ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 10| Issue: 6| June 2024|| Journal DOI: 10.36713/epra2013 || SJIF Impact Factor 2024: 8.402 || ISI Value: 1.188

ASSESSING CENTRE STATE RELATIONSHIP IN THE LIGHT OF OBJECT ORIENTED PARADIGM: A STUDY IN SEARCH OF SOCIALLY OPTIMUM

Sugata Sen[1] Santosh Nandi[2]
[1]*Associate Professor of Economics, Panskura Banamali College (Autonomous)*
[2]*Faculty, PG Department of Computer Science and Applications, Panskura Banamali College*

ABSTRACT
State-centre structural duality is one of the main reasons behind the systematic backwardness of the states in India. One of the major mechanisms to address the problem of dual economy is the bottom-up decentralised development planning. But decentralised participatory planning suffers from serious problems related to implementation. Apart from many reasons behind the implementation problem it is the incentive incompatibility which creates serious impediments in the path towards the said implementation. Unless this problem of incentive incompatibility is minimized the idea of decentralised participatory planning will not be able to implement itself successfully. This type of inclusive decentralised planning can help India to become a super power through the path of holistic growth. To wipe out the structural duality it is needed to evolve a method where the problem of incentive incompatibility is minimized along with accommodating the preferences of beneficiaries with in the top down method. Actually, these decentralised planning has failed to deliver desired result due to the existence of centralised components inside the method and incentive incompatibility. This failure is compared here with the non-existence of abstract relationship between the centre and state. Abstract relationship as described here fails to deliver desired outcome when the methods declared at the centre class fail to find proper agent for execution at the derived state class. This mismatch between the method and the agent is described here as the failure of the development program or failure of runtime polymorphism. Thus, this work wants to explain this structural phenomenon through Object Oriented Paradigm. It is concluded here that this backwardness arises due to the non-existence of abstract relationship between the centre and state areas. This impediment can be corrected through development of interface using the concept of successive approximation.

KEYWORDS: *State Development, State–Centre dualism, Object Oriented Paradigm, Abstract behaviour, runtime polymorphism, interface planning, successive approximation.*

INTRODUCTION

It is observed that, in India due to state-centre duality the state areas fail to participate in the development process properly. It is one of the main reasons behind the systematic backwardness of the state areas. This work wants to explain this structural phenomenon through Object Oriented Paradigm (OOP). It is deduced that this backwardness arises due to the non-existence of abstract relationship between the centre and state areas. This impediment can be corrected through the development of interface space with the concept of normative value judgement and multiple regression analysis through successive approximation. The development of these interactive planning techniques will open new vistas of development planning through due importance to localisation.

EXISTING LITERATURE REVIEW

State centre dualism is the existence of two separate economic sectors within one country, divided by different levels of development, technology, and different patterns of demand. The concept was originally created by Julius Herman Boeke (Boeke, 1953) to describe the coexistence of modern and traditional economic sectors in a single economy. Dual economies are common in less developed countries, where one sector is geared to local subsistence needs and another to the modern market-oriented needs. These characters are not homogenous throughout the economy and varies substantially from space to space.

One of the major mechanisms to address the problem of dual economy is the bottom-up decentralised development planning(Pal, 2008). A decentralized-planned economy is a type of economic system based on decentralized economic planning, in which decision-making is distributed amongst various economic agents or localized within different geographical spaces. Decentralized planning is held in contrast to centralized planning where economic information is aggregated and used to formulate a plan for production, investment and resource allocation by a central authority. Recent proposals for decentralized-economic planning have used the term participatory planning to highlight the cooperative and democratic character of this system(Rao, 1989). Proponents present decentralized and participatory economic planning as an alternative to centre centric market-oriented growth.

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 10| Issue: 6| June 2024|| Journal DOI: 10.36713/epra2013 || SJIF Impact Factor 2024: 8.402 || ISI Value: 1.188

But in parallel it has also been observed that the decentralised planning has failed to deliver desired result due to existence of centralised components inside the method and incentive incompatibility (Rao, 1989). A mechanism is called incentive-compatible (IC) if every participant within a set can achieve the best outcome to him/herself just by acting according to his/her true preferences (Ledyard, 1989).

This failure is compared here with the non-existence of abstract relationship(Schildt H. , 2002)between the centre and state. Abstract relationship as described here fails to deliver desired outcome when the methods declared at the centre class fail to find proper agent for execution at the derived state class. This mismatch between the method and the agent is described here as the failure of the development program or failure of runtime polymorphism(Mat Marcus, 2007).

