Clinical Psychiatry
ISSN: 2471-9854

Open Access Review Article

On the Foundation of the Repetitive Cognitive Behaviour in Software Development and Software Use

Pronab Pal*

Department of Computer Science, The Australian Defence Force Academy (ADFA), UNSW, Australia

ABSTRACT

This paper proposes "Intention Patterns" as the automatic human base for software development. A model called Virtual Cognitive Model (VCM) is presented, which repeats but remains transparent in software making and use. VCM effectively becomes a real-time meta-level control of user interface and program execution. Patterns play a fundamental role not only in software design but also in human cognition and system sciences. This paper explores the concept of intention patterns within a Virtual Cognitive Machine (VCM) framework, highlighting their relevance in software development and user interaction. Drawing parallels with repetitive learning in physical activities like swimming, the paper argues that repetition forms cognitive patterns essential for both developers and users in understanding, constructing, and executing software systems. Intention Patterns, defined as sequences of scenes resolved through transitions of Focus, provide a structured framework for understanding how human minds process and manipulate complex information. The VCM acts as a dynamic mental construct that aids in software design and usage by facilitating the resolution of intentions across interconnected scenes. By formalizing these cognitive processes through a Met model, the paper proposes insights into enhancing software development methodologies and user experience through a deeper understanding of cognitive patterns.

Keywords: Pattern; Software design; Software engineering; Pattern composition; Cognition

INTRODUCTION

Patterns, in general, have received interest in software during the last ten years, especially as design patterns [1]. However, patterns play a more vital role in system sciences and complexity theory [2]. There have been several approaches to the composition of design patterns in software through pattern systems or UML [3,4]. However, none of these approaches takes into consideration the fact that patterns have got a foundation in our minds and come as a common cognitive element in the design, development, and use of computer systems. To see how patterns are closer to human cognition than just a good design principle, I shall draw an analogy.

Repetition and Cognition

For a human being, repetition is the key to building complex, rich, and beautiful things. A good swimmer, for example, learns more amazing techniques in fast swimming by sheer practice by repeating the same strokes over and over again several hundred times. There the repetition holds out a pattern in the swimmer's mind that makes him/her a better

Received:	10-August-2023	**Manuscript No:**	IPCP-23-17270
Editor assigned:	14-August-2023	**PreQC No:**	IPCP-23-17270 (PQ)
Reviewed:	28-August-2023	**QC No:**	IPCP-23-17270
Revised:	15-January-2025	**Manuscript No:**	IPCP-23-17270 (R)
Published:	22-January-2025	**DOI:**	10.36648/2471-9854.11.1.56

Corresponding author: Pronab Pal, Department of Computer Science, The Australian Defence Force Academy (ADFA), UNSW, Australia; E-mail: pronob@visualanalytics.com.au

Citation: Pal P (2025) On the Foundation of the Repetitive Cognitive Behaviour in Software Development and Software Use. Clin Psychiatry. 11:56.

swimmer. The exact nature of this pattern is not in the repetition; repetition provides a frame for the swimmer to build a way in the mind that fuses with proper motivation from the swimmer and somehow makes him a better swimmer both mentally and physically.

We don't know the exact process by which a good swimmer picks up the skills. The same practice and drill give different results to different swimmers. However, we know even a good swimmer has not learned the techniques in a day. The feel for the water, the whole body synchronization, and a sense of balance, other motivations everything has probably contributed to and helped him become a good swimmer.

Repetition and Software

In the software field, the software developer gets the feel required to develop good software by interacting with items that are not as concrete as the water in the swimming pool. Instead, he has to build up most of this world in his mind: A world where he can develop good software through mental constructs or through interaction with other existing pieces of software, which in turn are designed by other software developers.

Like the swimmer, the software developer too has to repeat certain drills about good programming and design practices. However, for the software developer, repetition is not only in how he develops it but also in what he develops. Firstly, how software of a certain kind is made is learned by the developer by trying to develop the same kind of software several times. He learns how certain situations can be best resolved by sheer experience in solving certain kinds of problems. Secondly, any software is supposed to be run an infinite number of times under certain conditions. Thus the software is supposed to repeat a set of behavior when it goes out in the real world. Good software is supposed to do that without any conflict in the real world where it executes in terms of what it does internally. It has to maintain and honor the memory boundary designed for it and not create any conflict in terms of interaction with any kind of user-the operator or the end user. In this paper, I use the terms 'developer' and 'user' in a generic sense anyone who contributes something towards the development of the software piece is a developer, and whoever interacts with the finished software piece to achieve a goal is a user. This user may be another programmer or a business person.

