Introduction of “Empiricism” to Traditional Approach of Computing

Submitted by

-

Satyanand Jha, University ID 1055536
Department of Computer Science
University of Warwick

ABSTRACT

Digital development has given flexibility and speed for information processing, but on the other hand we are formalising nearly everything that we intend to process or compute. While this process of formulising, it seems that we are limiting our ability to observe and losing out our free thought process to rule based thinking, usually pre-defined by people in order to digitalise the system. This motivates the introduction of “Empiricism” to our traditional approach of computing. This paper discusses basic features of Empirical Modelling (EM) and tries to highlight some of its advantages over other traditional programming methodologies.

1. A GENERAL IDEA OF EVOLUTION OF KNOWLEDGE

Learning in life starts with observation. We observe things knowingly and unknowingly. The knowledge accumulated through inspection and interpretation of observation is often called as acquaintance. In simple language, this acquaintance is usually a conglomeration of logical mappings of information and calculations in our mind. The acquaintance is further formulised for common acceptance. In general, the aim of scientific work is to improve the relationship between our actual experiences (observations of the world) and the scientific definitions (acceptance of theories and concepts about the world).

Justification of acceptance depends on a constant re-interpretation and a re-examination of the observation. Therefore, both the above mentioned process (acceptance and observation) need to go hand in hand for the proper acceptance and adaptation of knowledge evolution in the dynamic human civilization.

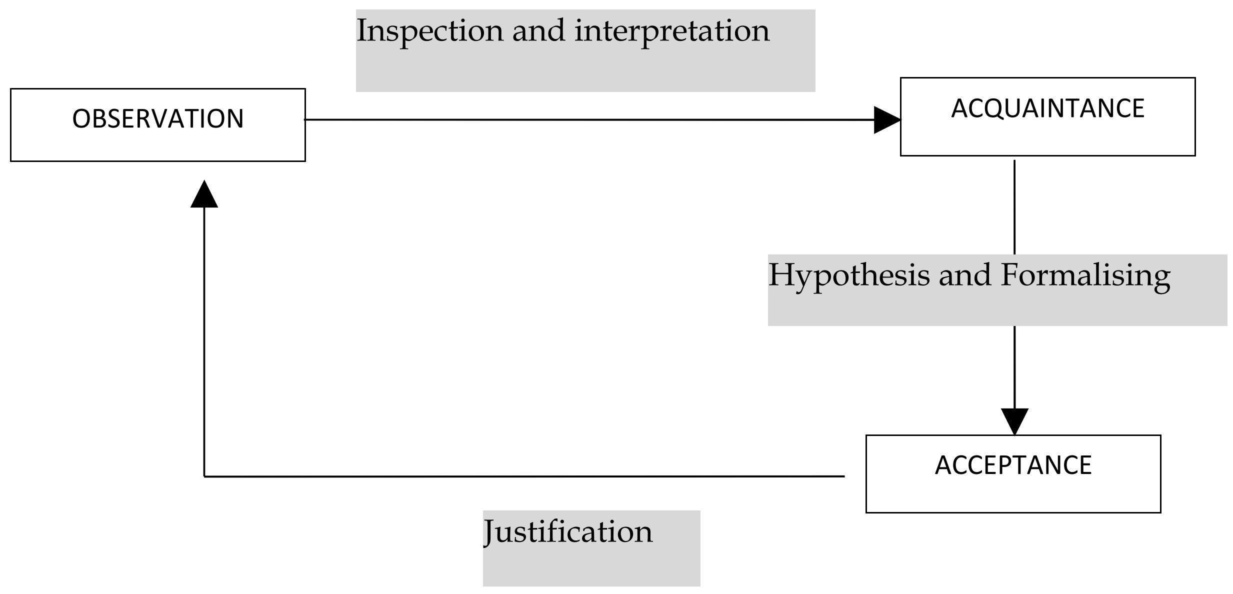

FIGURE 1. KNOWLEDGE EVOLUTION

On the basis of above mentioned knowledge evolution, Empirical Modelling (EM) intends to improve the relation between our actual experiences (observations) and its computerization. In other words, it provides a complete picture of computing; capability of computing along with responsibility to restore the free thought process.

2. COMPUTERIZATION OF KNOWLEDGE
(A traditional approach)

Figure 2 shows traditional way of computerization of knowledge. The domain experts are people who hold the real world knowledge and its contextual dynamics. This knowledge is passed on to system analyst that usually thinks from the perspectives of computerising it rather than emphasising on the contextual correctness and human interpretation during development states.

