
- Follows iterative pattern of sensemaking / abductive reasoning
 - Sets goals and follows plans [13] [61]
 - Forms increasingly complete hypotheses [12] [86] [29] [83]
 - Tests hypothesis through experimentation [13] [75] [29] [83]
 - Updates framing of the problem based on results [50] [13] [29] [83]
- Creates disorientation following recursions and execution paths [89]
- Taxes working memory [73] [89]
- Requires declarative and procedural knowledge retrieval and generation [13]
- Uses translation—determining how the code would be implemented in a higher-level language [75]
- Uses beacons; beacons for binary PC are more diverse than for source-code-based PC [12] [29] [83]
- Relies upon overview of the binary executable [83]
- Uses external memory aids [27] [77]
- Relies upon determining what to ignore [54]
- Experts use a breadth-first approach with system thinking; novices, both depth- and breadth-first without system thinking [82]
- Experts use a top-down approach if the program domain is familiar, otherwise, bottom-up [74]

Continuing our effort to understand the circumstances of the problems of binary PC, we next look at findings in the literature about underlying cognitive theory with the aim of augmenting or improving the cognitive activities of binary PC.

4 UNDERLYING COGNITIVE THEORY AND APPLICATIONS TO BINARY PROGRAM COMPREHENSION

Now that we have a baseline understanding of the characteristics of mental/cognitive models behind binary PC, we explore possible avenues for enhancing the abilities of practitioners. In this section we briefly review the underlying theory behind external and embodied cognition along with cognitive load, and then consider methods that may reduce cognitive load or otherwise optimize the cognitive processes of binary PC practitioners.

4.1 External and Embodied Cognition

Cognition involves many specific and interdependent processes: Eysenck and Brysbaert [34] identified attention; perception; memory; learning; reading, speaking, and listening; and problem-solving, planning, reasoning, and decision-making; each with their own design implications. Norman described cognition as multimodal—that there are many different types of thinking—and specifically addressed experiential cognition and reflective cognition [60]. Experiential cognition is when a person is experiencing and responding to the environment without the need for significant mental effort (which also happens with extensive expertise), while reflective cognition happens when a person is putting substantial thought into considering and making decisions or forming new ideas. Norman further discussed both forms in relation to effective tools for enhancing one or the other, such that a tool designed to aid reflective thought is inappropriate for experiential cognition and vice versa [60]. Cognition also occurs in varied places, times, and situations, and there is movement in the field toward understanding cognition *in situ*—rather than limiting the scope of reasoning about cognition to what occurs in the mind, considering how the environment can improve and affect cognition [67].

External cognition, per Scaife and Rogers [71], concerns the cognitive interaction with external manifestations of knowledge in the environment, e.g., images, video, virtual reality, etc. The primary cognitive benefits of external cognitive activities include using external knowledge representations (e.g., notes and reminders) to reduce memory load; using computational tools (e.g., calculators) to make tasks easier; and annotating and reordering or restructuring external representations of knowledge (e.g., checking off to-do lists or arranging desktop icons) [67]. In considering the impact of the environment on cognition, Gibson coined the term *affordance* as a “specific combination of the properties of its substance and its surfaces” perceived in relation to the viewer [37]. Norman further elaborated that an affordance is most importantly constrained by properties that indicate how something could be used: “When affordances are taken advantage of, the user knows what to do just by looking: no picture, label, or instruction needed.” [59]. Affordances leverage internal cognition to enable the phenomena of external cognition and embodied cognition.

Embodied cognition [81], closely linked to dynamical systems theory, proposes that cognition happens in real time, with the brain simultaneously receiving input from, processing, and interacting within and with the nervous system, physical body, and external environment. In this theory, cognition is deeply impacted by sensorimotor interaction with the environment and is profoundly grounded in the ability to act [38]. Shapiro and Spaulding’s *Embodiment Thesis* states: “Many features of cognition are embodied in that they are deeply dependent upon characteristics of the physical body of an agent, such that the agent’s beyond-the-brain body plays a significant causal role, or a physically constitutive role, in that agent’s cognitive processing.” [72]

In line with the concept of the joint cognitive system of human and machine described earlier, Kirsh [49] made the case that tools are absorbed into the body schema and change how we think, that we think with not just our minds but also our bodies (enabled by kinesthetic perception), and that we even think with tools. Similarly, Hornecker et al. [47] presented the position that our sensorimotor interactions with, and manipulation of, the world develop our capacity for abstract thought, starting with simple concepts such as in/out, over/under, up/down — that body and mind are

inseparable even in the domain of abstract thought. Ale et al. [4] suggested that embodied memory is a potentially rich research area, in which physical objects or locations serve as memory palaces; and that whole-body stimuli can expedite storage and retrieval of memory (e.g., using physical motion or aroma triggers to encode and store memories).

4.2 Cognitive Load Theory

Cognitive Load Theory (CLT), per Sweller and Chandler [78], assumes that humans take in and process information through two main channels—through hearing and through visualizing. Further, only a finite amount of processing can occur in each channel at any point in time. The magnitude of the cognitive load depends not only on sheer number, but on how much interactivity occurs between elements. The ability to process this information is limited by working memory, but can be improved through employing a suitable schema with which to leverage long-term memory and reduce cognitive load. Cognitive load theory currently incorporates three categories: *intrinsic* load inherent in the cognitive task at hand; *extraneous* load caused by how information is presented; and *germane* load. Germane load represents the ability of our mind to connect what we are learning with long term memory, and linked to intrinsic, is the demand on our cognition of using or focusing working memory for intrinsic learning [79]. Paas et al. [63] asserted that cognitive performance will degrade at either end of the load spectrum (underloading or overloading).

Hollender et al. [45] surveyed 65 papers on cognitive load related to human-computer interaction. They cataloged methods to leverage phenomena to reduce extraneous cognitive load as follows. The worked example effect is from learning from studying solved sample problems. The split-attention effect is from presenting information from multiple visual sources in an integrated way to reduce load required to perform mental integration. The modality effect is from presenting multiple information sources through different modalities (e.g., visual and aural) to allow the inputs in each modality to be processed simultaneously. The redundancy effect is from reducing the level of redundant information presented in different modalities/sources, thereby reducing the load of reconciling the underlying concepts across the inputs from those modalities. They also reviewed methods to increase germane cognitive load and foster schemata development (most applicable in educational settings, and a way to increase the capacity of working memory): specifically introducing a variety of tasks; linking concrete information to abstract concepts; and self-explanation. Also, in an educational context, they reviewed methods to adjust intrinsic cognitive load via adjusting the sizes and quantity of information chunks presented over time.

4.3 Applications to Program Comprehension

Next, we will review contributions from studies applying underlying cognitive theory to PC and closely-related tasks. Helgesson and Runeson [42] studied cognitive load drivers in software engineering and proposed a set of perspectives with which to reason about these drivers. The *Task* perspective accounts for CLT’s intrinsic load, which results from the inherent cognitive intensity of software engineering. The *Environment* perspective accounts for CLT’s germane load, which results from constructing mental schemata for processes and tools, plus additional load from re-learning new processes and tools that are meant to replace old ones, but can set up competing mental schemata. The remaining perspectives comprise CLT’s extraneous load: *Structural* (e.g., technical debt); *Information* (e.g., poor/missing code documentation); *Tool* (e.g., friction from unintuitive, cumbersome, or unreliable tools), *Communication* (e.g., lack of communications amongst the development team), *Interruption* (the cognitive cost due to resumption lag), and *Temporal* (e.g., tracing a component’s change history in version control).

In observing how programmers comprehend code, Siegmund et al. [74] took a novel approach by employing functional magnetic resonance imaging (fMRI) to find the brain regions activated during source-code-based PC. In a controlled

study of 17 computer science undergraduates, designed to elicit bottom-up source code comprehension and minimize extrinsic cognitive load, they observed activation of five Brodmann areas (of 52 Brodmann areas associated with cognitive process) [11] using fMRI. Those five areas are associated with division of attention, silent word reading, verbal/numerical working memory, and problem solving. Based on further analysis, Siegmund et al. theorized that bottom-up PC uses two areas for keeping values in mind, another area for analyzing words and symbols, and the remaining two areas for integrating statements and chunks.

Smits [76] performed a study of reducing cognitive load in reverse engineering. Under the assumption that the complexity and volume of outputs from RE tools induces cognitive overload, Smits implemented two primary techniques for managing cognitive overload: (1) *Information Filtering* to remove extraneous information so as to reduce extraneous load; and (2) *Information Organization* to organize data in a manner familiar to the user to reduce germane load. The two techniques would therefore increase the capacity available for intrinsic load. The implementation focused on improving the common Control Flow Graph (CFG) visualization with the concept of the *Proximity View*: Simplify the view by removing most instructions, variables, and constants; keep only variables and constants that are arguments for function call nodes; and insert empty nodes to maintain the graph structure. A user study of 41 participants comparing this view to a traditional view demonstrated that subjects in the Proximity View group had statistically significant better performance in challenges solved, but took longer to solve them.

4.4 Section Summary

In this section, prompted by RQ2, “What are effective techniques for understanding and improving cognition in the context of binary PC?”, we reviewed relevant cognitive theory and its applications to binary PC. Starting with the basic concept of cognition, we progressed through external cognition, embodied cognition, and the concept of affordances, which are especially relevant to understanding how immersive interaction can influence and improve accomplishing analytic tasks. We then reviewed Cognitive Load Theory and some of its prior applications in software engineering and program comprehension.