In many instances, a software user goes through a mental model-building phase where he can learn most of the necessary details about what the software is designed to do repeatedly. In doing so, the user of the software goes through repeating a part of the model building and mental interaction that the builders of the software went through.

Proposition

This paper, firstly, proposes a theory that states that human beings do share a common mental model to develop, use and execute software. This model brings about the repetition that is necessary both at the design stage and also at the execution phase. The model thus acts as the water where the developer and the user act as the swimmers-giving a framework for repetition in a way that holds out some patterns to the software user and the developer. The recognition of these patterns, being a cognitive affair, remains independent of the hardware or even the application.

This paper also proposes that such a model, which remains hidden and transparent within human behavior during software making and use, can be made explicit in a Met model in the software development and use phase. Such exposure to this model will bring benefits in software design, making, and usage of the software.

My main objective is to reveal this common cognitive model in all software activities.

Organization

I shall first define a few terms and concepts about how we analyze the real world. Based on this analysis, I shall build up a structure called an "Intention Pattern". Then I shall introduce virtual cognitive machine to bring out the dynamism inherent in such structures. The structure and the dynamism are proposed repetitions in our regular software activity.

LITERATURE REVIEW

Analysis of Reality

Attributes and relations: Attributes are generally ascribed to entities, e.g., my car can have an attribute like color, type, etc. However, it is also possible to see such attributes as some attributes of our own mind. The fact that every time I see my car, I find it red is a fact, and if I find it different someday, I would want to find a reason for the change of color: Either I have gone color blind, or someone has changed the color of my car, or there should be some explanation to it. The fact that every time I find it red is a brilliant joint act of my mind as well as the car.

The same aspect can be discerned in any relationships we establish between items. E.g., if I say my piece of my cake is larger than yours, that relation between your piece of cake and mine is a characteristic of the pieces of cake as well as my power of observation, a characteristic of my mind.

Intentions: From this perspective, I define the term Intention as an attribute of the mind we have. It is not necessary to define here 'what the mind is.' I shall use it as the common English word we know: That part of us that listens, understands, and acts coherently in reality. Intention is defined as an attribute of our mind which brings the attributes and relation between items in the world. Intention can bring about attributes and relations through the following five types of actions in mind:

- **Observation:** When we make a statement like the sky is blue.

- **Question/answer:** When we agree or not agree to a fact through question and answer. For example," Is Mt Everest the highest mountain peak?" and answer "Yes"; here, the person who raised the question intended to relate all the mountain peaks in the world in a generic way to Mt Everest, and the answer has resolved that intention one way or the other.
- **Abstraction:** When we establish some relationship between things we can see or feel immediately and those that we cannot see or feel in the same way: e.g., "the egg in my hand is oval like a balloon."
- **Rationalization:** When we say certain things are related in a certain way because some other things are related in a certain way, e.g., "the winter is coming, so the tree leaves are falling."
- **Command:** When we wish some relationship to come about: e.g., "Put the cup on the table".

An intention can be one of the above types.

Focus

There is a limit on how many items a mind can focus on at any moment. Of course, it might vary from person to person, and also, with time, how many things a person can focus on at one point in time. I shall define a Focus as a collection of items with the intention that somehow binds those items together. For example, when I am watching the night sky, I am focusing on small parts of the sky and trying to make a shape out of the placement of the stars in the night sky. The intention is to make some abstract shapes like a rider on a horse. This can be broken up into several related intentions, like making the tail of the horse, making the head, and so on.

Key Points and their Types

The definition of focus implies some definition of items that the focus consists of. These items have been chosen by the person who is creating the focus, and I shall call them the key points. The key points have a type that gives some characteristics to intentions which we shall see next. The type of the key points gives us a way to categorize the key points. E.g., an apple in the real world will be of a different type to the apple represented in the computer even though they both are called apples. For us human beings, they feel different. The type of an item is, however, closely related to the focus and intentions in the focus. E.g., if I am going to pick up an apple lying next to some oranges to offer it to my son who has asked for an apple, in the focus of the action of picking up the apple, I shall consider an apple is a different type than the oranges; however, if my intention is to offer some fruit, I would consider apple and orange are of the same type in that focus.