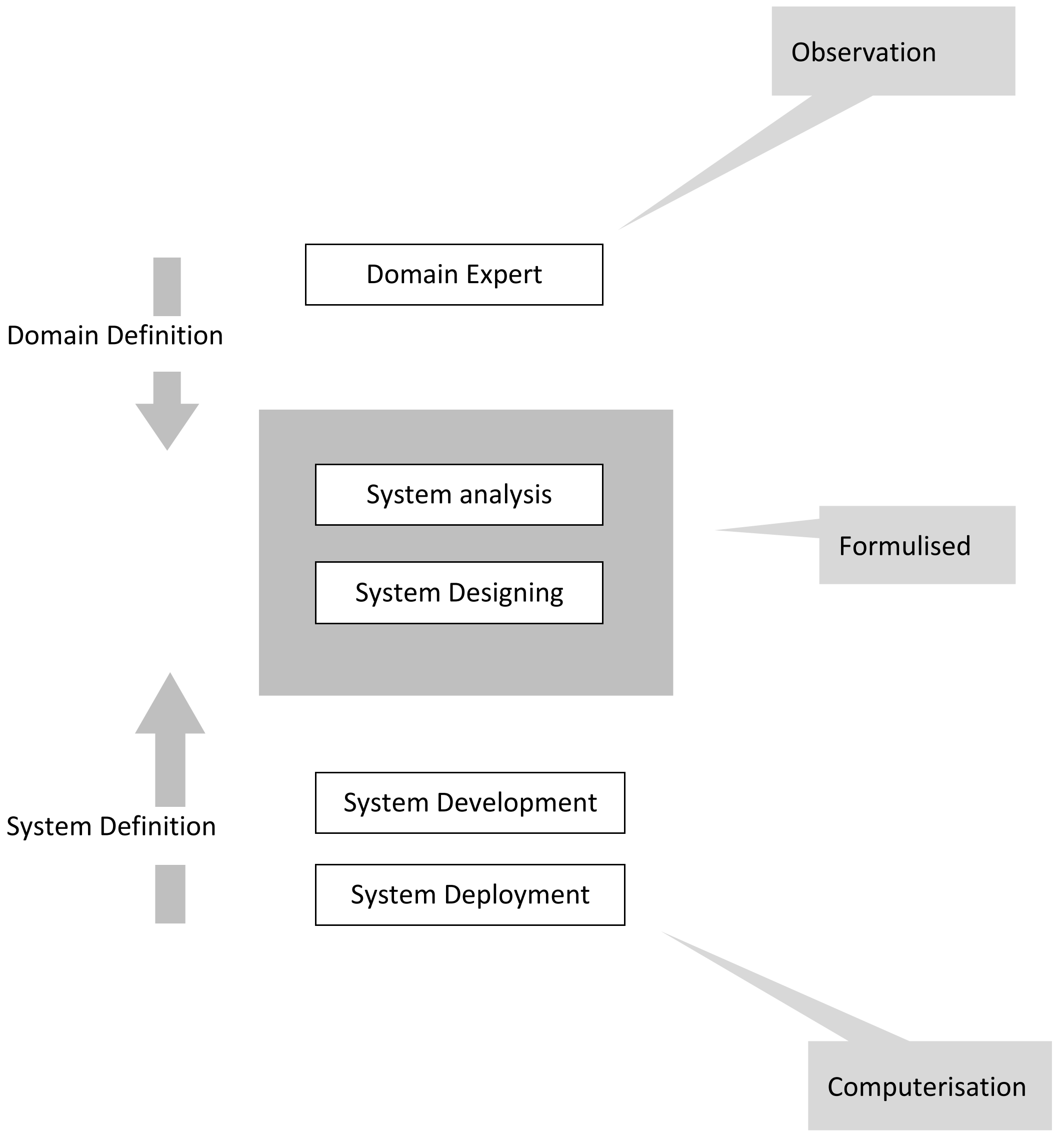

FIGURE 2 Computerization of knowledge

The process of computerization takes place by defining the knowledge in terms of program logics and algorithms, which can be implemented on to the computer. At this stage the empirical nature of the context starts to differ it's meaning to the process of formalising. Further stages include simple implementation of these logics and algorithm that isolates and restricts the knowledge under a formal boundary.

3. COMPUTATIONAL SYSTEMS

In general, computational systems can be classified as being either transformational or event-based.