In reviewing the prior research, we identified the following primary concepts and techniques that stood out, some across multiple sources, as being especially relevant to binary PC:

- External Cognition
 - Use external knowledge representations to reduce memory load, e.g., notes and reminders [71] [67]
 - Use computational tools (e.g., calculators) to make tasks easier [71] [67]
 - Annotate and reorder or restructure external representations of knowledge [71] [67]
- Embodied Cognition and Memory
 - Cognitive processing is influenced by the body and sensorimotor interactions [38] [47] [72]
 - Tools extend the body schema [49]
 - Physical objects or locations serve as memory palaces [4]
 - Whole-body stimuli can expedite storage and retrieval of memory [4]
- Cognitive Load Theory
 - Balance immediate problem-solving (intrinsic load) and long-term schema development (germane load) [79]
 - Worked example effect of learning from studying solved sample problems [45]
 - Split-attention effect of presenting information from multiple visual sources in an integrated way to reduce load required to perform mental integration [45]

- Modality effect of presenting multiple information sources through different modalities (e.g., visual and aural) to allow the inputs in each modality to be processed simultaneously [45]
- Remove redundancy of information presented in different modalities/sources to reduce the load of reconciling the underlying concepts across the inputs from those modalities [45]
- Both underloading and overloading can degrade performance; underloading is unlikely due to the high intrinsic load of PC [63] [42]
- Reduce extraneous load: Use human-centered design to reduce the wasted time and friction of poor tools and interactions used to solve a problem [42]
- PC activates areas of the brain associated with working memory, written language, and integration [74]

Now that we have reviewed elements of cognitive theory that underlie the cognitive processes of binary PC, we will examine how immersive technologies can improve those cognitive processes.

5 COGNITIVE AUGMENTATION USING VISUALIZATION AND IMMERSION

The third of three principles of CSE [46] directs that we should pursue practical solutions to the problem areas that we identified and further developed in Sections 2 through 4 by following the first and second principles of CSE. In pursuit of those solutions, we present in this section prior work in applying immersive technologies to improve cognition and performance in analytic tasks in general, with a few examples specifically for program comprehension and understanding.

5.1 Leveraging Affordances of Immersive Technologies for Analytic Tasks

As described in Section 4, affordances are perceivable aspects of the environment that foster interaction and enable external and embodied cognition. Some common affordances of virtual, augmented, and mixed reality (VR, AR, MR) include immersive visual and aural displays, spatial interaction / motion tracking, gesture and voice recognition, haptic feedback, plus additional emerging modalities. In this subsection, we look at prior work in using the affordances of immersive technologies to improve cognitive processes used by analytic and sensemaking tasks.

One key feature of an immersive environment is an abundance of space. Although the work of Andrews et al. [6] focused on the use of large two-dimensional displays, they proposed several benefits that working in a large spatial environment brings to the sensemaking process. They completed an observational study of participants performing well-known data analytic tasks using their desktop computing environment, which included a 32-megapixel display of approximately 100 inches by 31 inches. Generally, participants used the space to arrange and organize documents and applications in ways that reflected their relationships. The study showed evidence of a number of avenues to exploit for improving cognition by using a large spatial environment including persistence, context, physical navigation, presence of detail, memory refresh, situational awareness, and spatial semantics. These concepts are listed in full in the section summary, citing the authors, and will be further explored in Section 6. The findings by Andrews et al. endorse concepts we covered in Section 4 regarding external cognition and memory (augmenting working memory and structuring external knowledge representations) and embodied cognition (exploiting and extending the body schema).

In their hybrid survey/position paper, Moloney et al. [58] consider the question of whether VR technology provides affordances for analyzing big data. Their survey covers 47 selected publications in the field of visual data mining, which employs visual cognition to augment algorithmic analysis by differentiating variables by mapping them to distinct graphic attributes to harness human visual perception and creativity. Starting with the Computer Aided

Virtual Environment (CAVE) in 1992, they provide a timeline of advances through ‘VR 2.0’ and IA, and present future research challenges proposed by research teams. The authors perform an analysis of affordance theory in immersive VR as it exists in the literature and present a position on the shift from allocentric (attention focused externally) visual analytics—traditional visual analytics [23], which is performed using a 2D screen, even for 3D visualizations—to egocentric (attention focused internally) spatial coding and the affordance of VR. The key principles they identified include tuning the environment to human perception; using mimetic/naturalistic references; coordinating multiple modes of interaction; and aligning data selection with naturally-occurring distribution patterns. These principles are listed in full in the section summary, citing the authors, and will be further explored in Section 6.

Gračanin [39] makes the case that VR and MR technology can provide significant advantages in forming insights about complex data sets by leveraging the theory of affordances to increase embodied resources. The author proposes a framework for creating stimuli in an MR environment based on data from sensors and the user’s interactions with the system, which can reveal the most effective immersive and embodied stimuli through an iterative feedback process. Gračanin concludes that the immersion provided by VR may not be sufficient to fully leverage embodied cognition, and that the physical/real-world affordances provided by MR resolve that gap. Billinghurst et al. [9] similarly have developed a VR system incorporating a feedback loop using sensor data to adapt the interaction. Their work has demonstrated that electroencephalogram (EEG) measurements can assess cognitive load in VR training tasks, and the simulations can be adapted based on those measures to benefit training transfer effects. Ahmadi et al. [2], in related research, further refined the findings to pinpoint that power spectral density measurements of the EEG alpha band may reliably indicate cognitive load in moderate VR tasks.

Batch et al. [8] conducted an experiment to determine the effects of using IA for tasks in economic analysis. They used an iterative, human-centered approach with subject matter experts to extend the ImAxes tool [18], which provides embodied data axes in a VR environment. They employed a VR headset instead of AR due to better resolution and field of view. Some of their findings were surprises—e.g., fatigue was not a significant factor; no clear concerns about legibility; and participants with little gaming/VR experience used the tool effectively. Participants created egocentric presentation layouts, but did not appear to use the physical space to group similar views near each other, simply using the closest free space. Although this observation contradicted their hypothesis, the authors assert that what they did observe is consistent with the ‘sensemaking loop’ described by Pirolli et al. [65]: in the early bottom-up search for information when executing a task, retained information is left somewhat unstructured except to store it in a “shoebox” for later processing. These findings of Batch et al. corroborate those of Andrews et al. [6] when employing large two-dimensional displays for sensemaking tasks. Ultimately, Batch et al. [8] concluded that participants still did not fully use the three-dimensional space available in the environment beyond their immediate proximity, and speculate causes may include the small size of the physical space in which the study was conducted; ability to perform the techniques or gestures required to move the visualizations; and lack of automatic layout, indicating a need for constraints and organization frameworks.

Prather et al. [66] surveyed 104 papers on cognitive augmentation using immersive technologies, specifically work employing biosensor-based measures of user cognitive capabilities in immersive and semi-immersive environments. The authors performed this survey to support their aim of designing and developing cross-reality (XR) systems where task parameters are adapted to optimize the user’s cognitive load, particular in ‘Industry 4.0’ use cases where machines carry out repetitive and increasingly complex tasks. These systems would perform as intelligent cognitive assistants to enhance human capabilities. Their survey found that existing research is predominantly in the health and rehabilitation domain rather than industrial engineering. Many efforts target affective/emotional wellness with an emphasis on

short-term therapeutic tools rather than an enduring assistive system. Over one third of papers described work in adapting the user experience based on physiological data (predominantly electroencephalography (EEG)), with a small but significant number of them applying Artificial Intelligence (AI) and Machine Learning (ML) to that data. The authors included no papers that directly addressed CLT, but several addressed mitigation of mental exertion.

Some recent experimental findings in research on immersive technology and cognitive load pinpoint where the technology is most and least effective. Frederiksen et al. [36] performed a study of cognitive load in surgical training using immersive and non-immersive VR. The immersive VR method increased cognitive load significantly more than non-immersive VR in both nonstressor and stressor phases, which was attributed to extraneous cognitive load due to the high level of interaction with immersive VR elements. De Melo [25] et al. sought to reduce cognitive load with a virtual embodied AI-based assistant that used a human-appearing avatar in the virtual world to provide guidance to participants in completing tasks. In a controlled experiment comparing the embodied assistant to voice-only assistant and no assistant, embodied assistants led to lower cognitive load than voice-only assistants and both were lower than no assistant. Albus et al. [3] studied the effects of the signaling principle in VR learning environments. Per Mayer [55], using signals to direct a user to pay attention to specific information can create deeper understanding; this is the signaling principle. Compare this concept to beacons described earlier. In their study of a learning exercise with and without annotations (signals), Albus et al. found that the signals increased germane cognitive load, but did not reduce extraneous cognitive load, and did not result in significantly better deep understanding. Chen et al. [16] performed a controlled experiment measuring engineering creativity in a secondary education setting, with and without VR. They found that while VR improved cognition, motivation, and the novelty and usefulness of the designs, it did not improve creative thinking. Additionally, VR improved extraneous and germane cognitive load, but had no effect on intrinsic load.