This work wants to explain this structural phenomenon through Object Oriented Paradigm(Lafore, 1999). It is concluded here that the backwardness of the state areas arises due to the non-existence of abstract relationship between the centre and state areas. This impediment can be corrected through development of interface using the concept of successive approximation within interface planning. Interface planning evolves new method through continuous interface between the existing methods and the agents. The child class will implement the evolved method with the help of this interface(Schildt H. , 2017) This interface is facilitated through the process of normative value judgement and multiple regression analysis (Maddala, 2001).

The failure to deliver optimum result by the planning procedure may be compared with the non-existence of abstract behaviour (Schildt H. , 2002)between the centre and state areas. A relation which is declared at abstract class and defined at derived class is called abstract relationship. A class whose objects or instances cannot be created elsewhere is known as abstract class. Here it is considered that centre area is the parent abstract class and state is the child class. Here the method which is declared in abstract and define on the derived child class acts on data according to their access control specifications. With the help of OOP (Lafore, 1999)this relationship can easily be constructed.

The object-oriented paradigm took its shape from the initial concept of a new programming approach, while the interest in design and analysis methods came much later. Object-oriented programming uses objects, but not all of the associated techniques and structures are supported directly in languages that claim to support OOP. The object-oriented design paradigm is the logical step in a progression that has led from a purely procedural approach to an object-based approach and now to the object-oriented approach. The progression has resulted from a gradual shift in point of view in the development process. The procedural design paradigm utilizes functional decomposition to specify the tasks to be completed in order to solve a problem. The object-based approach, typified by the techniques of Yourdon, Jackson and Booch, gives more attention to data specifications than the procedural approach but still utilizes functional decomposition to develop the architecture of a system. Abstract relation and abstract class, Polymorphism, inheritance are three of the main features provided by OOP.

Abstract base classes are useful for creating polymorphic methods. An abstract base class defines an interface without an implementation. Each abstract base class has one or more well-defined derived classes. Derived classes implement the interface defined by the method of abstract base class. Graf and Saidi (Graf & Saidi, 1997) developed a technique for automatically creating a finite-state system (for which a fixed-point analysis will terminate) from an infinite-state (or very large finite-state) system (for which a fixed-point analysis will, in general, not terminate). With abstraction relationship, the concrete states of a system are mapped to abstract states (or classes) according to their valuation under a finite set of predicates. Predicate abstraction has been used to construct abstractions of hardware and protocol designs in the model-checking community. In this work we have also followed the same technique where the polymorphic method failed to find the finite state.

Smith (Smith, 2002)has proposed the use of runtime polymorphism in An Elemental Design Pattern Catalogue. Starting from the initial pattern designing of 16 pattern's, they are broken down into three main groupings: Object Elements, Type Relation, and Method Invocation. In our proposed work these patterns can be treated as polynomial. These patterns, precisely important in possible formalization. Here the object elements can be treated as data of our defined class and type relation is treated as methods, invocation of method can be treated as technique as polymorphism.

Tandon(Runjhun Tandon, 2016)has applied polymorphism in pharmaceutical compounds and their intermediates. It is very important and an integral part of drug development. He has tried to explain the isomerism properties of different drugs through polymorphic methods. As a corollary of this work we can also formulate different technique and methods through the polymorphic technique as has been used by this cited work.

Abrahams (Abrahams, 2003)has used Runtime polymorphism to Build Hybrid Systems with Boost. Python. Boost. Python is an open source C++ library which provides a concise IDL-like interface for binding C++ classes and functions to Python. The author developed meta programming techniques which is achieved entirely in pure object-oriented programming without introducing a new syntax. Like our work we can treat it as a technique of polymorphism such that in different cases we can construct different polynomials.

These polymorphic methods can fail due to the mismatch of data and method. this mismatch appears due to the wrong approximation of the child class by the abstract class. This failure can be treated as the failure of alternative development plans or abstract relationship. To rectify the problem of broken abstract relationship we introduce here a new concept of interface planning. Where the existing methods will interface

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 10| Issue: 6| June 2024|| Journal DOI: 10.36713/epra2013 || SJIF Impact Factor 2024: 8.402 || ISI Value: 1.188

with the agents and the new methods would be evolved. The existing methods would be improved through the influence of the agents of the child class. Here the child agents will influence the original methods and new methods will be evolved. This new method is nothing but a new technique represented by a new polynomial. Now the child class will implement that the evolved method with the help of this interface instead of inheritance(Lafore, 1999).