3.1 Transformational System

In the transformational definition, a system is a function that receives one or more system inputs (I) from an external environment, transforms them with process (T), and then releases them as system outputs (O) to an external environment [1]. This input-output (I/O) relationship can be expressed symbolically as

T (I) = O or T : I -----> O.

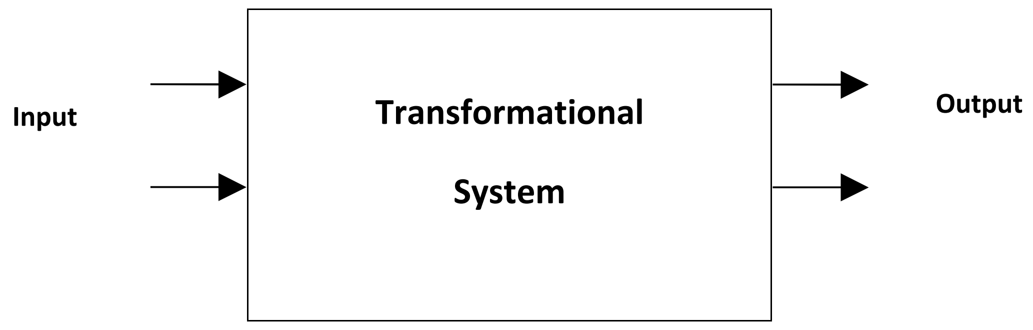

FIGURE 3 Transformational System

The Formal Methods approach to software construction is similar to the concept of transformational systems. It treats a program and its execution as mathematical objects and applying mathematical and logical techniques to specify and analyze the properties and behaviour of these objects [2]. Formal Methods are usually specified in terms of input/out. However, many of our real life experiences are not merely mathematical or logical problems which can be solved through formal method approach; rather they are constant interactions with our environment, usually defined as reactive systems.

3.2 Reactive systems

Reactive systems are systems whose role is to maintain an ongoing interaction with their contextual environment rather than produce some final value upon termination. These systems typically exist to collaborate or interact with some entity or entities in their contextual environment. In general, they never have all of their inputs ready; rather, the inputs arrive in endless and perhaps unexpected sequences [3]. Typical examples of reactive systems are controlling mechanism for trains, planes, or ongoing processes such as a nuclear reactor etc. Such programs are usually specified and verified in terms of their behaviours.

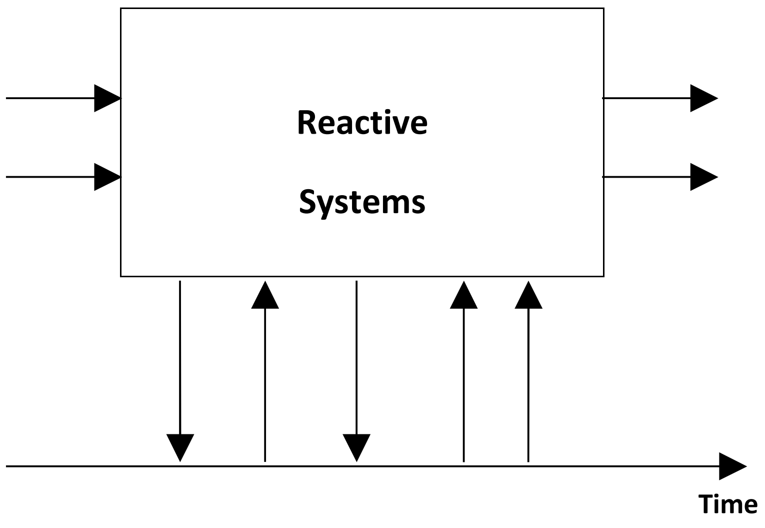

FIGURE 4 Reactive system

The computation techniques or programming languages developed have been majorly developed around the formal methods and transformational systems but they do not suffice the need of reactive systems. It is widely acknowledged that the current techniques have yet to meet the challenge of the software specifications for reactive systems, and that the development of such specification necessarily involves an iterative process, of preliminary design, simulation, analysis and revision. Therefore, it shall be argued that how far the traditional computation can compute the real life scenarios or shall we actually use the word “compute” while we are in the process of constant interaction and observation.

4. PROGRAMMING LANGUAGES

Programming languages, in general, are form of defining algorithm and data in order to perform calculations and manipulations. An algorithm *(which is important part of traditional programming)* is any set of detailed instructions which results in a predictable end-state from a known beginning.