The field of IA emerged only recently [15] [31] with an aim to immerse users in their data to improve sensemaking and formation of insights for real world data analytic tasks. Most recently, Ens et al. [33] have proposed ‘Immersive Analytics 2.0.’ They recognize that evidence found in prior work in IA points to benefits resulting from several areas: “use of unlimited space around the users, spatial memory, proprioception [awareness of body motion], 3D spatial interaction, tangible interaction [props and haptics], collaboration, kinesthesia [body motion], [and] engagement.” They call for work to more deeply understand the motivations for spatial and embodied interaction, and to clearly compare immersive versus non-immersive experiences in analytics.

5.2 Visualization Applied to Program Comprehension

Employing visualization can impact the effectiveness of analysts in binary PC and how they exercise their cognitive models. This section covers prior work in visualization to support PC, including that for binary PC and applicable work in source-level PC.

The bulk of prior work in this application area is aimed at the 2D desktop interaction metaphor, and we provide a few notable examples. Early work by Waguespack [85] implemented visualizations to aid novice Pascal programmers, for example, representing data types as different shapes, and using a variety of visual representations to differentiate type declarations, variable instances, and literal values. Structures are represented as containers of constituent components. This work demonstrated the utility of *chunking*: collecting lower-level details in a single higher-level abstraction that can help understanding of a large problem, then providing the ability to decompose as necessary for detailed analysis. Conti et al. [17] implemented a 2D application to visualize binary and data files as bitmaps; e.g., representing one byte per pixel with shading based on the byte’s value or presence in an address range. Rendering a binary file in that way, compared to viewing it in a hex editor, gives a view of the entire file at a glance at the expense of low-level details.

The method complements the text-based viewer to quickly identify major segments in the file, recurring or unusual patterns, and so on. Gregio et al. [40] demonstrated two 2D/pseudo-3D visualization methods of behavior of suspected malware: timeline plots and icons arranged in a spiral. Both methods display operating system actions taken by the code on objects (e.g., read file; terminate process; etc.) over a period of time in a single, compact view that complements traditional logs. The visualizations were particularly useful in identifying where two sample programs acted similarly, indicating shared code across different programs, or different revisions of the same program.

Considering a narrower application area, Wagner et al. [84] surveyed a pool of 220 papers related to code visualization and identified 25 papers specifically about malware visualization systems. From those papers, they identified nine *data providers* behind those visualization systems. The data providers collect information about the suspected malware and provide it to the visualization systems—automated and manual applications that execute static and dynamic analysis techniques to collect data useful in profiling and classifying the malware. The visualization systems were binned into three broad groups: individual malware analysis, malware comparison, and malware summarization. Additionally, they were categorized based on well-established taxonomies from the visualization community: the type of provided data, visualization techniques used, mapping and representation space, temporal aspects, interactivity, and goal/action. They identified challenges in bridging between the three broad groups, integrating disparate data sources, characterizing and abstracting problems, improving expert interaction, and integrating analytical methods with the visualizations. These systems are still aimed at the 2D desktop interaction metaphor—while some of the systems analyzed employed 3D visualizations, none were reported in this survey to employ virtual reality or immersive (embodied/egocentric) techniques.

Elliott et al. [32] explored the use of VR in software engineering to address problems in navigating and comprehending code. Their work builds upon prior research in how developers use the affordance of *spatial memory* in traditional 2D development environments, such as using scrollbar and tab positions as cues [51], or using an “infinitely”-scrollable document canvas [26] [10]. This work extends that concept to the affordances of VR applied to software development: spatial cognition, cues, and presence; manipulation and motion to improve perception and retention; and immediate feedback on the state of the system. With these affordances implemented in their RIFTSKETCH (live development) and IMMERSION (code review) tools, the authors provide a proof-of-concept and a vision of VR-based development in the future, though no formal user studies were conducted.

In a user study conducted by Dominic et al. [28], 26 graduate students were tested on comprehending simple Java programs of the type that one may find in first-year programming course homework assignments. They compared the traditional desktop experience with a VR configuration entitled “VirtualDesk” using a headset and tracked keyboard and mouse that were mapped 1:1 to the real world. The study did not implement specific affordances of VR as proposed by Elliott et al. [32], but instead compared the performance of PC using common 2D tools on a traditional desktop environment against using the same tools in VR. Their results show 75% of programs were comprehended correctly in the traditional desktop experience compared to 65% in VR. Additionally, their results from conducting the NASA Task Load Index (TLX) survey [41] showed significantly more task load—the demand or difficulty in performing a task—in VR. Finally, results of a survey of self-reported concentration and productivity showed that users in VR had lower levels of concentration and no significant difference in perceived productivity.

One rich facet of human-computer interface theory is the use of metaphors to influence the design of affordances. Lakoff [52] curated an extensive list of metaphors encountered in linguistics. Metaphors leverage common experiences amongst most users to facilitate understanding of new concepts. Many of these metaphors can apply to perception and cognition in interactive applications, e.g., “seeing is touching,” “the visual field is a container,” “theories are

constructed objects,” etc. Averbukh et al. [7] surveyed applications of VR for (high level) program visualization and visual programming, and in particular, the metaphors employed in those applications. They reviewed the city, molecule, and heliocentric cosmic metaphors, asserting they share important qualities: “unlimited context, organization of inner structure, naturalness, and resistance to scaling,” and that these natural metaphors simplify spatial orientation and navigation in the VR world.

The city metaphor recurs in many VR-based visualization efforts. One early instance was by Fittkau [35] et al., who implemented a VR experience to aid the PC process that uses the metaphor of a program as a city block: the buildings are classes and packages, and the execution trace is represented as straight-line “footpaths” between the buildings. Participants experienced the tool ExplorViz via immersive VR, using gestures to translate, rotate, zoom, and select. The experience was intended to provide analysts a novel tool while employing familiar metaphors. These participants rated the experience of answering basic comprehension questions with this tool as suitable for performing PC, and as an alternative, albeit needing adaptations, to a classic experience.

In another exploration using the city metaphor, Oberhauser and Lecon [62] employed immersive VR to aid PC by providing participants the ability to fly through a 3D representation of code in their tool Gamified Virtual Reality FlyThruCode (GVR-FTC). Two metaphors, “universe” and “terrestrial,” related the code components to familiar concepts, where packages/classes/dependencies were represented as solar systems/planets/light beams and cities/buildings/pipes respectively, and the scale of objects represented various metrics such as number of class methods. The team evaluated the tool using two games that motivated players to comprehend the dependency structure and modularization of a code project compared to a common text editor. Although the subject pool was too small to demonstrate statistical significance, the results did show higher comprehension using VR.

In a larger-scale experiment, Romano et al. [70] compared the relative effectiveness of three tools on the task of source code PC. The baseline was a traditional integrated development environment (IDE) with extensions for code metrics and smells, which was compared to a city-metaphor-based virtual reality environment in both immersive (Code2City_{VR}) and non-immersive (Code2City) forms. This implementation of the city metaphor [14] creates a building (parallelepiped) for each class, where class properties are reflected in the size and color of the building. In their study [70], 42 participants solved PC tasks based on two large open-source Java projects. Both VR-based tools resulted in significantly better correctness in the completion of PC tasks than the IDE. Additionally, the time to complete the tasks was significantly shortened in the immersive environment than both the non-immersive VR and the IDE.

More recently, Hoff et al. [44] performed a similar experiment for source code PC comparing their approach, Immersive Software Archaeology, with another VR method and an IDE. Their approach is focused on providing an overview, with multiple levels of abstraction, of a software system’s architecture. The higher levels of abstraction (architectural) were represented by solar system/planets/continents, and the lower levels (design) were represented by cities/building/floors. Their study of 54 participants demonstrated that their solution provided similar or better performance in tasks exercising accessing information and finding horizontal and vertical relationships in the system’s architecture.

Finally, we mention a variation of the city metaphor. Weninger et al. [87] introduced the concept of *Memory Cities* to visualize how an application uses heap memory over time, rather than using it to visualize code. Objects on the heap are grouped and represented as buildings in a 3D visualization. Attributes such as color, opacity, area, and height represent various metrics of the heap, and the buildings evolve as the program executes. The authors describe how their tool helps users identify memory leaks in two use cases and plan user studies in the future.

As mentioned at the start of the section, we are careful to distinguish research contributions that focus on source-level PC from those that focus on binary PC. We include them because, even though we will not have higher-level structures, abstractions, and internal documentation for a binary program, and even though any higher-level abstractions generated by tools will have significant uncertainty (see Section 2), we believe many of the findings presented in this subsection can inform immersive VR for binary PC.

