HUMAN INTENTION SPACE - NATURAL LANGUAGE PHRASE DRIVEN APPROACH TO PLACE SOCIAL COMPUTING INTERACTION IN A DESIGNED SPACE

Pronab Pal

Computer Science and Cloud Engineering Specialist, KeyByte Systems, Australia
Melbourne Australia

ABSTRACT

Publishing and sharing content through software has become a regular part of Social Computing today [the term Social Computing is used in the sense defined in Wikipedia (Social computing - Wikipedia, defined in [6].)]. This paper shows how we can achieve social cohesion despite varied software pieces working in their unique way and providing their specialized content. It defines a software methodology to design better socially responsive software by representing Intentions in code and using that as an open inter-component communication mechanism with more ownership and responsibility for both publisher and receiver. Intention Space introduces, for the first time, a close knitting of context and intentions at the core of the computation process, which has not been possible until now.

KEYWORDS

Intentions, Software Component Interaction, Execution Path, Common Path Of Understanding and Execution (CPUX) ,Social Computing, Emotional Design

1. CODING IS A DESIGN WITH INTENT. LET'S TREAT IT THAT WAY

There is no doubt in anybody's mind today that AI (Artificial Intelligence) is rising. The possibilities and trepidation are enormous. There are some missing pieces in this puzzle of perceived AI domination; how can human beings with the brilliance and mind-powers of engineers and extraordinary visionaries end up creating so much mistrust, doubt, and possible harm to a society where we face an existential crisis? What are these missing pieces? How the future of Social Computing look like?

Whenever someone writes any software code related to any social activity or a social gadget like a smartphone today, they are inherently engaged in some form of design. Even if the code has been copied and pasted from elsewhere, it carries a built-in intention and specific actions, much like a designed artifact such as a knife or spoon.

2. TRADITIONALLY INTENTS ARE NOT REPRESENTED IN CODE OR DESIGN

In the traditional programming model of ' input-process-output ', intentions stay in the coder's mind; it is absent in the code, which means the software code is devoid of intent. As it remains outside of general attention, the significance and implications of intention in software development still need to be considered and accounted for, especially when the software becomes a designed tool in society.

The absence of explicit intent in design, in general, and code in particular, is primarily due to the lack of any formal approach that can expose, analyze and persist intentions, let alone provide a mechanism to ensure that such intentions align seamlessly within a target scenario.

2.1. The Science of the Artificial by Herbert Simon

In the realm of artifact design in society, ambiguity of intent is often accepted or considered unavoidable. Herbert Simon's concept of "Bounded Rationality" in "The Sciences of the Artificial" posits that humans make decisions based on simplified models of reality [3]. This foundational work posits that design is a transformation process beginning with certain objectives (or intentions) and culminating in solutions. Intention, in this context, serves as the foundation from which all design actions emanate.

2.2. Ackoff's Systematic Approach to Problem-Solving

Ackoff's systematic approach to problem-solving emphasizes the importance of clearly articulating objectives and systematically aligning actions with those objectives to achieve desired outcomes [1]. However, the role of intentions in context remains crucial in designing systems in Ackoff's thinking.

2.3. The Meta Layer and the Intention

While the role of intention in design has long been acknowledged, it traditionally remains in the meta layer, not directly integrated into the tools of the trade [2]. This detachment results in the ambiguity and lack of formal structure for representing and managing intentions within the software development lifecycle. As researchers [8] have noted, integrating intention-driven methodologies into software service design is crucial for aligning software behavior with user needs and expectations. Researchers [5] have brought intention-driven management of cloud service , At network level Intent driven approach is gaining momentum [9,10]

In software space there has always been a chasm between developer's world context and the business world context. The importance of context in understanding and executing intentions is emphasized by Terry Winograd in "Understanding Computers and Cognition," where he discusses how the situation significantly influences the interpretation and execution of actions (Winograd, 1986)[4]. Intention Space introduces, for the first time, a close knitting of context and intentions explicitly, simultaneouslyat the core of the comprehension and computation process, which has not been possible until now.

3. INTENTION SPACE -SHARED INTENTIONS, OBJECTS AND DESIGN CHUNKS

This article introduces the concept of Intention Space, a modern approach to modular software development. It allows its operation within a design space where Intentions become part of the functioning software as an unavoidable inter-component addressing mechanism. The Intention Space framework recognizes the fundamental role of intention in software development. It aims to provide a structured methodology to capture, analyze, persist, and validate the intentions embedded in code. It exposes a new way to represent the understanding of a design and execution time computation progression in a system, typically an 'App'. The article outlines ways Intention Space can take today's software practices to more socially transparent norms.

This approach of Intention Space upholds the variability of Intentions, it brings Intention as the driver of computation at execution time,at the same time maintaining the uniqueness of intention-object-intention in the computation space , which ensures the uniqueness of context representation as computation progress and thus allows to uphold the value of situated action as described by Terry Winograd as ‘situated action’ in the book [4]

3.1. Unified Model of Computing: PnR (Prompt and Response) Computing

Current computing paradigms, which follow the von Neumann model of input-process-output, fall short when integrating persistent storage and managing context across various components. This inconsistency creates unnecessary complexity and hinders the seamless execution of software processes. Intention Space proposes a unified model that encompasses both transient and persistent data, allowing for a more coherent and efficient computational framework.

By adopting a structured model based on prompt-response pairs, Intention Space provides a robust mechanism for tracking and managing the flow of intentions and data across different components. This model not only facilitates the alignment of business goals with software execution but also ensures that every computational step is cognitively transparent and traceable. The prompt and response in Intention Space coincide with what we call prompt and response in large language model interactions. However, in our case, each prompt-response pair is uniquely identified for a specific application runtime. Additionally, each PnR can incorporate regular expressions as part of the prompt and response.
For example, in a scenario where a visitor is served a food dish from the menu, the PnRs might look like this:

```
[{ "question": "what is your name", "answer": "_" },
{ "question": "what is your name", "answer": "panda" },
{ "question": "do you have veg or non veg", "answer": "veg" },
{ "question": "dinner for", "answer": "panda" },
{ "question": "dish selected", "answer": "Vegetable-Biryani" },
{ "question": "dish made", "answer": "Vegetable-Biryani" },
{ "question": "dish presented", "answer": "Vegetable-Biryani" }
]
```

PnR serves as the global app storage as the state of the app changes. This means the execution of computation is managed through the transformation of the prompt-response set designated at design time. In a subproject, we are further exploring PnR systems to leverage the algebraic characteristics of PnR transformations for developers' benefit.