An interface contains definitions for a group of related functionalities that a class or a structure can implement. When the primitive goal is to include some behaviour from different sources, interface comes into action. In C# there is no possibility to introduce multiple inheritance. Here interface plays a important role. Any class or structure that implements the interface must contain a definition for the method that matches the signature that the interface specifies. When a class or structure implements an interface, the class or structure must provide an implementation for all of the members that the interface defines. The interface itself provides no functionality that a class or structure can inherit in the way that it can inherit base class functionality. However, if a base class implements an interface, any class that's derived from the base class inherits that implementation(Schildt H. , 2002).

In Visual Basic *Interfaces* define the properties, methods, and events that classes can implement. Interfaces allow to define features as small groups of closely related properties, methods, and events; this reduces compatibility problems because the programmer can develop enhanced implementations for interfaces according to need without jeopardizing existing code. One can also add new features by developing additional interfaces and implementations if needed(Jerke, 1999).

An interface in the Java programming language is an abstract type that is used to specify a behaviour that classes must implement. They are similar to protocols. Interfaces cannot be instantiated, but rather are implemented. A class that implements an interface must implement all of the non-default methods described in the interface, or be an abstract class. Object references in Java may be specified to be of an interface type; in each case, they must either be null, or be bound to an object that implements the interface. Implementing an interface allows a class to become more formal about the behaviour it promises to provide. Interfaces form a contract between the class and the outside world, and this contract is enforced at build time by the compiler. If your class claims to implement an interface, all methods defined by that interface must appear in its source code before the class will successfully compile(Sarang, 2012).

In object-oriented paradigm, the term interface is often used to define an abstract type that contains no data or code, but defines behaviours as method signatures. A class having code and data for all the methods corresponding to that interface is said to implement that interface. Furthermore, a class can implement multiple interfaces, and hence can be of different types at the same time.

The creation of new technique through interface can follow the process of successive approximation(Skinner, 1953). Let us use the definition of “shaping” to explain successive approximations. Our definition of "shaping" is: "a behavioural term that refers to gradually moulding or training a method or technique to perform a specific response by reinforcing any responses that come close to the desired response. Successive approximation is used for determining the range where the solution can possible exists instead of finding the exact solution.

Finally, the statistical significance (Wooldridge, 2013) of the developed model will be tested over a sample of 320 households drawn through multi-stage stratified random sampling without replacement(Das, 2010).

CONTEXT

It appears from the above discussion that the poverty level and the development status of the state areas in is significantly different from that of the centre areas. The main reason behind this divergence can be attributed to the idea of state-centre dualism. Till date different forms of development planning have tried to address this dualism though all the efforts have culminated to sub-optimal output. One of the major components of these development plans is the participatory decentralised planning. But due to the existence of incentive incompatibility in the benefit delivery process the final receivers fail to participate in the bottom-up information gathering system and the planning execution fails to deliver the optimum output. Here the relationship between the planner set and the receiver set can well be documented through abstract relationship as developed by Object Oriented Paradigm. It can be said that the planning methods declared at the abstract class are defined at the derived state class through the techniques of runtime polymorphism. Thus the state centre dualism is nothing but the non-existence of this abstract relationship and the failure of state development plans are failure of polymorphic methods. In this perspective this work wants to develop a new method of development planning called ‘interface planning’. Under this conceptual procedure an interface space would be created where the existing plans or methods will interface with the beneficiary agents. This interface instead of inheritance between the methods and the agents will develop new planning techniques or methods by incorporating the actual characteristics of the agents. To that interface successive approximation can deliver desired support to construct the optimum method. Here numerical methods of differential equation can felicitate this successive approximation. Such that the specific objective or hypothesis of this study is as follows.

HYPOTHESIS

State-centre dualism exists due to the failure of abstract relationship between the centre and state areas. Successful development planning procedures can be developed on interface space through successive approximation.