5.3 Section Summary

In approaching RQ3, “How can the affordances of immersive interaction with virtual reality be employed to improve cognition in analytic tasks,” we reviewed relevant research contributions in the fields of immersive technologies and IA in particular, as well as visualizations and metaphors employed in PC. Through that process we identified many affordances, techniques, and concepts as follows:

- Signalling [3] [55]
- Incorporate common reverse engineering tools
- User-organization of visualizations in 3D space [8]
- Spatial semantics: spatial organization provides added semantic layer [6]
- Incremental formalism: structure is emergent with understanding [6]
- Physical navigation: enables efficient access to information through quick body movements [6]
- That immersion must be accompanied by embodiment [39]
- Embodied assistant [25]
- Context: physical location helps to restore state-of-mind [6]
- Persistence: exploits spatial (position and representation) memory to remember information [6]
- Use abstractions of environments to which human perception is attuned [58]
- Use representations tuned to intermediate zone where human perception is most discerning [58]
- Refresh: serendipitous glances refresh memory of information [6]
- Awareness: scanning the space quickly assesses overall status [6]
- Use a relevant metaphor to visualize programs (e.g., city block) [35] [62] [14] [7] [70] [44]
- Gamified VR for PC [62]
- EEG-based adjustment of cognitive load [9]
- Mimetic references overlaid with constructed affordances [58]
- Presence of detail: detailed information enables rapid access and synthesis based on rich content [6]
- Cross-modal mapping [58]
- Visualize binary and data files as bitmaps, complementing a text viewer [17]
- Timeline plots and icons arranged in a spiral, complementing log files [40]
- Application of Chunking: collecting lower-level details in a single higher-level abstraction that can help understanding of a large problem [85]

Next, we will analyze our findings regarding VR affordances in the context of our prior findings in cognitive models of binary PC and the cognitive theory behind those models.

6 ANALYSIS AND NEXT STEPS

In the framework of CSE, Section 2 characterizes the problems of binary PC, setting a context for this work; Sections 3 and 4 fill the roles of a functional decomposition and work domain analysis; and Section 5 describes immersive affordances that may improve human performance in certain applications. Our fourth research question asks, “How can we use these findings to effect improvements in the practice of binary PC?” To answer this question, we examine the connections between the findings in the previous sections and identify the most salient themes—performing an indirect cognitive task analysis—in order to propose directions in the information and interaction design for this problem domain.

6.1 Overview of Elements and Connecting Threads

Consider the findings of this survey partitioned into three groups matching the three previous sections: (A) cognitive models of binary PC, (B) concepts of cognitive theory, and (C) VR affordances/tools/techniques. List 1 provides the full text of each element with attribution, by group. Conceptual *threads* tie an individual cognitive model element from group A to an element of cognitive theory in group B that supports it, and then to a VR technique in group C that, when implemented, could improve that element of the cognitive model in an immersive binary PC tool. Figure 5 provides an overview of these threads tying together the elements from each group (element labels are abbreviated). The authors determined these threads subjectively, linking elements between groups that the authors believe to be sufficiently directly related. We acknowledge that the threshold of “directness” is arbitrary and tuned to produce tight groups, and that strong arguments can be made to link the elements in other ways. We present the entire body of threads in the figure to illustrate just one take on the many logical avenues of further research this problem domain provides before we further narrow the set. Next, we will narrow our focus.

6.2 Primary Themes

We aligned related threads into overarching *themes* that are critical to the process of binary PC. In this alignment process, we concentrated on that we expect will have a significant impact on the task of *binary* PC; due to the lack of related research specifically in embodied immersion for binary PC, many of our sources focus on source code PC or software engineering in general, and while we believe many of those concepts may apply in the binary domain, here we’ve exercised subjective judgement to narrow the focus to those elements that most strongly apply. Clustering these threads revealed the themes we present here:

- Enhancing abductive iteration (hypothesis loop)
- Augmenting working memory
- Supporting information organization and discovery of important features

Figure 6 depicts the three themes and the most closely related elements from groups A, B, and C. These three themes touch upon every element from group A, cognitive modelling of binary PC, which is important because we want to find the most effective ways to augment as many of the cognitive model elements as possible. Moving into the group B elements of cognitive theory, we are more selective in our areas of concentration. Finally, we derive a central set of VR affordances from group C on which we will focus our efforts in future work. We will examine each theme more closely in the remainder of this section.

<u>Cognitive models of binary PC</u>	<u>Concepts of cognitive theory</u>	<u>VR affordances/tools/techniques</u>
<p>A1. Uses multi-level internal semantic representation of the program [73]</p> <p>A2. Abductive iteration: Sets goals and follows plans [13] [61]</p> <p>A3. Abductive iteration: Forms increasingly complete hypotheses [12] [86] [29] [83]</p> <p>A4. Abductive iteration: Tests hypothesis through experimentation [13] [75] [29] [83]</p> <p>A5. Abductive iteration: Updates framing of the problem based on results [50] [13] [29] [83]</p> <p>A6. Creates disorientation following recursions and execution paths [89]</p> <p>A7. Taxes working memory [73] [89]</p> <p>A8. Requires declarative and procedural knowledge retrieval and generation [13]</p> <p>A9. Uses translation—determining how the code would be implemented in a higher-level language [75]</p> <p>A10. Uses beacons; beacons for binary PC are more diverse than for source-code-based PC [12] [29] [83]</p> <p>A11. Relies upon overview of the binary executable [83]</p> <p>A12. Uses external memory aids [27] [77]</p> <p>A13. Relies upon determining what to ignore [54]</p> <p>A14. Experts use a breadth-first approach with system thinking; novices, both depth- and breadth-first without system thinking [82]</p> <p>A15. Experts use a top-down approach if the program domain is familiar, otherwise, bottom-up [74]</p>	<p>B1. External Cognition: Use external knowledge representations to reduce memory load, e.g., notes and reminders [71] [67]</p> <p>B2. External Cognition: Use computational tools (e.g., calculators) to make tasks easier [71] [67]</p> <p>B3. External Cognition: Annotate and reorder or restructure external representations of knowledge [71] [67]</p> <p>B4. Embodied Cognition: Cognitive processing is influenced by the body and sensorimotor interactions [38] [47] [72]</p> <p>B5. Embodied Cognition: Tools extend the body schema [49]</p> <p>B6. Embodied Memory: Physical objects or locations serve as memory palaces [4]</p> <p>B7. Embodied Memory: Whole-body stimuli can expedite storage and retrieval of memory [4]</p> <p>B8. CLT: Balance immediate problem-solving (intrinsic load) and long-term schema development (germane load) [79]</p> <p>B9. CLT: Worked example effect of learning from studying solved sample problems [45]</p> <p>B10. CLT: Split-attention effect of presenting information from multiple sources in an integrated way to reduce load from mental integration [45]</p> <p>B11. CLT: Modality effect of presenting multiple information sources through different modalities (primarily visual and aural) to reduce the integration load [45]</p> <p>B12. CLT: Remove redundancy of information presented in different modalities/sources to reduce the load of reconciling the underlying concepts [45]</p> <p>B13. CLT: Both underloading and overloading can degrade performance; underloading is unlikely due to the high intrinsic load of PC [63] [42]</p> <p>B14. CLT: Reduce extraneous load: Use human-centered design to reduce the wasted time and friction of poor tools and interactions used to solve a problem [42]</p> <p>B15. PC activates areas of the brain associated with working memory, written language, and integration [74]</p>	<p>C1. Signalling [3] [55]</p> <p>C2. Common Rev Eng tools</p> <p>C3. User-organization of visualizations in 3D space [8]</p> <p>C4. Spatial semantics: spatial organization provides added semantic layer [6]</p> <p>C5. Incremental formalism: structure is emergent with understanding [6]</p> <p>C6. Physical navigation: enables efficient access to information through quick body movements [6]</p> <p>C7. That immersion must be accompanied by embodiment [39]</p> <p>C8. Embodied assistant [25]</p> <p>C9. Context: physical location helps to restore state-of-mind [6]</p> <p>C10. Persistence: exploits spatial (position and representation) memory to remember information [6]</p> <p>C11. Use abstractions of environments to which human perception is attuned [58]</p> <p>C12. Use representations tuned to intermediate zone where human perception is most discerning [58]</p> <p>C13. Refresh: serendipitous glances refresh memory of information [6]</p> <p>C14. Awareness: scanning the space quickly assess overall status [6]</p> <p>C15. Use a relevant metaphor to visualize programs (e.g., city block) [35] [62] [14] [7] [70] [44]</p> <p>C16. Gamified VR for PC [62]</p> <p>C17. EEG-based adjustment of cognitive load [9]</p> <p>C18. Mimetic references overlaid with constructed affordances [58]</p> <p>C19. Presence of detail: detailed information enables rapid access and synthesis based on rich content [6]</p> <p>C20. Cross-modal mapping [58]</p> <p>C21. Visualize binary and data files as bitmaps, complementing a text viewer [17]</p> <p>C22. Timeline plots and icons arranged in a spiral, complementing log files [40]</p> <p>C23. Application of Chunking: collecting lower-level details in a single higher-level abstraction that can help understanding of a large problem [85]</p>

Table 1. Elements of (A) cognitive models of binary PC, (B) concepts of cognitive theory, and (C) VR affordances/tools/techniques

6.2.1 *Enhancing abductive iteration (hypothesis loop)*. One very prevalent theme in cognitive models of PC is the iterative pattern of sensemaking using abductive reasoning. In this analysis, that pattern is broken into four elements of abductive iteration: *setting goals/following plans*, *forming hypotheses*, *experimenting to test hypotheses*, and *updating what is known* [13] [61] [12] [86] [29] [83] [75] [50]. The cognitive model elements of *multi-level internal semantic representation* [73] and *heavy dependence on working memory* [73] [89] also broadly apply to this theme.