While prompts and responses (also referred to as questions and answers) are identified by unique IDs, they can also incorporate regular expressions. As computation progresses from one design chunk to the next, this pool of PnRs provides the vehicle for data passing from one design chunk to another, facilitated through Objects and Intentions.

Objects in Intention Space are unique. Unlike standard software development, where objects contain software code, Objects in Intention Space do not hold code. Instead, they receive ‘Intentions’ from one chunk of code and reflect an ‘Intention’ to another chunk of code. The chunks of code reside in a ‘Design Chunk’. This emission of Intentions and their absorption in another Design Chunk transforms the prompt-response sets, which the Objects respond to through ‘Intentions’.

This new computing model presents the progress in computation as a transformation of the PnR sets attached to different design chunks belonging to an app. In aggregation, these form a PnR set for the app, over which the transformation occurs at execution time.

3.1.1. PnR Computing in an LLM World

The new computing model provides a robust framework based on simple human-understandable phrases. While the design chunk holds all the code content, the PnR sets act as the interface between different design chunks. Along with Objects and Intentions, since PnRs are also expressed in natural language, the computation actors and the interfaces between them are expressed in natural language. This allows a natural integration of Intention Space and LLMs, where LLMs can manage Intentions, Objects, and Design Chunks and, in many cases, generate Design Chunk code content. The following sections describe how Intention Space leverages PnR computing for a healthy, socially responsive living space.

4. INTENTION SPACE: A NOVEL APPROACH

Intention Space addresses the gap between design and coding by providing a persist-able structure of computing, where the transformation of prompt-response (or prompt-response) pairs forms the basis of computation. This approach allows designers to map out multiple paths (CPUX) for achieving a goal, thereby offering a more transparent and manageable way to align software execution with design intentions.

By representing intentions, objects, and design chunks through natural language phrases, Intention Space ensures that the computational model is both understandable and operable by humans and machines alike. This alignment with natural language also lays the foundation for integrating large language models (LLMs) to further enhance the consistency and adaptability of software systems. LLMs can leverage the prompt-response structure to ensure that the intentions guiding software behaviour are aligned with the overarching goals of the application. This way Intention Space brings the correspondence between Intentions in Design and affordance in Design.

Intention Space is also a software architectural experiment in progress at enterprise software builders Keybyte Systems Australia to address the missed opportunities of being unable to represent Intent as part of the software code.

4.1. Design Chunks: Holds Software Code & Proprietary Content

In the context of Intention Space, software components are items in specific "Design Chunks" (DC) - carefully designed and structured units that embody particular Intentions. These DCs interact within a field of phrase pairs or prompt-responses (PnR Domain), emitting Intentions to Objects which reflect Intentions to other DCs. Any coded instruction in the systems of Intention Space must reside only in the Design Chunks. Intention Space doesn't bring any additional syntax or execution framework within a Design Chunk. Thus, Design Chunk acts as a scoping mechanism where any traditional piece of software can operate as usual but can be configured to be triggered by particular Intentions or emitting specific intentions at operation time. A Design Chunk can hold a series of functions with particular names recognizable within the Design Chunk. The Design Chunk in Intention Space concept exhibits a versatile scalability that accommodates both macro and micro levels of software code representation. It can encompass the entirety of an App, encapsulating complex functionalities and interactions while facilitating the containment of specific and granular logic, such as the control of a few transistors. Design chunks can also hold artistic artifacts, proprietary content etc.

4.2. Intentions: Software Component Communication Mechanism through Human Read-able Statements

The Intention Space unravels Intentions as some new kind of communication between codes and objects in people's minds. Design Chunks emit Intentions to coordinate or communicate with another Design Chunk. Design Chunks receives or consumes Intention from Objects representing things, real or virtual, in people's mind. Design Chunks can receive Intentions reflected by Objects. Intentions can carry a payload, but Intentions exist only by their description. This restriction is necessary to maintain the philosophy or design thought that any code in the systems of Intention Space must reside only in the Design Chunks. Intentions can be seen as a communication mechanism with other Design Chunks using Objects as intermediaries which can carry payloads.

4.3. Objects in Intention Space: Reflector of Intentions into Design Chunks

Objects give direction to Intentions emitted by Design Chunks, to be received by other Design Chunks. Objects represent things, real or virtual, in people's minds; compared to the traditional approach of putting code inside an object or class, Intention Space puts the code outside of the object but is invokable through Intentions defined in Intention Space. In intention space, Objects , conceptually, don't contain the software code but can be configured to direct Intentions coming from certain Design Chunks to be reflected into another Intention reaching another Object or Design Chunk. Objects can also split an Intention into several intentions. Objects participate with Intentions in the communication and coordination mechanism between Design Chunks by carrying the payload initiated by Intentions.

4.4. Instruction Execution Path Leading through intentions and Objects (CPUX)

We shall see that the Intention Space model as an operation space of Design Chunks allows identification of the execution path for every interaction in Intention Space, thereby bringing transparency without losing privacy. This is possible by co-mingling natural language phrases in the open with domain-specific terminology and concepts in a controlled manner. [Fig 1]. This execution path will be referred to as the Common Path Of Understanding and Execution (CPUX), emphasizing that Intentions in the path aid in understanding the system behaviour. Each CPUX represents a locus of a use-case as the design chunks are put to use.