It is therefore stated in [4] that the mathematical theory of computation, with its origin that predates the computer, has been most influential in the development of the programming languages, tool and methodology.

However, things that we experience in everyday life do not follow standard set of rules (they are dynamic rather than static) and our experience or observations cannot be computed or calculated on the basis of pre-defined rules. It is therefore argued that the programming problems we have resolved since 1950 are primarily those within the scope of the conventional mathematics theory of computation whilst the difficulties of the reactive systems engineering stem from the limitation of the present theoretical framework [5].

4.1 Traditional Paradigm

With an imperative approach, a developer writes code that describes the detail of the steps that the computer must take to accomplish the goal. These statements are said to change the state of the program as each one is executed in turn. Figure 1 shows the traditional programming paradigm.

With an imperative approach, a developer writes code that describes the detail of the steps that the computer must take to accomplish the goal. These statements are said to change the state of the program as each one is executed in turn. Declarative programming attempts to minimize or eliminate side effects by describing what the program should accomplish, rather than describing how to go about accomplishing it. Declarative programming tries to remove the problem of non-contextual states just by removing it. This is a contrast to imperative programming, which requires an explicitly provided algorithm.

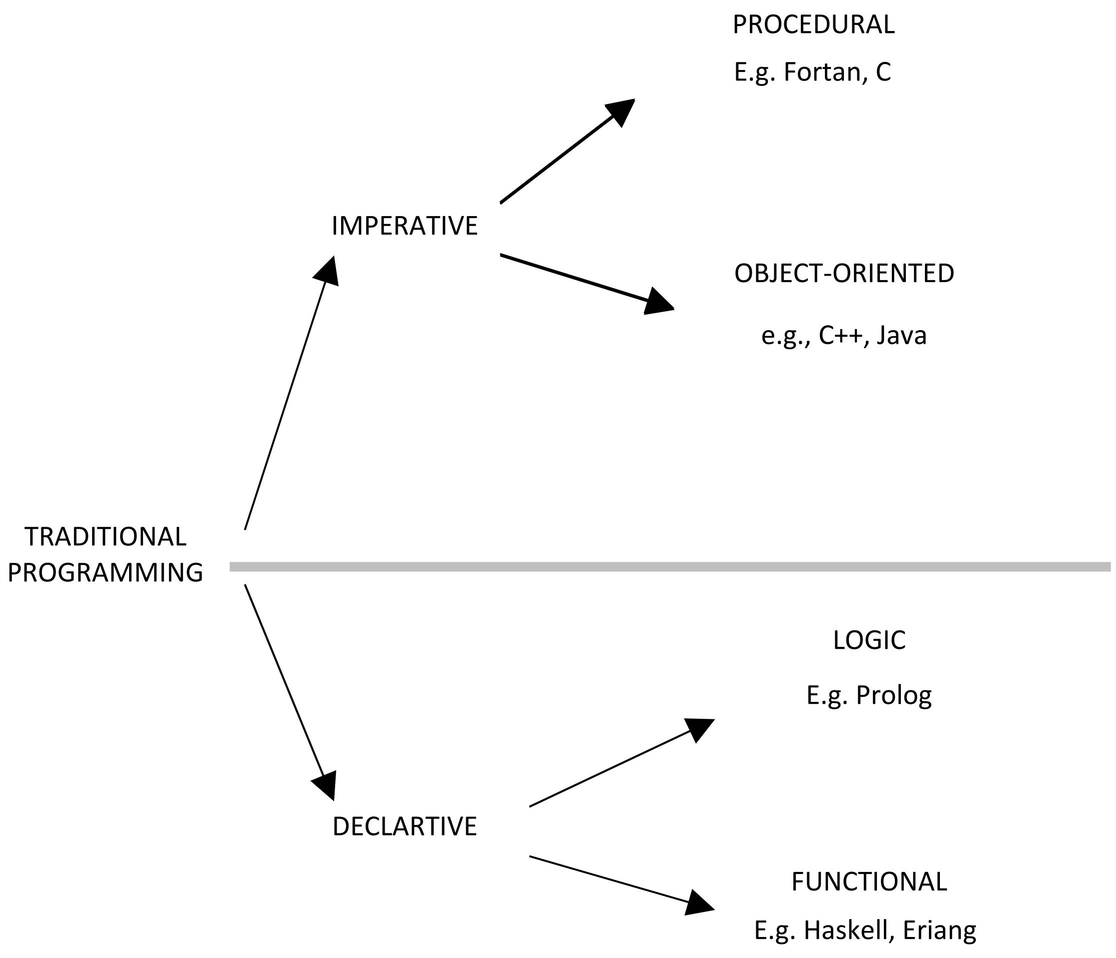

FIGURE 5 Programming Language Paradigm

A paradigm is simply regarded as a mean to specify a computation, selected on the basis of convenience rather than principle." [6]. Although the approach is different but the emphasis in both the approach is to execute the program, specifically designed for machine level development and implementation. At the same time these approaches are deployed with an idea of formalising things so that it can be further used to deduce algorithms and standardised the execution procedures. This in turn reduces the idea of free thought process that is an important part of evolution of human knowledge.

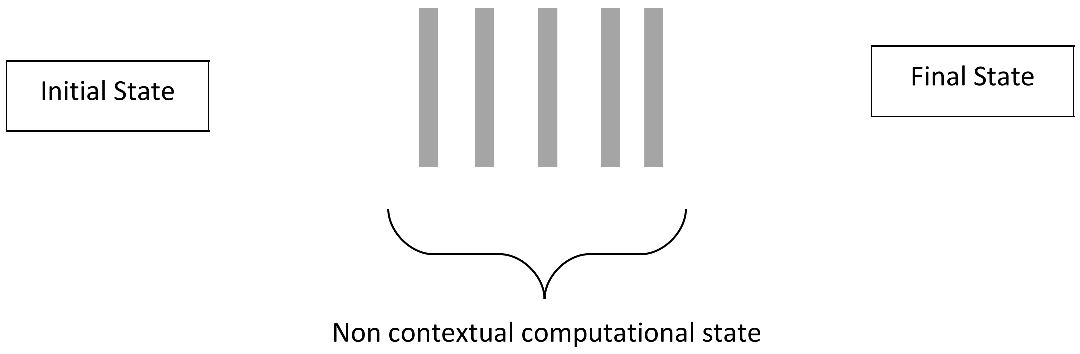

FIGURE 4. Non- contextual Phases in traditional Programming

4.2 Meaning of Interpretation of states in EM