Binding of Key-Points

When we focus on something, there is always one or more intention which is part of that focus. These intentions will have some key points associated with it. E.g., 'I want to withdraw some money' being the intention, the key points in focus may be 'ATM', 'amount,' and 'my account'; I shall call such an association a binding of key points by the intention. The binding also establishes a relationship between the type of intention it refers to and the type of the key point (s) in the binding. Thus if some of the types of key points bound by an intention are of different types, the Intention doing the binding is of type abstract.

Scenes

The above notions of intentions and focuses are more related to a person engaged in some skillful activity. However, no human being is isolated from the rest of the world during such an engagement. This brings us to the notion of scene-a rough boundary of things the person is aware of at a point in time or over an extended time while he or she is engaged in some skillful activity.

The above definition of scene also distinguishes between what we build in computers and what is out there in the real world. The distinction is that what we see in our computers are segments of the real world. The reality in computers is already bounded or limited to the action of some skillful activity by one or more persons. The real world is a continuous phenomenon, whereas the moment we visualize and represent the world through our computers, we lose the continuity of the real world.

They become isolated segments of the real world; even though we are watching a continuous video, the whole video will be an isolated segment of the real world. In the software world, reality thus comes as a set of scenes, each scene having any number of abstract or physical items put together, all in the same league. For example, a scene can have an ATM physical machine and the account number, which is abstract.

Similarly, a scene can have items created by other software already running in the real world.

Also, a scene should have one main focus reflecting what the person is focusing on for a specific skillful engagement. It may have several other focuses reflecting what other focus the person might get attracted to, and that is somehow related to the main focus through sharing of some key-points. The main focus gives a characteristic and name to the scene. The other focus brings in the other possible focus. The person engaged in the skillful activity, as indicated by the main focus, might focus on relating the items in the main focus to the reality of the computer world.

Hence my definition of scene: They are segments of the real world that one is aware of while one is engaged in a skillful activity having one main focus reflecting the skillful activity and several associated focus to help the main focus.

There is a particular vagueness in the definition of a scene, which is deliberate. It will not interfere with the formality I am presenting here. The vagueness is a reflection of reality where the scene belongs. That is, the scene is a dynamic construct from reality to suit the person engaged in some skillful activity. A diagrammatic representation of the scene follows (**Figure 1**):

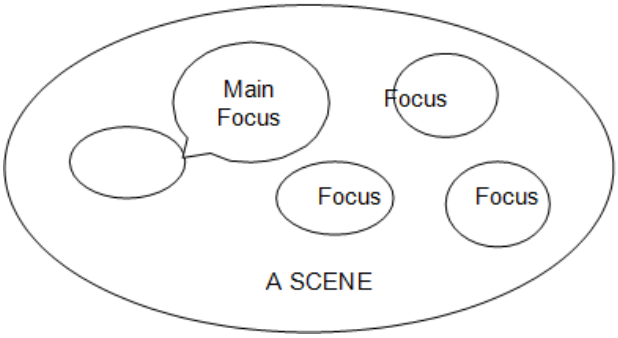

Figure 1: A diagrammatic representation of the scene.

A typical example of a scene will be a few related pages in a book, a limited sequence of consecutive dialogues, events, and screenshots in a movie, or particular user computer interaction when the user is trying to do a task the user has in mind. A main focus, "Transfer of money to and from ATM," can have other focus in the scene like "Location of the ATM' which will be influenced by the policy of the Council, type of residence around, etc.

Connecting the focuses: Next thing I shall introduce something that connects several focuses in our minds. This is some kind of pattern that makes way for the relation between several isolated scenes and provides dynamism in the mind. This dynamism is part of the nature of patterns [5]. Because of the special nature of my construct, I shall call my patterns "Intention Patterns". The concept of resolution of Intention and transition of Focus is part of the dynamism that intention patterns bring in. It is necessary to define these terms before I define intention pattern.

Resolving an intention: An intention is resolved when the intention relates to the key points, all of which belong to the current focus. When this is not the case, we say the Intention causes the transition.