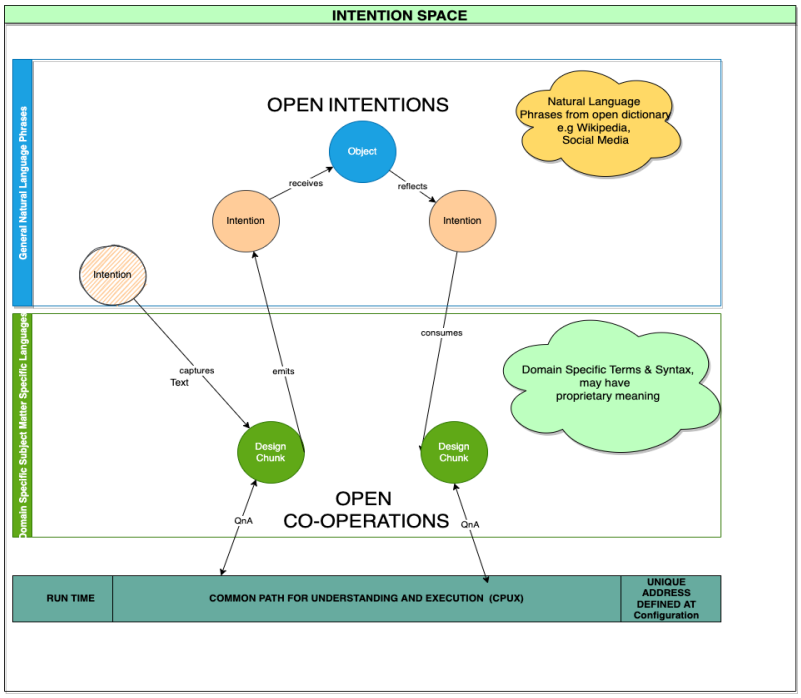

Fig 1. Intentions and Objects in natural language facilitate component interactionns

Intention Space's ability to map out CPUX for different use cases offers significant advantages in software security and maintenance. By providing a clear path of execution for each use case, it becomes easier to identify and mitigate potential security risks. Each such paths carry a enumerable set of prompts that can be mapped along the CPUX before the software is deployed for execution. This approach supports the development of more reliable, testable and accountable software systems, as the intentions guiding each action are explicitly documented and can be audited and open to some algebraic scrutiny arising from relation between a set of phrases.

5. INTENTION SPACE FOR BROADER SOCIETY AND DESIGN PROFESSION

Design chunks can hold any content in a specific context and, thus, can have any media, text, document, or software code or components. In a way, the discipline of Intention Space is beneficial not only for software development but also in any other field where a design is used in a society, including education, effectively including any field of science, arts, humanities, and architecture. The primary reason that Intention Space can bring this substantive openness and the following benefits of having Intentions as shareable between people is, it expresses the intentions and objects as some standard natural language phrases without bringing any other specialised syntax or attributes. Fig 1 illustrates this point. The arrangement also implies that the components need not be tied up with specific intentions and can be configured at the operation time.

In this broader sense, Intention Space can be considered a ‘Design Space’, although specific tools are required to make Intention Space useful in specific disciplines and real-world scenarios.
By giving a unique address for design chunks sequence for design operation by just providing the intentions that help in ‘understanding’ a design, Intention Space helps manage the gap between any design and its operation. Managing this gap is a well-known issue in any design field like Architecture and Construction.

It can have much other use in the realm of unique address and identity of creative works, even if it is rendered through media technologies, defining a safe execution scope for software code operation independent of underlying OS etc. Intention Space offers a unique advantage in the software maintenance space because every execution of software is backed by persistent address progression captured at operation time, as described below.

6. INTENTION SPACE FOR SOFTWARE OPERATION

A Design Chunk in Intention Space embodies a computational process small enough to be understood and managed by a small group of people between 1-3 from its inception to its full embodiment or deployment in a social context. A Design Chunk can be just a small part of a full App, or a Design Chunk can be a full App, Design Chunks interact with their environment through prompt-response pairs which act as a field of operation by Design Chunks with the help of the two other operators :Intention and Objects.

6.1. JSON Statement Pair Set (PnRs) as Computation Field

Every design chunk in Intention Space can be associated with a set of phrase pairs, represented as JSON objects, which are generally referred to as PnR Statement Pair in Intention Space, where the pair can include placeholders and regular expressions,

We can look at the execution of the design chunk to include either reading or writing the JSON statement pair set(PnRs) associated with the design chunk. In that sense, a Design Chunk can be looked upon as an operator that works on a field of PnRs. A design chunk with no PnR ,however, do also take part in performing computation steps of Intention Space as defined in the computation progression model , described below. In the model we shall refer to aJSON statement pair as simply as PnR. We shall see this set of PnRs provide us the domain on which the Intention Space operators operate.

6.2. Design Chunks as Software Components

A system may have several design chunks; as a system is developed ,a design chunk constantly morphs within its boundary and interacts with the outside world through well-defined intents. In the Intention space, a Design Chunk can have content in any form that makes sense to the human user. In the coding world, a single function or a collection of functions can be in a design chunk; it may have some UI/UX elements related to those functions, and there can be a prototype component to join UI/UX with functions etc. Thus the domain of operation of Design Chunks are not the numbers and strings, but they are the JSON PnR's s that make sense to a human, but is interpret-able by the machine.Intention Space takes computation progression as an act of taking a set of PnR's and producing another set of PnR's.

Instead of the traditional notion that a computation step has an input, generating an output, a Design Chunk as a software component, acts on a set of PnR's and passes PnR's to another Design Chunk as computation proceeds. The concept of 'function parameters,' a foundational element in traditional definitions of functions in computation, is looked upon as a JSON statement pairs received or produced by the Design Chunks; e.g., instead of just declaring a parameter variable called 'apple' and assigning a value, say $5 ,as the price of apple, e.g., var apple = 5 in the traditional way, Intention space equivalent will be a JSON PnR pair, represented as a JSON object: {'prompt':' price of apple','response':'$5' }. Any operation in Intention Space operates over a set of JSON statement pairs or PnRs; as an operator,a Design Chunk passes

PnRs to another operator Design Chunk through some rules of operation in Intention Space using two more operators. A particular set of PnRs are associated with a Design Chunk at design time.

6.3. Operators in Intention Space over PnRs Domain

Intention Space's provides the set of operators that operate on the PnR s and sets up the rules . Three kinds of operators given by the Intention Space are : Design Chunk, Objects and Intentions ; Each Design Chunk has a name; we can refer to it as a Design Chunk Phrase; similarly an Object Phrase is an Instance of Object, and Intention Phrase is an instance of Intention . Intention Space maintains three dictionaries of phrases corresponding to each phrase category or operator type in Intention Space as defined below.

Categories Of Phrases in Intention Space

i. Design Chunk Phrases
ii. Intention Phrases
iii. Object Phrases

The word Object has special significance in Intention Space. An Object has a name, and its only behaviour/capability is to receive and reflect 'intentions'. Objects in Intention Space don't hold any executable code; they are named entities that receive Intentions from Design Chunks and reflect Intention to another Design Chunk. A received Intention can be mapped to a different reflected Intention.