The elements of cognitive theory most closely related to this theme include managing cognitive load in the iterative sensemaking loop through *balancing the intrinsic and germane loads* [79] and by *reducing extraneous load as much as possible through human-centered design* [42]. As this theme encompasses the overarching execution of the PC task,

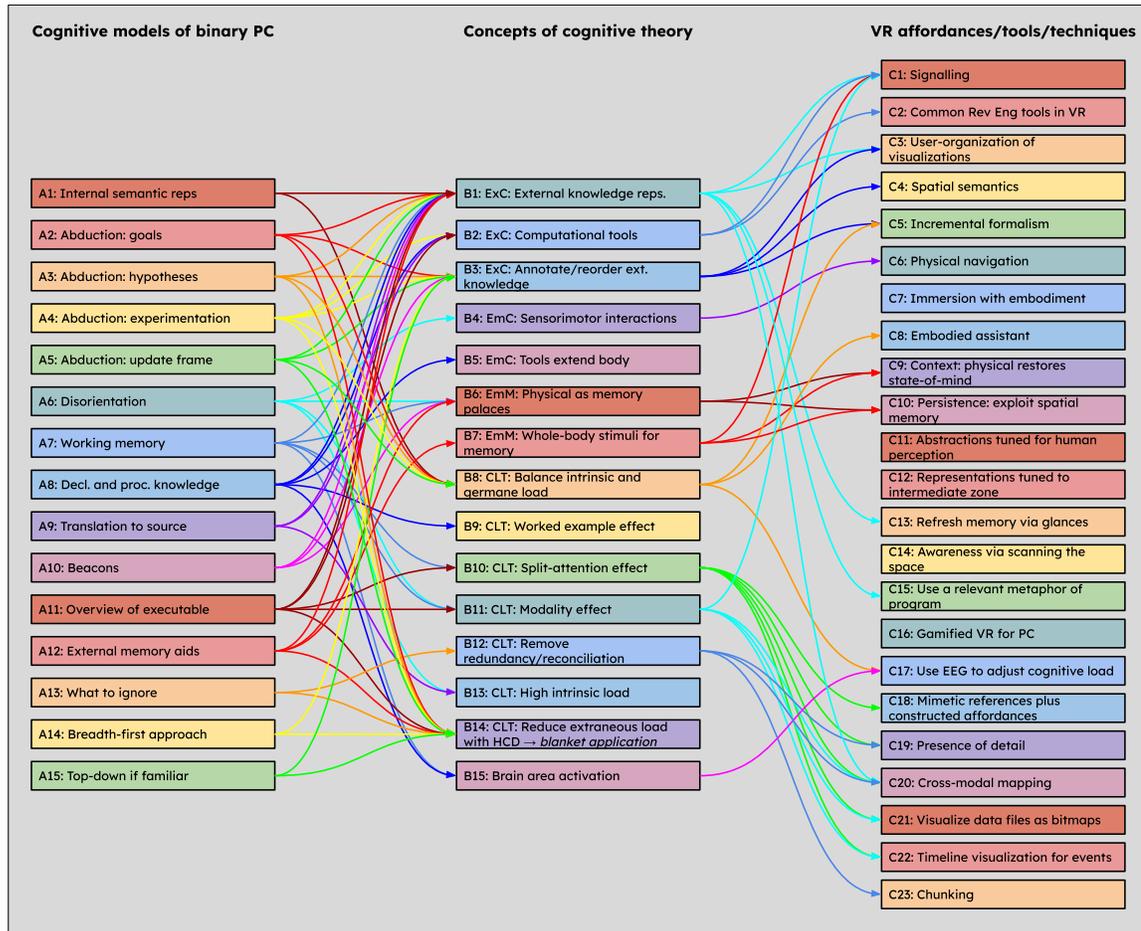


Fig. 5. Threads connecting elements of (A) cognitive models of binary PC, (B) concepts of cognitive theory, and (C) VR affordances/tools/techniques

it has overlap with the other themes. For example, relevant elements from cognitive theory include those related to information organization and memory. In this theme, we narrow the focus specifically to facilitating the iterative loop and reserve examination of specific elements related to information and memory for later in this section.

In the iterative process, the prime opportunity for augmentation is in amplifying the expert's ability to identify and track progress toward goals and hypotheses. *What is the current goal? What is the current hypothesis? What evidence has been built? What are prior decision points to which we can return upon hitting a dead end?*

The immersive affordances most applicable include *incremental formalism* [6], in which the visualization of the iterative process evolves its structure as progress is made; *spatial semantics* [6], in which the spatial organization of the information provides semantic information; and the *embodied assistant* [25] that would (ideally) track the expert's decision path and provide the interactive interface by which the expert can ask questions, make notes, etc.

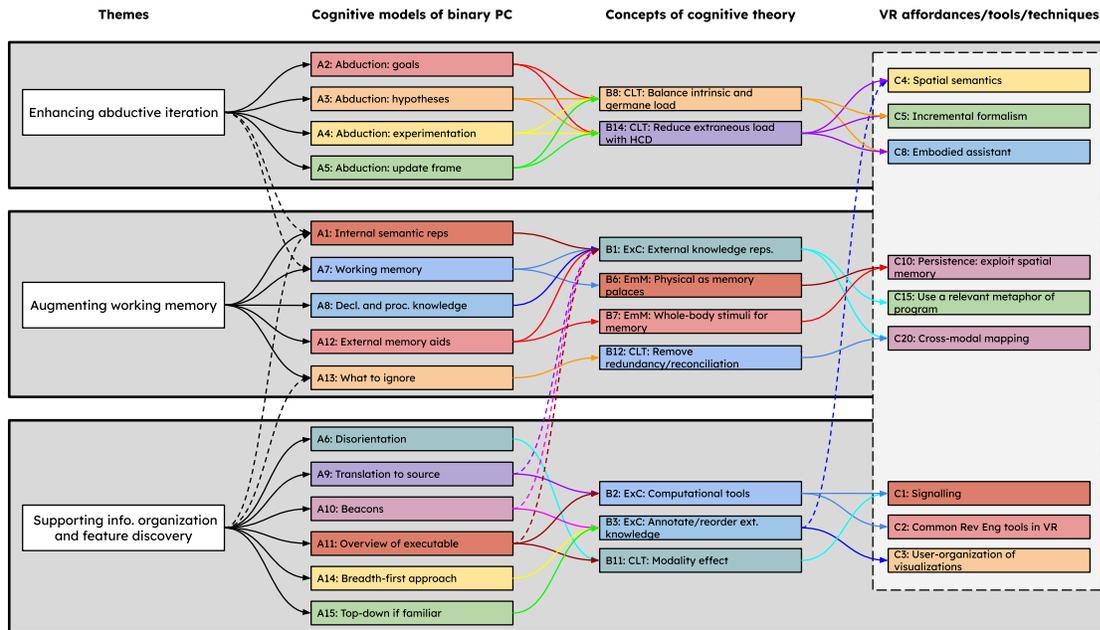


Fig. 6. Primary themes for analysis with most closely-related elements; highlighted area indicates highest-priority VR affordances

6.2.2 Augmenting working memory. The limitations of working memory impact the effectiveness and efficiency with which analytical problems are solved, binary PC or otherwise. Closely-related elements from cognitive models of PC include *multi-level internal semantic representation* [73], *taxing working memory* [73] [89], *generation, storage, and retrieval of declarative and procedural knowledge* [13], *using external memory aids* [27] [77], and *determining what to ignore* [54].

The most closely-related elements of cognitive theory include *using external knowledge representations to reduce memory load* [71] [67] by offloading some knowledge that otherwise would be held in working memory, *employing memory palaces aided by physical objects or locations and leveraging whole-body stimuli to expedite storage and retrieval of memory* [4], and *removing redundancy of information presented in different modalities* [45] to streamline the intake of new information into working memory.

Following those threads from the cognitive model to cognitive theory leads us to a few considerations for immersive VR: *employing persistence to exploit spatial memory* [6] to remember information, *using physical metaphors* [35] [62] [14] [7] [70] [44] for the task's information and operations, and *cross-modal mapping* [58] to present information using multiple senses. Additionally, reducing extraneous cognitive load, as described in the previous theme, can ensure working memory capacity is freed for solving the intrinsic problem.

6.2.3 Supporting information organization and feature discovery. Of the three themes, *supporting information organization and feature discovery* is the most context-dependent; while the themes of iterative abductive process and enhancements to working memory can apply to almost any analytic problem, this theme is most tightly-integrated with the problem domain and its existing methods and tools.

The elements from cognitive models of binary PC most closely tied to this problem domain include *disorientation following execution paths* [89], *translating the binary back to source code* [75], *using or marking beacons* [12] [29] [83], *referring to an overview of the binary program* [83], and *approaching the task in a breadth-first and top-down manner, depending on the expert's familiarity* [82] [74]. Additional related elements, crossovers with the other two themes, include *multi-level internal semantic representations* [73] of the composition of the binary program and *determining what to ignore* [54] from the masses of data in the program and system.

Several cognitive theory elements support this theme. *Annotating and reordering/restructuring external representations of knowledge* [71] [67] addresses how experts take notes and organize process artifacts. *Using computational tools to make tasks easier* [71] [67] covers the use of current well-known reverse engineering tools to discover key characteristics of the binary in an automated or semi-automated fashion. The *modality effect* [45] can also be leveraged given multiple sources of information and interpretations of the same binary program, for example, to provide signalling cues. Finally, as a crossover element with the working memory theme, *using external knowledge representations to reduce memory load* [71] [67] acknowledges the need to use the external knowledge representations mentioned at the start of this paragraph.