Transition of focus: A transition of focus occurs when the intention that relates some key points in the focus also includes some key points not included in that focus. Such a transition thus involves a group of key points in the two focuses, which are related through the intention.

Intention patterns: Intention patterns are a sequence of focuses through sequences of Scenes such that there is a continuous resolution of intentions through that sequence with the transition of focus. In other words, intention patterns weave through consecutive scenes, creating one or more focus in each scene. For example, I may walk out of my house with the intention pattern "to draw some money" This pattern may consist of scene sequences as follows: "My house front," "the street to the highway, the petrol station at the corner" and "the ATM." There are no hard and fast rules about how the scenes are formed. They are just a sequence of familiar segments of the real world, each having a main focus. The main focus may or may not belong to the Intention pattern. This is illustrated in the diagram below (**Figure 2**):

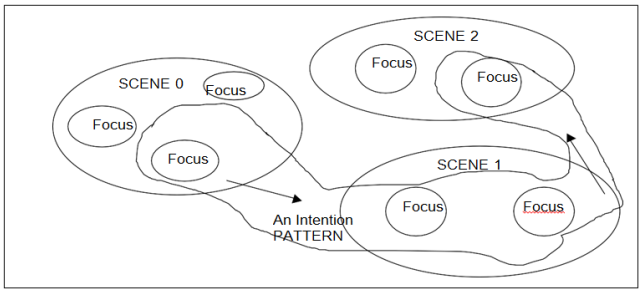

Figure 2: An intention pattern.

Also, an intention pattern may have a type that shall provide the role of the intention pattern.

Modelling the Dynamism in Intention Pattern

So far, I have introduced the terms and concepts that are needed to analyze the real world. Intention pattern holds dynamism in our mind through a change of focus. How can we model dynamism? This I propose to do in my model virtual cognitive machine. Our mind seems to have a Virtual Cognitive Machine (VCM) that keeps on working while we busily go about our daily activities. "Intention patterns" are the repeating structures that our VCM manipulates and does a virtual execution of software in our mind holds. Intention Patterns are also the common currency of interchange between software developers and users.

DISCUSSION

Virtual Cognitive Machine

The virtual cognitive machine being part of the human way of doing computation can be found as a common frame of repetitive action that the software developer and user engage in. In other words, our mind holds this virtual cognitive machine while we engage ourselves in daily activity, build and use software in the real world. The reason I have used those three words is: It is virtual in the sense there is no real hardware in the sense of electronic hardware in our mind; it is definitively cognitive one shall recognize it by observing one's own mind; and thirdly, it is a machine because, if we follow our instincts about facts, rationality, our sense of abstractness, along with our human faculty of working in small context while aware of larger issues, the switch or transition between the contexts or between focuses in the pattern or across pattern do occur almost involuntarily as if we ourselves not responsible for the switch. Most importantly, VCM allows us to do our regular day-to-day work while it busily builds up the world for us. VCM brings us some form of continuity. The continuity has a structure and dynamics that repeats with any software activity. VCM provides the framework for this repetition. Crucial to the operation of VCM is the concept of resolution of intention in a focus. We shall see, through intention resolution, the intentions in a scene are the main drivers in the transition of scenes.

The Flow in VCM

Whenever an intention is not resolved in a focus, the VCM carries it along until it is resolved in a scene in the intention pattern. VCM also collects the unresolved intentions and the key points associated with an unresolved intention. An intention can thus originate in one scene and get resolved in another focus in another scene in the pattern.

Our memories do play a vital role in the modeling of the VCM.

VCM operates by going from one focus to another along an intention pattern, which I defined as a sequence of scenes. The consecutive focuses that VCM goes through may be within the same scene or in the next scene in the sequence. At each focus, it goes through the intentions and may collect the key points that the intention bounds. The key points within a scene correspond to our short-term memory. When VCM goes from one scene in the intention pattern to the next, it adds the key points that it already collected in the first scene to its collection "Contextual Collection," which VCM creates on its own. This contextual collection corresponds to our longer-term memory, and any unresolved intention carried in contextual collection will get resolved in some scene in the sequence in the current pattern or in a pattern that is pointed to by an intention in a scene in the current pattern. This sort of resolution through patterns may bring in some nesting of patterns.