As the Design Chunk names, Intents and Objects are just human-readable entities in an Intention Space, the special method of the ' Intentions and Object' based software execution model maintains an operation time-defined structural integrity that defines the run-time address space. We shall see, from a software execution perspective, these Design Chunks provide a uniquely identifiable execution boundary for any future possible use of the software App.

7. MODEL OF COMPUTATION PROGRESSION: AN INTENTION STARTS A COMPUTATION IN A DESIGN CHUNK

7.1. Computation in a Design Chunk

In Intention Space, a computation triggered by an intention instance (i.e. a particular intention phrase) is captured by a Design Chunk, which consumes a PnR set and the design chunk either produces a prompt-response set as a result or continues the computation by emitting an intention phrase from a set of enumerable intentions, aka dictionary in the Intention Space. However, this emitted intention does not reach another design chunk directly, instead, it has to be directed or reflected by an object. This rule brings the decoupling effect of two Design Chunks; their code is never hard bound with each other at execution time.

7.2. Computation continues by reflecting an Intention from an Object

Intentions emitted from a Design chunk, in turn, are reflected by an object instance (i.e. a particular Object phrase) , chosen from an enumerable set of object phrases (aka dictionary of objects) and is subsequently consumed by another Design Chunk. The process continues, as shown in Fig 2. As mentioned, Design Chunk names come from an enumerable set of Design Chunk names, aka Design Chunk dictionary in Intention Space.

The special benefit,from the perspective of software engineering in large are :
The sequence of steps can identify any use case of the system.

A Design Chunk is not forced to implement specific intentions but can achieve desired result by using its intentions in the Intention Space

The intentions from one design chunk can be joined with any other design chunk by joining open intentions and objects chain.

7.3. Intention Space Computation Vector: from one Design Chunk to Another

A step in Intention Space computation is transitioning from a Design Chunk with a PnR to another Design Chunk with PnR, with the Intentions and Objects reflecting them, working as a control transfer mechanism between Design Chunks. As such Intentions and Objects sequence offers a protocol of 'transfer of control' independent of the underlying mechanism (e.g., event-driven or a traditional function invocation or a call module mechanism etc)

In Intention Space, a Design Chunk can be associated with a maximum of one set of prompt-response. Because the transition Of Design Chunk can only happen through intention-object-reflected intention sequence, we have a five-element vector that facilitates the transition from DC to another, represented by the starting and ending element in the vector =>***{dc, in, ob, in, dc}*** . It is also possible that the starting and ending design chunk are the same. This also mean the same design chunk can have different intentions emitted depending on the absorbed intention.

The word vector implies a specific direction. In Intention Space, this direction is the progression of computation starting from the initial design chunk leading to the final design chunk in the vector through the intermediate intention->object->intention trio while transforming a set of prompt-response to a resulting prompt-response sets in the model of computation described below.

In Intention Space Design Chunks hold the traditional software codes. It is only the interaction of the software code in a design chunk with the software code in another design chunk that Intention Space facilitates. Intention Space brings about this interaction through the vectors defining the sequence of the three types of entities :Design Chunk, Intentions, Objects.

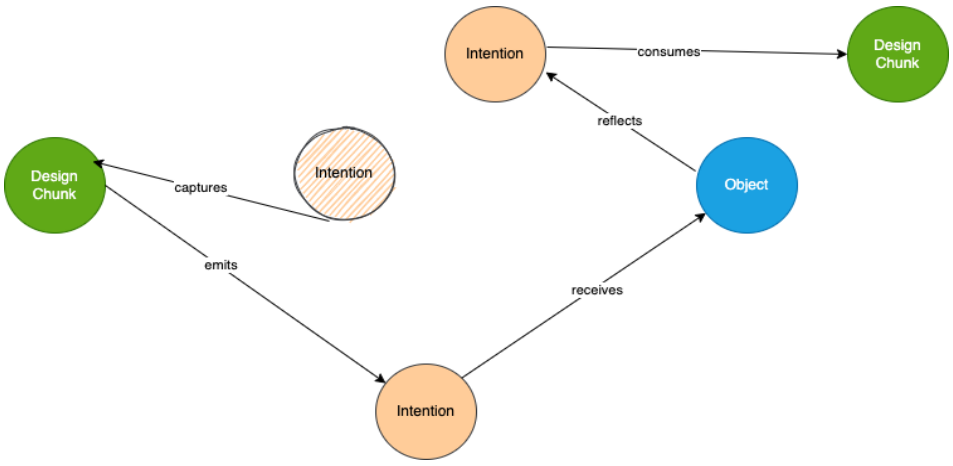

Fig 2. 5 element vector [design chunk - intention - object -intention - design chunk] in Intention Space

7.4. Execution Steps and Progression with multiple Vectors

In Intention Space, a step and progression means the transition from a PnR set to another PnR set, and this is referred to as a software progression; thus while a single vector can bring the transition from one Design Chunk to another, which may not have a PnR associated, but may emit an intention to carry the progression of computation until it reaches a Design chunk that has an associated PnR. The step in Intention Space will be the sequence of vectors that took from a Design Chunk with an associated PnR to another Design Chunk with its associated PnR.

Looping and Conditionals in Intention Space can be brought into the Design Chunks through the payload in the Intentions, which may indicate to reflect to the originating Objects where the intentions were reflected from.

Design Chunks themselves can hold an Intention Space and have several sub-Design Chunks. Thereby creating the necessity to have a registry of Intention Spaces. This article will not get into those aspects with multiple Intention Space,however, work at Keybyte Systems is in progress to bring a frame work called Context Flow which shall address those possibilities.

7.5. Illustration of Intention Space computation progress actions

The computation model of Intention Space is illustrated through the accompanying NodeJS (JavaScript) code in the file attached, which makes the relation between design chunks and code more explicit. The code implements a demo scenario where a guest wants dinner and is served according to the guest's choice. Notice that Intention Space, compared to the traditional approach , promotes an inside-out design of the Objects where Objects only have a name, and all the relevant design details about the behaviour or data structure in the code only come out as intentions reflected by the object into a design chunk. As the system is executed, a flow is created for each execution instance that carries the prompt -responses in each Design chunk.