Several VR affordances apply to this theme. First, *incorporate common reverse engineering tools* in a way that allows the expert to naturally interact with them from within the immersive environment; this task may require a novel interaction paradigm. Second, *signalling* [3] [55] will help the expert focus on what is most important; what is signalled may come from self-sourced notations, or it may come from cues inferred from the reverse engineering tools. *Spatial semantics* [6] applies here just as in the iterative abduction theme: the spatial organization of the data provides meaning itself, over and above the meaning of the data. Finally, *user organization of visualizations* [8] applies, especially in combination with the preceding element, where that organization in space is itself meaningful.

6.3 Section Summary

Three primary themes capture the salient aspects of cognitive models of binary PC. Following the threads that connect cognitive models to cognitive theory and then to VR affordances, we derived a set of the most relevant qualities for an immersive experience for binary PC:

- *Incremental formalism* [6], in which the visualization of the iterative process evolves its structure as progress is made
- *Spatial semantics* [6], in which the spatial organization of the information provides semantic information
- *Embodied assistant* [25] that (ideally) tracks the expert's decision path and provides the interactive interface by which the expert can ask questions, make notes, etc.
- *Employing persistence to exploit spatial memory* [6] to remember information
- *Meaningful and relatable metaphors* [35] [62] [14] [7] [70] [44] for the task's information and operations
- *Cross-modal mapping* [58] to present information using multiple senses
- *Incorporate common reverse engineering tools* to leverage powerful tools already familiar to experts
- *Signalling* [3] [55] focuses the expert's attention on potential key discoveries
- *User organization of visualizations* [8], in conjunction with spatial semantics, allows the expert to capture knowledge in the positioning of the data itself

In addition to these specific elements, we must consider overarching considerations for an immersive VR implementation. Any successful implementation will follow an *iterative human-centered design and development process*. It

will incorporate automation as much as possible, including taking advantage of large language models. Additionally, it will be *adaptive* and self-tune the parameters of the immersive experience to give the expert user the most effective experience.

The actual visualizations, environments, interactions, and metaphors to be used in an immersive experience still require investigation; these qualities should serve as guidelines heavily influencing the selection, design, and development of those experiences. Some features to consider implementing following the above guidelines include:

- **Intuitive representation of the code**, highlighting features like function calls and branching/switch statements
- **Basic discovery of the binary**: the compiler that built the binary; clusters of system calls; presence of encryption functions; places where a code block is referenced; etc.
- **Identifying, marking, and tracking beacons**; point, click, annotate to create
- A simplified environment, **masking out what can be ignored**; allowing the user to “collapse” and “expand” portions of the data
- Providing an expedited way to get an **overview of the complete binary program** and then drilling down
- Expediting the **creation, storage, and retrieval of external knowledge** through a natural interface

7 CONCLUSION

The process of binary PC is difficult and requires very specialized expertise to perform it effectively. The problem is only getting harder as computer architectures become more varied and complex and binary obfuscation techniques become more sophisticated. Augmenting the cognitive process of binary PC is crucial to maintaining or improving the current level of effectiveness of experts performing this task.

Following a CSE approach, we surveyed prior work in three progressive groups to support an indirect cognitive task analysis: characteristics of mental or cognitive models of binary PC, cognitive theory and applications to binary PC, and cognitive augmentation using visualization and immersive technologies. We identified several common or important elements of each group and presented threads of related elements. Through an inductive process, we identified three primary themes that encapsulate the elements of cognitive models surveyed, related threads of cognitive theory, and immersive affordances that provide a path for implementing cognitive augmentation of the binary PC process. Finally, we derived a set of guidelines by which to design and develop immersive experiences for binary PC.

8 ACKNOWLEDGMENTS

The primary author is supported by the United States Air Force Science & Engineering Career Field Team Science, Technology, Engineering, Mathematics + Management (STEM+M) program.

9 AUTHOR DISCLAIMER

The opinions contained herein are those of the authors only and should not be construed as those of the Department of Defense or United States Government.

REFERENCES

- [1] US National Security Agency. 2023. Ghidra software reverse engineering suite of tools. <https://ghidra-sre.org/>
- [2] Mohammad Ahmadi, Huidong Bai, Alex Chatburn, Marzieh Ahmadi Najatabadi, Burkhard C. Wünsche, and Mark Billinghurst. 2023. Comparison of Physiological Cues for Cognitive Load Measures in VR. In *2023 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. 837–838. <https://doi.org/10.1109/VRW58643.2023.00261>

- [3] Patrick Albus, Andrea Vogt, and Tina Seufert. 2021. Signaling in virtual reality influences learning outcome and cognitive load. *Computers & Education* 166 (2021), 104154. <https://doi.org/10.1016/j.compedu.2021.104154>
- [4] Moyosore Ale, Miriam Sturdee, and Elisa Rubegni. 2022. A systematic survey on embodied cognition: 11 years of research in child-computer interaction. *International Journal of Child-Computer Interaction* 33 (2022), 100478. <https://doi.org/10.1016/j.ijcci.2022.100478>
- [5] J. R. Anderson and C. Lebiere. 1998. *The atomic components of thought*. Lawrence Erlbaum associates, Mahwah, New Jersey.
- [6] Christopher Andrews, Alex Endert, and Chris North. 2010. Space to Think: Large High-Resolution Displays for Sensemaking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, Georgia, USA) (*CHI '10*). Association for Computing Machinery, New York, NY, USA, 55–64. <https://doi.org/10.1145/1753326.1753336>
- [7] Vladimir Averbukh, Natalya Averbukh, Pavel Vasev, Ilya Gvozdev, Georgy Levchuk, Leonid Melkozerov, and Igor Mikhaylov. 2019. Metaphors for Software Visualization Systems Based on Virtual Reality. In *Augmented Reality, Virtual Reality, and Computer Graphics*, Lucio Tommaso De Paolis and Patrick Bourdot (Eds.). Springer International Publishing, Cham, 60–70.
- [8] Andrea Batch, Andrew Cunningham, Maxime Cordeil, Niklas Elmqvist, Tim Dwyer, Bruce H. Thomas, and Kim Marriott. 2020. There Is No Spoon: Evaluating Performance, Space Use, and Presence with Expert Domain Users in Immersive Analytics. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2020), 536–546. <https://doi.org/10.1109/TVCG.2019.2934803>
- [9] Mark Billinghurst, Bruce Thomas, Javaan Chahl, Ina Bornkessel-Schlesewsky, Maarten Immink, and Mathias Schlesewsky. 2019. *Enhancing Human Performance Using Virtual Reality, Wearable Computing, Cognitive Neuroscience and Mental Training*. Retrieved December 19, 2022 from <https://www.dst.defence.gov.au/sites/default/files/events/documents/9.0%20Improving-cognitive-performance-UniSA-Billinghurst-Chaal.pdf>
- [10] Andrew Bragdon, Steven P. Reiss, Robert Zeleznik, Suman Karumuri, William Cheung, Joshua Kaplan, Christopher Coleman, Ferdi Adeputra, and Joseph J. LaViola. 2010. Code bubbles: rethinking the user interface paradigm of integrated development environments. In *2010 ACM/IEEE 32nd International Conference on Software Engineering*, Vol. 1. 455–464. <https://doi.org/10.1145/1806799.1806866>
- [11] K. Brodman. 2006. *Brodman's Localisation in the Cerebral Cortex*. Springer, New York.
- [12] Ruven Brooks. 1983. Towards a theory of the comprehension of computer programs. *International Journal of Man-Machine Studies* 18, 6 (1983), 543–554. [https://doi.org/10.1016/S0020-7373\(83\)80031-5](https://doi.org/10.1016/S0020-7373(83)80031-5)
- [13] Adam Bryant, Robert Mills, Gilbert Peterson, and Michael Grimaila. 2012. Software reverse engineering as a sensemaking task. *Journal of Information Assurance and Security* 6 (01 2012), 483–494.
- [14] Nicola Capece, Ugo Erra, Simone Romano, and Giuseppe Scanniello. 2017. Visualising a Software System as a City Through Virtual Reality. In *Augmented Reality, Virtual Reality, and Computer Graphics*, Lucio Tommaso De Paolis, Patrick Bourdot, and Antonio Mongelli (Eds.). Springer International Publishing, Cham, 319–327.
- [15] Tom Chandler, Maxime Cordeil, Tobias Czauderna, Tim Dwyer, Jaroslaw Glowacki, Cagatay Goncu, Matthias Klapperstueck, Karsten Klein, Kim Marriott, Falk Schreiber, and Elliot Wilson. 2015. Immersive Analytics. In *2015 Big Data Visual Analytics (BDVA)*. 1–8. <https://doi.org/10.1109/BDVA.2015.7314296>
- [16] Yi-Ching Chen, Yu-Shan Chang, and Meng-Jung Chuang. 2022. Virtual reality application influences cognitive load-mediated creativity components and creative performance in engineering design. *Journal of Computer Assisted Learning* 38, 1 (2022), 6–18. <https://doi.org/10.1111/jcal.12588> arXiv:<https://onlinelibrary.wiley.com/doi/pdf/10.1111/jcal.12588>
- [17] Gregory Conti, Erik Dean, Matthew Sinda, and Benjamin Sangster. 2008. Visual Reverse Engineering of Binary and Data Files. In *Visualization for Computer Security*, John R. Goodall, Gregory Conti, and Kwan-Liu Ma (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–17.
- [18] Maxime Cordeil, Andrew Cunningham, Tim Dwyer, Bruce H. Thomas, and Kim Marriott. 2017. ImAxes: Immersive Axes as Embodied Affordances for Interactive Multivariate Data Visualisation. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) (*UIST '17*). Association for Computing Machinery, New York, NY, USA, 71–83. <https://doi.org/10.1145/3126594.3126613>
- [19] Jennifer Cowley. 2014. Job Analysis Results for Malicious-Code Reverse Engineers: A Case Study.
- [20] Kenneth John William Craik. 1943. *The Nature of Explanation*. Cambridge University Press, Cambridge.
- [21] Beth Crandall, Gary A Klein, and Robert R Hoffman. 2006. *Working minds: A practitioner's guide to cognitive task analysis*. Mit Press.
- [22] Douglas Crockford. 2008. *JavaScript: The Good Parts*. O'Reilly Media, Inc.
- [23] Wenqiang Cui. 2019. Visual Analytics: A Comprehensive Overview. *IEEE Access* 7 (2019), 81555–81573. <https://doi.org/10.1109/ACCESS.2019.2923736>
- [24] Yaniv David, Uri Alon, and Eran Yahav. 2020. Neural reverse engineering of stripped binaries using augmented control flow graphs. *Proceedings of the ACM on Programming Languages* 4, OOPSLA (nov 2020), 1–28. <https://doi.org/10.1145/3428293>
- [25] Celso M. de Melo, Kangsoo Kim, Nahal Norouzi, Gerd Bruder, and Gregory Welch. 2020. Reducing Cognitive Load and Improving Warfighter Problem Solving With Intelligent Virtual Assistants. *Frontiers in Psychology* 11 (2020). <https://doi.org/10.3389/fpsyg.2020.554706>
- [26] Robert DeLine and Kael Rowan. 2010. Code canvas: zooming towards better development environments. In *2010 ACM/IEEE 32nd International Conference on Software Engineering*, Vol. 2. 207–210. <https://doi.org/10.1145/1810295.1810331>
- [27] Françoise Détéienne and Frank Bott. 2001. *Software Design—Cognitive Aspects*. Springer-Verlag, Berlin, Heidelberg.
- [28] James Dominic, Brock Tubre, Jada Houser, Charles Ritter, Deborah Kunkel, and Paige Rodeghero. 2020. *Program Comprehension in Virtual Reality*. Association for Computing Machinery, New York, NY, USA, 391–395. <https://doi.org/10.1145/3387904.3389287>
- [29] Patrick P. Dudenhofer. 2019. Modeling and Automating the Cyber Reverse Engineering Cognitive Process. In *23rd Colloquium for Information Systems Security Education* (Las Vegas, NV, USA) (*CISSE '19*).
- [30] Patrick P. Dudenhofer and Adam R. Bryant. 2017. Establishing a Cognitive Understanding of Cyber Reverse Engineering Tasks.