Nesting of Intention Patterns

Nesting of patterns can arise in two situations. Firstly, an intention in a scene in a pattern possibly bound to certain key points in the scene points to another pattern for its resolution. E.g., a cooking recipe may point to a pattern of making the vegetable stock while outlining the steps to prepare fried rice.

The second way nesting may arise is by "shelling" a pattern with another pattern. This means whenever VCM starts to traverse the shelled pattern, it first traverses the shelling pattern first. E.g., whenever I go to pick up my daughter from her school, I want to check the day's timetable on the fridge first. Here the pattern "picking up my daughter" will be shelled by the pattern "look up the day's timetable."

Resolution and Continuity

We have seen the intentions bring about the dynamism needed for switching from one focus to another. However, the important aspect of this switching within an intention pattern is that the switching should provide a certain kind of continuity in meaning between scenes. The act of resolution of intentions provides this continuity. An intention pattern ensures an Intention is resolved in one of the Scenes in the pattern or is being directed to another pattern for resolution.

The pattern through the VCM is represented in the following diagram (**Figure 3**):

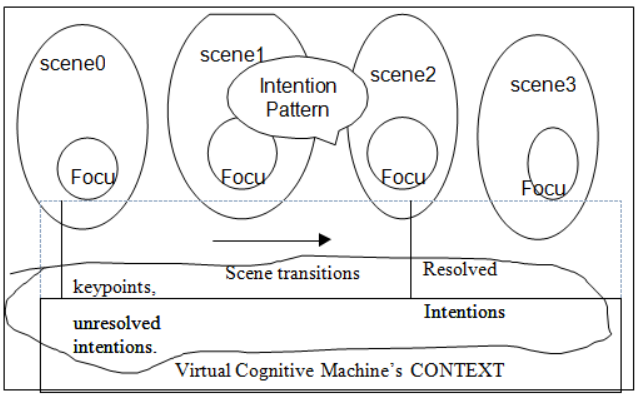

Figure 3: Virtual Cognitive Machine's context (VCM).

Summary

Next, I shall go through, in summary, an example of how VCM is working while I go on about building a small piece of software. This is a simulation of VCM while I developed the working program in Java language. I have chosen a small and simple sample that will allow us to focus on the process of software making rather than the application. At the same time, it has got a hint of the complexities of a real-world application. I start with a requirement that may come from a business analyst and go through the analysis and design phase to code the software. In the following, the word pattern is used to imply an "intention pattern."

The requirement is to make a visible dial called "Moody Dial," which shows a white arc on a yellow dial. The dial is manipulated using a slide bar.

The angle the arc makes is variable with the slide. The angle reflects the mood or amount of interest of the customer in the item that is being discussed during an interview.

I shall ignore, for this exercise, the business justification, ethical fitness, and appropriateness of usage of the software.

The case study follows some conventions in regard to the resolution of intentions in every Focus in a Scene and a few other little details, which I explain below:

Patterns are represented as Pn, where n is an integer, and each Pn has a name and a type. The type indicates the role of the person who sets up this pattern.

Scenes are represented as Sn, where n is an integer, each Sn has a name. Also, each Scene carries a note which describes the focus in plain English. For each pattern, there is a Scene, with the same name, as the starting scene.

An intention in focus is represented as follows:

Name (Type).

The type can be one of the following

O: Observation

A: Abstraction

R: Rationalisation

C: Command

Q: Question/Answer

I have represented the following patterns:

P1: Customer brief (business analyst)

P2: Making of the dial (Analyst)

P3: Objects in the customer dial (Analyst)

P4: MVC-model view controller in customer dial (Analyst)

P5: Flow of events (Programmer)

P6: Test script (Tester).

The scenes involved are:

S0: Customer brief

S1: Moody dial

S2: Dial operation

S3: Change flexibility

S4: Making of a dial

S5: Objects in moody dial

S6: MVC in moody dial

S7: Dial frame

S8: Arc panel

S9: Angle adjustment listener

S10. Flow of events

S11 observable angle

S12: Constructor

S13: Test script

With the above numbering, the patterns come out as follows:

P1: [S0,S1,S3]

P2: [S4,S1,S2,S3]

P3: [S5,S8,S9,S11]

P4: [S6]

P5: [S10,S12,S8,S11,S9]

P6: [S13,S0,S1,S2,S3]

Some of the intentions in the scenes are:

What is required (Q) What is going to change (A) Improve interview effectiveness (C).