Let us imagine a scenario where a customer asks for dinner and gets served the kind of dinner he/she wants. Firstly ,let us capture the primary Object, Intention, and Design chunks to start designing a functional system that caters to this scenario in a JSON data structure holding the objects ,intentions and design chunks.
:

```
/* Objects,Intentions and Design Chunks Example */
const jsonData = {
  objects: [
    {id:"ob1", name: "Dining Space", reflectors: [{ receives: "I want dinner", reflects: "Schedule a
dinner plate" }] },
    {id:"ob2", name: "Weekly schedule", reflectors: [{ receives: "Schedule a dinner plate",
reflects: "Prepare a dinner plate" }] },
    {id:"ob3", name: "Dinner Plate", reflectors: [{ receives: "Prepare a dinner plate", reflects:
"Present a dinner plate" }] },
  ],
  intentions: [
    { id:"in1",name: "I want dinner", PnR: [] },
    { id:"in2",name: "Schedule a dinner plate", PnR: [] },
    { id:"in3",name: "Prepare a dinner plate", PnR: [] },
    { id:"in4", name: "Present a dinner plate", PnR: [] },
  ],
dcs: [
    {id:"dc1", name: "Requesting dinner ", emits: [{ intention: "I want dinner", Object: 'Dining
```

```
Space' }], receives:[ { intention: "I want dinner", Object: "none" }] ,
PnR:[{ prompt: "what is your name", response: "_" },{ prompt: "do you have veg or non veg",
response: "_" }],
    invoke:'dc1_invoke(flow,dc.PnR)'},
    {id:"dc5", name: "Requesting a drink ", emits: [{ intention: "I want a drink", Object: 'Dining
Space' }], receives: null },
    {id:"dc4", name: "Scheduling a dinner", emits: [{ intention: "Schedule a dinner plate", Object:
'Weekly schedule' }], receives: [{ intention: "Schedule a dinner plate", Object: 'Dining Space' }],
    invoke:'dc4_invoke(flow,"do you have veg or non veg")',
PnR:[{ prompt: "dinner for", response: "_" },{ prompt: "dish selected", response: "_" }], },
    {id:"dc2", name: "Preparing a dinner", emits: [{ intention: "Prepare a dinner plate", Object:
'Dinner Plate' }], receives: [{ intention: "Prepare a dinner plate", Object: 'Weekly schedule' }],
PnR:[{ prompt: "dish made", response: "_" }],
    invoke:'dc2_invoke(flow,"dish selected")',},
    {id:"dc3", name: "Presenting a dinner", emits: null, receives: [{ intention: "Present a dinner
plate", Object: 'Dinner Plate' }],
PnR:[{ prompt: "dish presented", response: "_" }],
    invoke:'dc3_invoke(flow,"dish made")'}
  ],
};
```

The above Json structure holds enough information for full-filling the intention ‘I want dinner’ by a nodeJS script. In the realm of Intention Space the design chunks will hold the particular UI/UX components for the corresponding functionality and will have the means of executing the relevant functions within its own scope. For the illustration here we are just printing a console log with any function in a design chunk conveniently coded in the same file. The Json structure also shows the Design Chunk having executable codes which are independently invoked by the Intention received by the Design Chunk and it shows the intention emitted by a design chunk. As the system is executed a flow is created for each execution instance that carry the prompt - responses in each Design chunk . The full code for the example is attached. When we run the code ,the code registers the type and name of entities as it encounters starting with the invocation of Invoke_intention(’I want dinner’).

```
/* illistration of unique sequence of the intention-object-intention joining two Design Chunks */
  operation Sequence: [
  'ci:I want dinner',
  'dc:Requesting dinner ',
  'ei:I want dinner',
  'ob:Dining Space',
  'ri:Schedule a dinner plate',
  'ci:Schedule a dinner plate',
  'dc:Scheduling a dinner',
  'ei:Schedule a dinner plate',
  'ob:Weekly schedule',
  'ri:Prepare a dinner plate',
  'ci:Prepare a dinner plate',
  'dc:Preparing a dinner',
  'ei:Prepare a dinner plate',
  'ob:Dinner Plate',
  'ri:Present a dinner plate',
  'ci:Present a dinner plate',
  'dc:Presenting a dinner'
```

]
notation:
(ci:consumed intention,dc:design chunk,ei:emited intention,ob:object,ri:reflected intention)

The flow above illustrates the Intention Space model of computation . Treating each participating entity of the model as a node , we can represent the code in a graph diagram as below which also shows the relationship between the entities formed through intentions, which form the essential glue between objects and design chunks.

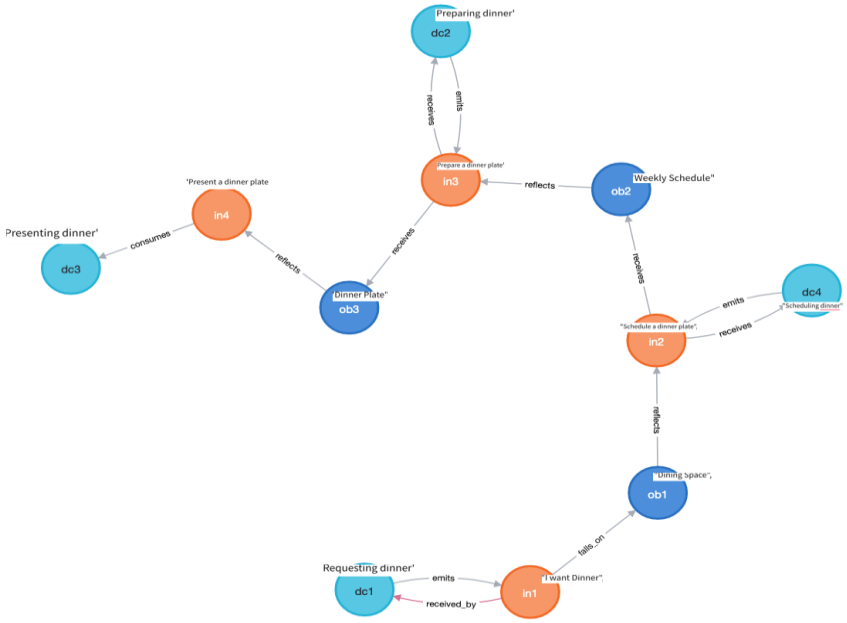

Fig 3. The Graph Diagram of Intention Space with DCA
:dc1.in1.ob1.in2.dc4.in2.ob2.in3.dc2.in3.ob3.in4.dc3

Although it is a contrived example, the code used captures the essence of Intention Space. The App in the Intention Space progresses through Design Chunks, and the execution code, in turn, is represented by the sequence of three 5-element vectors,

which is determined at operation configuration time through the sequence => dc1.in1.ob1.in2.dc4.in2.ob2.in3.dc2.in3.ob3.in4.dc3, referred to as Design Chain Address [DCA]. As the execution proceeds, the DCA sequence carries a 'flow' object which picks up any new PnR elements from the Design Chunk as it is poised for execution, and then the DC member function can access the flow object to update PnR.