- [31] Tim Dwyer, Kim Marriott, Tobias Isenberg, Karsten Klein, Nathalie Riche, Falk Schreiber, Wolfgang Stuerzlinger, and Bruce H. Thomas. 2018. *Immersive Analytics: An Introduction*. Springer International Publishing, Cham, 1–23. https://doi.org/10.1007/978-3-030-01388-2_1
- [32] Anthony Elliott, Brian Peiris, and Chris Parnin. 2015. Virtual Reality in Software Engineering: Affordances, Applications, and Challenges. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 2. 547–550. <https://doi.org/10.1109/ICSE.2015.191>
- [33] Barrett Ens, Maxime Cordeil, Chris North, Tim Dwyer, Lonni Besançon, Arnaud Prouzeau, Jiazhou Liu, Andrew Cunningham, Adam Drogemuller, Kadek Ananta Satriadi, and Bruce H Thomas. 2022. Immersive Analytics 2.0: Spatial and Embodied Sensemaking. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems (New Orleans, LA, USA) (CHI EA '22)*. Association for Computing Machinery, New York, NY, USA, Article 91, 7 pages. <https://doi.org/10.1145/3491101.3503726>
- [34] Michael W. Eysenck and Marc Brysbaert. 2018. *Fundamentals of Cognition 3rd Edition*. Routledge, London.
- [35] Florian Fittkau, Alexander Krause, and Wilhelm Hasselbring. 2015. Exploring software cities in virtual reality. *2015 IEEE 3rd Working Conference on Software Visualization (VISSOFT) (2015)*, 130–134.
- [36] Joakim Grant Frederiksen, Stine Maya Dreier Sørensen, Lars Konge, Morten Bo Søndergaard Svendsen, Morten Nobel-Jørgensen, Flemming Bjerrum, and Steven Arild Wuyts Andersen. 2020. Cognitive load and performance in immersive virtual reality versus conventional virtual reality simulation training of laparoscopic surgery: a randomized trial. *Surgical Endoscopy* 34, 3 (01 Mar 2020), 1244–1252. <https://doi.org/10.1007/s00464-019-06887-8>
- [37] James J. Gibson. 1977. The Theory of Affordances. In *Perceiving, acting, and knowing: Toward an ecological psychology*, Robert Shaw and John Bransford (Eds.). Lawrence Erlbaum Associates, 67–82.
- [38] Arthur M. Glenberg, Jessica K. Witt, and Janet Metcalfe. 2013. From the Revolution to Embodiment: 25 Years of Cognitive Psychology. *Perspectives on Psychological Science* 8, 5 (2013), 573–585. <https://doi.org/10.1177/1745691613498098> arXiv:<https://doi.org/10.1177/1745691613498098> PMID: 26173215.
- [39] Denis Gračanin. 2018. Immersion Versus Embodiment: Embodied Cognition for Immersive Analytics in Mixed Reality Environments. In *Augmented Cognition: Intelligent Technologies*, Dylan D. Schmorow and Cali M. Fidopiastis (Eds.). Springer International Publishing, Cham, 355–368.
- [40] André Ricardo Abed Grégio, Alexandre Or Cansian Baruque, Vitor Monte Afonso, Dario Simões Fernandes Filho, Paulo Lício de Geus, Mario Jino, and Rafael Duarte Coelho dos Santos. 2012. Interactive, Visual-Aided Tools to Analyze Malware Behavior. In *Computational Science and Its Applications – ICCSA 2012*, Beniamino Murgante, Osvaldo Gervasi, Sanjay Misra, Nadia Nedjah, Ana Maria A. C. Rocha, David Taniar, and Bernady O. Apduhan (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 302–313.
- [41] S. G. Hart. 1986. NASA Task Load Index (TLX). Volume 1.0; Paper and Pencil Package.
- [42] Daniel Helgesson and Per Runeson. 2021. Towards Grounded Theory Perspectives of Cognitive Load in Software Engineering. In *Psychology of Programming Interest Group (PPIG)*. Psychology of Programming Interest Group. Psychology of Programming Interest Group Annual Workshop 2021 ; Conference date: 21-06-2021 Through 25-06-2021.
- [43] Hex-Rays. 2023. Ida Pro binary code analysis tool. <https://hex-rays.com/ida-pro/>
- [44] Adrian Hoff, Lea Gerling, and Christoph Seidl. 2022. Utilizing Software Architecture Recovery to Explore Large-Scale Software Systems in Virtual Reality. In *2022 Working Conference on Software Visualization (VISSOFT)*. 119–130. <https://doi.org/10.1109/VISSOFT52527.2022.00020>
- [45] Nina Hollender, Cristian Hofmann, Michael Deneke, and Bernhard Schmitz. 2010. Integrating cognitive load theory and concepts of human-computer interaction. *Computers in Human Behavior* 26 (11 2010), 1278–1288. <https://doi.org/10.1016/j.chb.2010.05.031>
- [46] E. Hollnagel and D.D. Woods. 2005. *Joint Cognitive Systems: Foundations of Cognitive Systems Engineering*. CRC Press. <https://books.google.vu/books?id=IwRHwOK2IzYC>
- [47] E. Hornecker, P. Marshall, and J. Jörn Hurtienne. 2017. Locating Theories of Embodiment Along Three Axes: 1st–3d Person, Body-Context, Practice-Cognition. In *CHI 2017 workshop on Soma-Based Design Theory*.
- [48] P.N. Johnson-Laird. 1983. *Mental Models*. Cambridge University Press.
- [49] David Kirsh. 2013. Embodied Cognition and the Magical Future of Interaction Design. *ACM Transactions on Computer-Human Interaction* 20, 1, Article 3 (apr 2013), 30 pages. <https://doi.org/10.1145/2442106.2442109>
- [50] Gary Klein, Jennifer K. Phillips, Erica L. Rall, and Deborah A. Peluso. 2007. A data-frame theory of sensemaking. In *Expertise out of context: Proceedings of the Sixth International Conference on Naturalistic Decision Making*, R. R. Hoffman (Ed.). Lawrence Erlbaum Associates Publishers, 113–155.
- [51] Andrew J. Ko, Brad A. Myers, Michael J. Coblenz, and Htet Htet Aung. 2006. An Exploratory Study of How Developers Seek, Relate, and Collect Relevant Information during Software Maintenance Tasks. *IEEE Trans. Softw. Eng.* 32, 12 (dec 2006), 971–987. <https://doi.org/10.1109/TSE.2006.116>
- [52] George Lakoff. 1994. *Master Metaphor List*. University of California.
- [53] Alwin Maier, Hugo Gascon, Christian Wressnegger, and Konrad Rieck. 2019. TypeMiner: Recovering Types in Binary Programs Using Machine Learning. In *International Conference on Detection of intrusions and malware, and vulnerability assessment*.
- [54] Alessandro Mantovani, Simone Aonzo, Yanick Fratantonio, and Davide Balzarotti. 2022. RE-Mind: a First Look Inside the Mind of a Reverse Engineer. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX.
- [55] Richard E. Mayer. 2005. *Cognitive Theory of Multimedia Learning*. Cambridge University Press, 31–48. <https://doi.org/10.1017/CBO9780511816819.004>
- [56] Laura G. Militello and Robert J. B. Hutton. 1998. Applied cognitive task analysis (ACTA): a practitioner’s toolkit for understanding cognitive task demands. *Ergonomics* 41, 11 (1998), 1618–1641. <https://doi.org/10.1080/001401398186108> arXiv:<https://doi.org/10.1080/001401398186108>
- [57] George A. Miller. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review* 63 (1956), 81–97. <https://doi.org/10.1037/h0043158>