How do you represent customer mood? (A), Who Controls the Slide? (Q)

The customer behavior package interface provides the angle of mood. (O) What links the slide-bar movement with the change in the angle(R)

In summary, “customer brief” goes through how a requirement might be specified by a requirement analyst, starting with a summary of the business objective. It gives a rough specification of the functions required with some reference to operational deployment issues and finally talks about the future goal and changes.

The pattern "Making of the dial" comes from the analyst who goes to the specifics of individual function but still does not go through the show of things, nor does it go into details of how components coordinate. It shares some of the scenes from the "Customer brief" pattern. It brings a new operational issue scene but does not elaborate too much on it. It outlines how coding should coordinate with future changes. In the pattern “Objects in customer dial," the analyst goes into much detail into technical details. However, those details are still at the individual component level. Analyst “shells” the pattern "Objects in customer dial" by the shell "MVC-model view controller in customer dial". This pattern gives a brief overview of MVC and raises intentions which are all resolved in the pattern "Objects in customer dial." Finally, the programmer sets up the pattern "Flow of events," which goes into detail about the programmer's vision of how mouse-click is transferred to the change in the dial using the classes as vision by the analyst. “Test script” is for testing the program.

Implications

The study of VCM reveals that the pattern acts as the attraction or grouping agent of the two components: the intentions and the key points, while it makes its way through the scenes. This collection is referred to as the focus of each scene. I intend to do some empirical study on the following possibilities:

- The intentions and resolutions have some correlation with the pattern type.
- The type of key-points and their distribution through the intention pattern has some relationship to the role of the Pattern.
- The pattern should bring some characteristics to the nature of communication between project participants.
- Nesting of patterns is more common at the analysis level than at the programming level.

It is interesting to observe because the programmer is dealing with a pre-structured view of objects; the programmer's pattern includes more "R" type intentions. These intentions are trying to rationalize the process flow.

CONCLUSION

This model VCM is the model that repeats by design or by implication whenever good working programs are constructed. Yet, today's computer, which should have all the necessary ability to repeat this model, does not represent it anywhere in the system cycle. A model like VCM can be the common model above the machine language, roughly at the same level as the byte code in the Java engine. That way, any programming language or business description language can describe its own domain in terms of the operation of the VCM. The benefit of doing this is the business user and developer use the same reference model to exchange requirements and to work for a requirement. The lack of this

sort of common model, which is a common way of thinking among humans, is the main reason computers today are not suitable for cooperative development, and they are challenging to program. The proposed VCM model is not a framework because a framework would have some executables as part of the framework and could generate codes. Any code for any machine can be put in this model for both human reading and machine processing. Because patterns express the overall plan, this model can coexist with real readable material (e.g., specification of any standard) and the corresponding code (in any standard programming language). Unlike byte-code for Java, VCM, however, can work within its model while the program is executing, and there can be interplay between the control of VCM and the control of the program. This is possible by tracking scene flow at the program level as it would be done by VCM and then transferring the control to VCM when the program finishes its task. VCM effectively then becomes a real-time meta-level control on both user interface and program execution.

In the immediate future, my plan is to study the behavior of the distribution of intentions, by their types, across different roles in the project of software development and how such a distribution possibly reflects the overall quality of the project.

In the long run, a closer study of VCM will provide a more objective and generic view of patterns as repeating structures and how they can be related to the workings of our minds.

REFERENCES

1. Shalloway A, Trott JR (2004) Design patterns explained: A new perspective on object-oriented design. Addison-Wesley Professional.
2. Fritjof Capra (1997) The Web of Life: A New Synthesis of Mind and Matter. Flamingo, 1997.
3. Schmidt DC, Stal M, Rohnert H, Buschmann F (2013) Pattern-oriented software architecture, patterns for concurrent and networked objects. John Wiley & Sons.
4. Yacoub SM, Ammar HH (2001) UML support for designing software systems as a composition of design patterns. Springer Berlin Heidelberg.
5. Alexander C (1979) The timeless way of building. New york: Oxford university press.