8. VALUE PROPOSITION OF INTENTION SPACE IN SOFTWARE DEVELOPMENT

8.1. Looking At Every Run Time Execution Piece in Design Chunks through a Static Address Formed at Operation Time

Intention Space, meant for creating a manageable and understandable software execution environment, has the built-in characteristics of being able to break up the design chunks into smaller and manageable development chunks too.

Intention Space tackles the hard problem of managing re-usability, modularity, and context maintenance over time in the software development process by bringing together the understand-ability of software code through intention progression along with the unique vector addresses of the executable components (design chunks) while also persisting the sequence of the vectors as the computation progress(i.e., in the context of the actual usage of the design chunk at execution time) as the locus point for the particular usecase.

An Intention Space addresses the Design Chunk code pieces through a shared pool of Intentions and Objects phrases. The fact that there is only one intention instance of a given phrase that joins an object and a code piece which, in turn, is open to receive any intention and reflect it to another intention, opens up the possibility of having a unique sequence (1 or more) of the five element vectors joining two Design Chunks corresponding to every possible execution instance of software codes generating the design chain address [Common Path for Understanding and Execution or CPUX].

8.2. Composing, Testing, Prototyping, Maintenance, and Security management through CPUX

With the CPUX defined at operation configuration time, we get a unique advantage over the traditional software architecture where at execution time instance of a traditional binary chunk of code can not be tagged with anything defined at operation time[e.g., a reactJS component can not be uniquely addressed every time it is rendered on the browser in the application lifecycle component rendering chain]. Intention Space is not bound by the inability of binary code to reveal intentions allocated to it at design time. By design, every Design Chunk execution instance in Intention Space gets a design time boundary address in the form of the Design Chain Address for the execution. This opens up the possibility of using CPUX for prototyping and use case verification.

In Intention Space, composing software pieces, testing them, prototyping and maintaining them can be controlled and managed through operation time-defined definitions of objects and intentions and design chunks.

This allows Intention Space to have a real advantage in not only software maintenance but also in software security handling. Because the intents of any software code can be exposed without revealing the software code, it allows us to develop a security protocol based on the agreement of intentions before the use of any code. The detail of these implementations is in progress.

8.3. Managing Intentions, Objects, and Design Chunks

The advantage of Intention Space solely relies on our ability to manage Objects and Intentions at operation time while easing the cognitive burden on the designers and coders while making the software. A typical App can have hundreds of Intentions and Objects with PnR sets and subsets adopted by the design chunks. As Intention, Objects, and Design Chunks are identified by human-readable phrases in natural language, managing them well so that designers can choose the right Intentions, Objects, and Design Chunks and reuse them or map PnRs between different design chunks is a good challenge to be solved by the present day Large Language Models of AI. Suppose we can accurately identify intentions and objects that cross subject matter boundaries. In that case, it opens up possibilities for creating new language models to bridge different domains and facilitate more advanced and specialized natural language processing tasks.

8.4. When Objects are Only Intention Reflectors to Behaviors - Absence of Hierarchy

Intention Space also makes room for multiple Objects with different intentions or the same object with different intentions to point to the same design chunk or code piece at execution time, making the reuse of code in multiple design chunks through different prompt-response sets. While this particular positioning of Object drastically differs from traditional Objects in Software, which is supposed to hold some behaviour, Intention Space opens up computing in an phrasestructured space with much richer possibilities regarding managing software's Security, Performance, Maintainability, and Reusability.

The inherent characteristic of Design Chunks in Intention Space, where they don't depend on each other during development but can be integrated at operation time, aligns well with the principles of software development. This characteristic of loose coupling, which is a key design principle in software engineering.

Loose coupling promotes independence and modularity, allowing different components (or Design Chunks) to be developed and maintained independently of each other. This brings several benefits to software development, including:

1. Reusability: Design Chunks can be reused in different contexts, reducing redundancy and promoting efficient development.
2. Maintainability: Independent development and modularity make it easier to update or modify individual Design Chunks without affecting the entire system.
3. Scalability: New Design Chunks can be added without disrupting the existing system, enabling scalability and flexibility.
4. Collaboration: Different teams can work on different Design Chunks concurrently, fostering collaboration and speeding up development.

The ability of Intention Space to support loose coupling and independent development of Design Chunks while ensuring smooth integration at operation time makes it well-suited for complex software systems and social integration scenarios. It enhances flexibility, adaptability, and maintainability, which are crucial factors for successful software development and social integration in today's dynamic and ever-changing environments.

9. RATIONALITY AND BENEFIT OF INTENTION SPACE IN SOCIETY

9.1. Psychology of Computing through three emotions

The rationality of bringing the object phrases and intention phrases as coordinates of software codes in Design Chunk occurrence in a design space comes from the observation that we,as humans, have the tendency to be driven by a visceral perception of things around us mentally - either in reality or in a virtual world - expressed as '**What**' or the '*Object phrase*' , which triggers a motivational or utility prompt of '**Why**' or the '*Intention Phrase*', which in turn, is followed up with a more reflective analysis of the situation, leading to, the '**How**' of a design around that object of 'What', which the *Design Chunk* holds together.