Interpretation of states while developing a model in EM is more leaned towards the human understanding of the state in a given context. Interacting with such artefacts using computer technology enables us to think with computers, and is quite unlike conventional interaction with programs. EM proceeds by elaborating scripts to create interactive artefacts that are works of the imagination, reflecting experience and current understanding but open to many interpretations. [Website]

4.3 "Agent oriented modelling with definitive representation of state"

The main concern in both the above mentioned traditional approaches (i.e. Imperative and Declarative) is that they are focused on the Input / Output operations (i.e., the Initial state and the final state of the program). This deterministic approach towards Input / Output operations makes them inefficient to define the contextual states which are crucial to define the system in terms of human understanding, at any given point in time. With Definitive notation, the focus is on building up the contextual states and defining behaviour of different elements with reference to the context. This method of building up models in reference to the context does not define (promises) any determined final state; rather it's a way of exploring through meaningful states in the process of building up the contextual model.

Traditional Programming	**Definitive Notation**
Deterministic	Explorative
Traditional programming is more concerned with a defining a system in terms of deterministic initial and final computational state	Definitive notation is concerned with building up a model while exploring through every contextual state.

5. SMALL INTRODUCTION TO EMPIRICAL MODELLING

Empirical – The word "Empirical" means experimental or experiential. The idea of using the word "Empirical" in EM denotes the inspiration of taking up practical or real world context. The word "Modelling" has been used to denote the experiential aspect of the EM which tends to define the context visually. The word "Modelling" shall not be compared or confused with the one that is used in context of UML etc. of traditional programming. In other words, EM tries to incorporate the human skills along with the qualities of the technology.

For the purpose of understanding this paper considers the heap-sort model that has already been developed and discussed earlier.

6. DEFINITION OF HEAP-SORT

"The heap data structure is an array object which can be easily visualized as a complete binary tree. There is a one to one correspondence between elements of the array and nodes of the tree. The tree is completely filled on all levels except possibly the lowest, which is filled from the left up to a point. All nodes of heap also satisfy the relation that the key value at each node is at least as large as the value at its children."