- [58] Jules Moloney, Branka Spehar, Anastasia Globa, and Rui Wang. 2018. The affordance of virtual reality to enable the sensory representation of multi-dimensional data for immersive analytics: from experience to insight. *Journal of Big Data* 5, 1 (2018). <https://doi.org/10.1186/s40537-018-0158-z>
- [59] D.A. Norman. 1988. *The Psychology of Everyday Things*. Basic Books.
- [60] Donald A. Norman. 1993. *Things That Make Us Smart: Defending Human Attributes in the Age of the Machine*. Addison-Wesley Longman Publishing Co., Inc., USA.
- [61] Megan Nyre-Yu, Karin Butler, and Cheryl Bolstad. 2022. A Task Analysis of Static Binary Reverse Engineering for Security. In *55th Hawaii International Conference on System Sciences, HICSS 2022, Virtual Event / Maui, Hawaii, USA, January 4-7, 2022*. ScholarSpace, 1–10. <http://hdl.handle.net/10125/79608>
- [62] Roy Oberhauser and Carsten Lecon. 2017. Gamified Virtual Reality for Program Code Structure Comprehension. *International Journal of Virtual Reality* 17, 2 (Jan. 2017), 79–88. <https://doi.org/10.20870/IJVR.2017.17.2.2894>
- [63] Fred Paas, Alexander Renkl, and John Sweller. 2004. Cognitive Load Theory: Instructional Implications of the Interaction between Information Structures and Cognitive Architecture. *Instructional Science* 32, 1 (01 Jan 2004), 1–8. <https://doi.org/10.1023/B:TRUC.0000021806.17516.d0>
- [64] pancake. 2023. Radare2: Libre Reversing Framework for Unix Geeks. <https://www.radare.org/n/radare2.html>
- [65] Peter Pirolli and Stuart Card. 2005. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. (2005), 2–4. https://analysis.mitre.org/proceedings/Final_Papers_Files/206_Camera_Ready_Paper.pdf
- [66] Eric Prather, Arash Shabbaz Badr, Bruno Simões, and Raffaele de Amicis. 2020. A systematic literature review on dynamic cognitive augmentation through immersive reality: challenges and perspectives. In *Defense + Commercial Sensing*.
- [67] Jennifer Preece, Yvonne Rogers, and Helen Sharp. 2019. *Interaction Design: Beyond Human-Computer Interaction* (5 ed.). Wiley, Hoboken, NJ.
- [68] H. G. Rice. 1954. Classes of Recursively Enumerable Sets and Their Decision Problems. *Journal of Symbolic Logic* 19, 2 (1954), 121–122. <https://doi.org/10.2307/2268870>
- [69] Frank E. Ritter, Farnaz Tehranchi, and Jacob D. Oury. 2019. ACT-R: A cognitive architecture for modeling cognition. *WIREs Cognitive Science* 10, 3 (2019), e1488. <https://doi.org/10.1002/wcs.1488> arXiv:<https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/wcs.1488>
- [70] Simone Romano, Nicola Capece, Ugo Erra, Giuseppe Scanniello, and Michele Lanza. 2019. On the use of virtual reality in software visualization: The case of the city metaphor. *Information and Software Technology* 114 (2019), 92–106. <https://doi.org/10.1016/j.infsof.2019.06.007>
- [71] Mike Scaife and Yvonne Rogers. 1996. External cognition: how do graphical representations work? *International Journal of Human-Computer Studies* 45, 2 (1996), 185–213. <https://doi.org/10.1006/ijhc.1996.0048>
- [72] Lawrence Shapiro and Shannon Spaulding. 2021. Embodied Cognition. In *The Stanford Encyclopedia of Philosophy* (Winter 2021 ed.), Edward N. Zalta (Ed.). Metaphysics Research Lab, Stanford University.
- [73] Ben Shneiderman and Richard Mayer. 1979. Syntactic/semantic interactions in programmer behavior: A model and experimental results. *International Journal of Computer & Information Sciences* 8, 3 (01 Jun 1979), 219–238. <https://doi.org/10.1007/BF00977789>
- [74] Janet Siegmund, Christian Kästner, Sven Apel, Chris Parnin, Anja Bethmann, Thomas Leich, Gunter Saake, and André Brechmann. 2014. Understanding Understanding Source Code with Functional Magnetic Resonance Imaging. In *Proceedings of the 36th International Conference on Software Engineering (Hyderabad, India) (ICSE 2014)*. Association for Computing Machinery, New York, NY, USA, 378–389. <https://doi.org/10.1145/2568225.2568252>
- [75] Zachary D Sisco, Patrick P Dudenhofer, and Adam R Bryant. 2017. Modeling information flow for an autonomous agent to support reverse engineering work. *The Journal of Defense Modeling and Simulation* 14, 3 (2017), 245–256.
- [76] Sean Smits. 2022. *Enhancing Binary Analysis through Cognitive Load Theory*. Master’s thesis. Arizona State University.
- [77] M.-A. Storey. 2005. Theories, methods and tools in program comprehension: past, present and future. In *13th International Workshop on Program Comprehension (IWPC’05)*. 181–191. <https://doi.org/10.1109/WPC.2005.38>
- [78] John Sweller and Paul Chandler. 1994. Why Some Material Is Difficult to Learn. *Cognition and Instruction* 12 (1994), 185–233.
- [79] John Sweller, Jeroen J. G. Van Merriënboer, and Fred Paas. 2019. Cognitive Architecture and Instructional Design: 20 Years Later. *Educational Psychology Review* 31 (06 2019), 261–292. <https://doi.org/10.1007/s10648-019-09465-5>
- [80] Maura K. Tennor. 2015. *Reverse Engineering Cognition*.
- [81] Timothy van Gelder and Robert F. Port. 1996. It’s about time: an overview of the dynamical approach to cognition.
- [82] Iris Vessey. 1985. Expertise in Debugging Computer Programs: A Process Analysis. *International Journal of Man-Machine Studies* 23 (1985), 459–494.
- [83] Daniel Votipka, Seth Rabin, Kristopher Micinski, Jeffrey S. Foster, and Michelle L. Mazurek. 2020. An Observational Investigation of Reverse Engineers’ Processes. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 1875–1892. <https://www.usenix.org/conference/usenixsecurity20/presentation/votipka-observational>
- [84] Markus Wagner, Fabian Fischer, Robert Luh, Andrea Haberson, Alexander Rind, Daniel A. Keim, and Wolfgang Aigner. 2015. A Survey of Visualization Systems for Malware Analysis. In *Eurographics Conference on Visualization (EuroVis) - STARs*, R. Borgo, F. Ganovelli, and I. Viola (Eds.). The Eurographics Association. <https://doi.org/10.2312/eurovisstar.20151114>
- [85] Leslie J. Waguespack. 1989. Visual Metaphors for Teaching Programming Concepts. *SIGCSE Bull.* 21, 1 (feb 1989), 141–145. <https://doi.org/10.1145/65294.71203>
- [86] Kirk A. Weigand and Ronald Hartung. 2012. Abduction’s role in reverse engineering software. In *2012 IEEE National Aerospace and Electronics Conference (NAECON)*. 57–62. <https://doi.org/10.1109/NAECON.2012.6531029>
- [87] Markus Weninger, Lukas Makor, and Hanspeter Mössenböck. 2020. Memory Cities: Visualizing Heap Memory Evolution Using the Software City Metaphor. In *2020 Working Conference on Software Visualization (VISOFT)*. 110–121. <https://doi.org/10.1109/VISOFT51673.2020.00017>
- [88] Margaret Wilson. 2002. Six views of embodied cognition. *Psychon Bull Rev* 9, 4 (Dec. 2002), 625–636.

- [89] Iyad Zayour and Timothy C. Lethbridge. 2000. A Cognitive and User Centric Based Approach for Reverse Engineering Tool Design. In *Proceedings of the 2000 Conference of the Centre for Advanced Studies on Collaborative Research* (Mississauga, Ontario, Canada) (CASCON '00). IBM Press, 16.