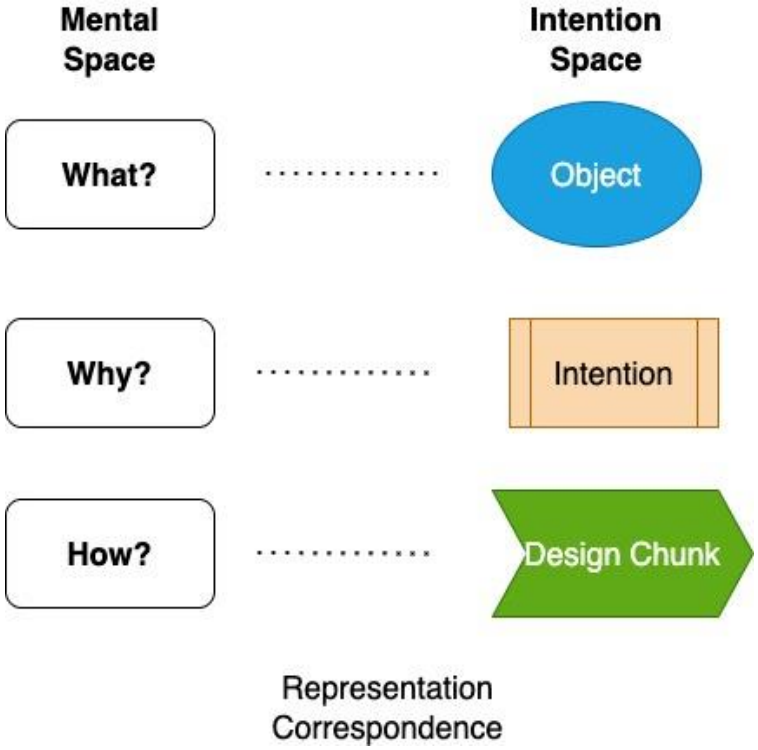

Fig 4. Basis of Intention Space

This observation, the basis of our work on Intention Space as in Fig 4, aligns with observations made by the famous UI/EX expert Dr Don Norman in his book on Emotional Design (*Emotional Design - Wikipedia*, n.d.). While Dr Norman takes his view on the UI/UX of what is designed, Intentional Space allows one to take it to the micro units of software development. It opens up a means of human communication between the designer and the user of an artifact.

10. USE OF DESIGNED ARTIFACT AND SOFTWARE EXECUTION ANALOGY: RE-ENGINEERING AFFORDANCE

The whole approach of Intention Space to software development is hinged on our view of any software artifact as a design piece, similar to any design artifact like a spoon, a door, or a staircase, where the execution of the software piece is akin to the usage of the design artifact in real life. As the software developed in Intention Space carries the extra dimension of its behaviour and specification being narrowly defined within the boundary of a design chunk at execution time with some specific intentions, we believe Intention Space opens up new possibilities in the Design world where affordance can be re-engineered based on human-readable Intentions embedded with design, giving pointers to the ‘why’ of any design artifact.

By considering software as a design piece, Intention Space emphasizes the importance of defining clear intentions and behaviours for each design chunk, similar to how a physical artifact is designed with a specific purpose and function in mind. The execution of the software, driven by these intentions, aligns with the usage of a physical artifact based on its design and intended purpose.

10.1. Open Pool of Shared Intents in a Society that Transacts Artifacts

An App in an Intention Space will typically have hundreds of Intentions. As intention work as pointers to code, sharing Intentions will be a common practice when developing multiple Apps in an Intention Space.

Sharing Intentions across multiple artifacts in social settings has a similar benefit in cooperative design. People design and construct things in components because there is a cognitive load boundary of how much an individual or group can handle and manage. Many times components build a hierarchy by putting one component within another.

Intention Space provides a linear representation of Intents, which allows for easy access to shared Intentions from multiple components while an Artifact is in use. This accessibility of Intentions at execution time enhances the design-time decisions and enables a more coherent and flexible user experience.

For example, in a car's context, having contextual prompts like "where is the temperature control" accessible while driving (associated with the "Cool my Car" intention) allows users to interact with specific design chunks without being burdened by the complexity of the entire car's design. Similarly, users can access other Intents that point to different design chunks like "Radio in Car" as needed, further enhancing the user experience.

So in effect, Intention Space makes the design time decisions readily accessible at execution time.

Exposing prompt-responses for any design artifact also allows the prototyping of design artifacts in an embedded environment, as we can work with the prompt-response pairs from other design chunks already in place.

11. CPUX: REMOVING UNACCOUNTABILITY, UPHOLDING PRIVACY

In Intention Space, there are always at least two design chunks whenever a social interaction happens. A CPUX is formed, which holds a unique identity connecting a provider of Intention and a Consumer of Intention. Any use of software happens only by a two way interaction between aprovider and the consumer,by mutual consent. A two-way interaction will have two unique CPUX. This makes it possible to make every user action within Intention Space identifiable without breaking the privacy of the design chunk.

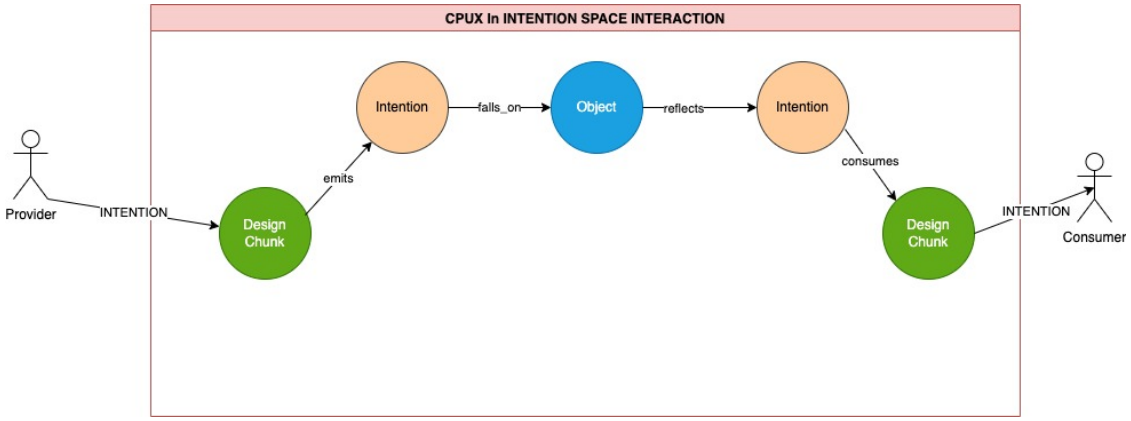

Fig 5. Interaction is always identifiable through unique CPUX

12. SOCIAL COMPUTING & EDUCATION THROUGH APPS GOVERNED BY INTENTION SPACE

12.1. Collaborative Apps Development

By looking at software code as an artifact or a Design Chunk and deriving prompt-response pairs from other sets of prompt-response pairs, we can introduce a more human-centric approach to software development. This approach allows for a deeper understanding of human intentions and behaviours, enabling the creation of more intuitive and user-friendly applications.