For further explanation of general working of heap sort is describe at the link below: http://www.iti.fh-flensburg.de/lang/algorithmen/sortieren/heap/heapen.htm

7. EM PERSPECTIVE OF HEAP-SORT MODEL

In general, the idea behind developing a heap-sort model in traditional programming is to compute and establish the principles of the heap-sort without giving much emphasis on how the underlying things are interacting and evolving. In other words, the system of heap-sort that we develop in traditional programming undergoes a transformational system as described in **(...)** ; It takes array of numbers as input and then builds up the sorted heap without mentioning much about the associated observables. Hence Traditional programming takes an deterministic approach towards building Heap-sort model. However, Heap-sort model in EM is more leaned towards the idea of learning, designing, discovery and reinterpretation. These points will be discussed in subsequent paragraphs along with other concepts.

(*Please note: Most of the concepts elaborated in these paragraphs are directly or indirectly derived from the talk given by Dr. Steve Russ on concept of Construal in EM)*

In heap-sort model, agents at the nodes of the heap can monitor and impose heap conditions. Associating atomicity to agent action is a crucial feature of observation-oriented modelling which in turn gives enough room for open interaction with the model.

The development of heap-sort model in EM includes definitive script that includes all the observables associated with the human interpretation and machine implementation *(Unlike what has been discussed about the traditional programming approach in Section 4.1)*. At present, the concept of deploying an external observer is incoherent and inconsistent. The idea of adding observables in EM provides an objective external perspective within which the interactions of the constituent agents can be rendered coherent and consistent.

Some observables that are needed to describe heapsort include:

- the values in the array

- the locations of the nodes in the tree
- the notion of a node being in the active segment of the array
- the notion of the heap condition being satisfied at a node
- the order relationship between the value at a node and its parent node

All these observables have explicit counterparts in the model that can be inspected at all times. The dependencies perceived to interrelate these observables are expressed by definitions.

Through the EM approach, it is possible to ***discover*** the facts that are involved in conceiving an algorithm such as heapsort. This in turn makes it feasible to develop the model which supports the concept of acquaintance and acceptance as discussed in Section 1.

Another important aspect associated with the heap-sort model development is an ***open input window*** through which definitions can be added to the pre-executing scripts and any definition in the script can be revised. This aspect of EM corresponds to reactive systems as discussed in Section 3.2. With the approach of open output window, one can interact with the eniviournmet and see the effect instantly. This gives a feeling of "using computers in support of experimental activity", which is also one of the important aspects of EM.

Interpretation of heap-sort model has prime importance while developing it in EM. First and foremost, it gives a Visual platform to illustrate the heap-sorting process. Secondly, the model helps to learn the concepts and perceptions that are involved in understanding heap-sort. And finally, as mentioned earlier, it

8. CONCLUSION

Empirical Modelling (EM) provides a new dimension to computational science that incorporates human interpretation with machine implementation. A further advancement in this direction will bridge the gap between an experiential and a formal account of computing.

ACKNOWLEDGEMENT

I would like to thank Dr Steve Russ and Dr W M Beynon, at the University of Warwick, for their support in the preparation of this paper. The help from my fellow classmates in MSc Computer Science and Application (2010 -2011 batch), at the University of Warwick, is also highly appreciated

REFERENCE

[1] A. Pnueli, NY University, Analysis of reactive systems, Fall 2002

[2] Accessed from [http://terpconnect.umd.edu/~austin/ence200.d/software.html] on 26 Dec 2010

[3] Accessed from [http://terpconnect.umd.edu/~austin/ence200.d/software.html] on 26 Dec 2010

[4] W M Beynon , S B Russ, University of Warwick, "The interpretation of states : a new foundation of computation ? "

[5] W M Beynon , University of Warwick, "New path for programming in theory and practice "

[6] W M Beynon , S B Russ, University of Warwick, "The interpretation of states : a new foundation of computation ? "

BIBLIOGRAPHY

[1] B C Smith, Stanford University, "Two lessons of logic", 1987

[2] W M Beynon, University of Warwick, Paradigms for Programming, Unpublished

[3] W M Beynon, University of Warwick, "Programming Principles for the Semantics of Programs

[4] W M Beynon , University of Warwick, "New path for programming in theory and practice "

[5] M Jackson, The open University, "What can we expect from the program verification?"

WEB-RESOURCES

[1] http://www2.warwick.ac.uk/fac/sci/dcs/research/em/intro/

(Official Website of Empirical Modelling, The University of Warwick)

[2] http://terpconnect.umd.edu/~austin/ence200.d/software.html

(official website of Prof. J P Desai)