The use of ordinary phrases in human interaction as Intentions and Objects in Intention Space simplifies the language used in software development, making it more accessible to a wider range of people. This can lead to increased participation and collaboration in software development, as individuals from diverse backgrounds can contribute to the design of applications.

Furthermore, with a good collection of artifacts (Apps) built in Intention Space, AI technologies can leverage common Intentions and Objects to build newer systems and generate prompt-response sets. This can lead to the development of AI-driven applications that are better suited to user needs and preferences, as AI can adapt and customize applications based on user interactions and feedback. The traditional input-process-output or request-response model of information server does not reveal the true nature of social computing in a way where Applications communicate with each other, making the internet a fairer place for any application to be judged by its content and ability to cooperate with other Apps on the same device.

12.2. Secured Inter-personal and Social Communication

The concept of “operator-approved intentions” and the use of design chunk CPUX can enhance the security and privacy of interpersonal communication in social computing. By defining and approving specific intentions for communication, users can have more control over their data and interactions, making the Internet a fairer and safer place for users.

The availability of Intentions upfront and the possibility of having a proof of concept before using a registered Intent in Intention Space can significantly impact the legal and regulatory aspects of software development and usage. By having a clear and transparent record of the Intentions used in the development of applications, there is a level of accountability and traceability that can be established.

12.3. Software Development Ethics

This approach can help address legal and ethical concerns related to software development, as it provides a verifiable trail of the intentions behind the design of an application. Traditional laws that emphasize intent can be easily applied and enforced in Intention Space, as the intent is explicitly stated and recorded, allowing for easier identification of any malicious or fraudulent activities.

Moreover, clear Intentions documentation can aid in protecting intellectual property rights and copyright. Software developers can register their Intentions and design chunks, ensuring their innovative work is recognized and protected from unauthorized use or duplication.

12.4. Smarter Education Systems

Intention Space can play a transformative role in the education system by providing a common path of understanding and execution (CPUX). By structuring educational content as design chunks, each can offer a contextual and focused learning experience around specific Intentions and Objects.

The flexibility and adaptability of Intention Space allow educators to create dynamic and interactive educational material that aligns with hands-on experiences and real-world problem-solving. Instead of presenting information in a linear and rigid format, Intention Space enables the development of educational content that adapts to the learner's progress and understanding.

For instance, in geometry theorem proving, each theorem and axiom could be represented as a design chunk with its own set of Intentions and Objects. Learners can engage with the material by exploring Intentions and Objects that guide them through the logical progression of the theorems. They can interact with the content, ask prompts, and receive relevant responses that lead them towards a deeper understanding of the subject.

Furthermore, Intention Space facilitates collaboration and sharing of educational content among educators, enabling the development of a rich ecosystem of educational resources that can be adapted and customized for different learners and learning environments.

Overall, Intention Space can revolutionize education by providing a platform for contextual and interactive learning experiences, where learners can actively engage with the subject matter and develop a deeper understanding of the concepts through hands-on exploration and discovery.

13. THE RISK OF NOT HAVING AN INTENTION SPACE

Today, our communities often extend globally via the internet. This has expanded our circles of empathy, allowing us to connect with diverse people, causes and the reality worldwide. It also has the potential to dilute personal responsibility, as our actions can feel removed from their impacts. Our time is marked by the coexistence of two seemingly contradictory trends: the globalization and homogenization of culture and the resurgence and appreciation of local, indigenous cultures. It's a delicate balance between appreciating, preserving, and reviving indigenous cultures' richness while integrating the advantageous aspects of modern, global cultures. The art and science of Computing are inherently cultural, but the lack of representation of human intentions as first-class citizens in the design of social computing systems can create a significant cultural gap between AI developers and those who do not engage with the technology. Elevating intentions to a first-class status allows for their direct presence with human users, eliminating the need for an intermediary agent and fostering a more seamless interaction between users and technology.

The impact of technology like Artificial Intelligence without building a space where the primary currency of human interaction is ignored will have unwanted and avoidable outcomes that push human progress backwards or have unimaginable repercussions like the act of the explosion of a nuclear bomb over human dwellings that occurred in a culture and time where there was no way to exchange or build upon intentions of ordinary people humanely. Intention Space offers only

one small step in making Intentions expressable, shareable and understandable by people from different cultures and different ways of living using technology; however, currently, it is the only platform that promises that for any software construction.

Accompanied Code dinnerspace.js: illustrates the execution flow in Intention Space

REFERENCES

[1] Ackoff, R. L. (1974). Redesigning the Future: A Systems Approach to Societal Problems. John Wiley & Sons.

[2] Ouali, Sami &Kraiem, Naoufel& Ben Ghezala, Henda. (2012). From intentions to software design using an intentional software product line meta-model. 66-71. 10.1109/INNOVATIONS.2012.6207776.

[3] Simon, H. A. (1969). The Sciences of the Artificial. MIT Press

[4] Winograd, T., & Flores, F. (1986). Understanding Computers and Cognition: A New Foundation for Design. Ablex Publishing Corporation.

[5] W. Chao and S. Horiuchi, "Intent-based cloud service management," 2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), Paris, France, 2018, pp. 1-5, doi: 10.1109/ICIN.2018.8401600.

[6] Parameswaran, M., &Whinston, A. B. (2007). Social Computing: An Overview. Communications of the Association for Information Systems, 19, Article 37. https://doi.org/10.17705/1CAIS.01937

[7] Norman, D. (2004). Emotional Design: Why We Love (or Hate) Everyday Things. Basic Books.

[8] Fki E., Soulé Dupuy C., Tazi S. and Jmaiel M. (2010). Intention Driven Service Composition with Patterns.In Proceedings of the 12th International Conference on Enterprise Information Systems - Information Systems Analysis and Specification, pages 236-241 DOI:10.5220/0002904102360241

[9] L. Pang, C. Yang, D. Chen, Y. Song and M. Guizani, "A Survey on Intent-Driven Networks," in IEEE Access, vol. 8, pp. 22862-22873, 2020, doi: 10.1109/ACCESS.2020.2969208.

[10] A.Mankin, D. Massey, Chien-Lung Wu, S. F. Wu and Lixia Zhang, "On design and evaluation of "intention-driven" ICMP traceback," Proceedings Tenth International Conference on Computer Communications and Networks (Cat. No.01EX495), Scottsdale,AZ, USA, 2001, pp. 159-165, doi: 10.1109/ICCCN.2001.956234.