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 10| Issue: 6| June 2024|| Journal DOI: 10.36713/epra2013 || SJIF Impact Factor 2024: 8.402 || ISI Value: 1.188

INTERFACE MODEL

It is accepted that centre and state areas are two distinctly different spaces with respect to their development status. Let us assume that the state class has been inherited from the centre class following the idea of Core-periphery Hypothesis. Such that the centre class can be denoted as the abstract class and the state class as the derived class or the child class. A class whose objects or instances cannot be created elsewhere is known as abstract class. This abstract class consists of method as well as data whereas the derived class only consists of data or agents. The methods are declared at the abstract class and defined at the state class. A relation which is declared at abstract class and defined at derived class is called abstract relationship. Here the method which is declared in abstract and define on the derived child class acts on data or agents of the derived class according to their access control specifications. With the help of Object Oriented Paradigm this relationship can easily be constructed(Lafore, 1999).

Here the centre class is denoted by UC and child class is denoted by RC and the methods are denoted by M_i and the data are denoted by d_j. Here i and j both lies from 1 to n. i,e $1 \leq i \leq n$ and $1 \leq j \leq n$. Here the polynomial is denoted by PM and polymorphism is denoted by PP. So we can write

$UC=\{M_i,d_j\}$, . i,e $1 \leq i \leq n$ and $1 \leq j \leq n$. M_i X D Ↄ R. where R is the relation between method set and data set.
i,e the Cartesian product between methods set and data set.

RC=I(UC) ie,I:UC→RC . her I is a function which maps RC to UC.

$PM=PP(d_i)$, where i=1 to n and PP: $d_i \rightarrow PM$

As the polymorphic operations act on the data of the child class, the child class can be considered as a finite group. Let us assume RC as a group and the operations are PP. So RC is as a finite group which can be denoted as (RC,PM).

This very relationship can be treated as development plan. The type of this linkage is determined centrally and executed at state areas. The abstract relationships as describe here or the development plans may fail to deliver desired outcome when the methods fail to find proper agent for execution. This mismatch between the method and the agent is nothing but the failure of the development programs. So the construction of the abstract relationship is very important for delivery of optimum desired output. Any breach of abstract relationship will make the planning process void. The declared planning methods are executed through runtime polymorphic methods. Polymorphism (Schildt H. , 2002) means ability to make more than one form. Actually it means that same function performs different operations on different circumstances (considering different parameters). Under runtime polymorphism (Mat Marcus, 2007) the binding between data and function is occurred at the run time, that is at the execution time. In this situation the code of a particular method to be executed is not decided at the time of function call, it is decided only at the run time with respect to call of that particular method or function.

It may happen that in some scenario all the polymorphic methods declared, may not work at all or fail. Failure of each of the polymorphic method is nothing but the failure of abstract relationship.

Let us consider f(x) as a development method declared at the centre class. Let us assume that this method is defined as randomised polynomial at the derived class as $\sum a_i(x_j)^n$where $0 \leq i \leq n$ and $0 \leq j \leq n$.Here a_i(where i=1,2,3,…….n) are coefficient and x_j (where j=1,2,3…n) are parameters. Through polymorphic methods we can form more than one polynomial and each polynomial can be treated as single method. Let us assume that f_j(where j=1,2,3..n) are different defined methods and they are defined as $f(x_i)=fj$. Here i=1,2,3…n and j=1,2,3, …….n.

The breaking of the abstract relationship or the proper interface between the agents will create dynamic loop of backwardness. This vicious circle of backwardness which arises out of improper abstract relationship can only be rectified through due information about the child class to the parent class. Again the incentive incompatibility of the derived class can also be minimized through acquiring proper development requirements of the agents.

To rectify the problem of broken abstract relationship we introduce here a new concept of interface planning. Where the existing methods will interface with the agents and the new methods would be evolved. The existing methods would be improved through the influence of the agents of the child class. Here the child agents will influence the original methods and new methods will be evolved. This new method is nothing but a new technique represented by a new polynomial. Now the child class will implement the evolved method with the help of this interface (Schildt H. , 2017) instead of inheritance (Lafore, 1999). Due to the existence of interface set the problem of incentive incompatibility will disappear or be minimised to a greater extent.

Let us conceder IN as an interface which contain only data and abstract method. Where the method and data are already denoted so IN can be considered as a set of method and data ie, $IN=\{M_i,d_j\}$, $M_i(d_j)$ is not valid i.e. $M_i \rightarrow d_j$ does not exist. Let us consider that R is a relation which is define as R⊄ M_i X D. where D is the data set which is defined as $D=\{d_j\}$ where $1 \leq j \leq n$.

The creation of new technique through interface can follow the process of normative value judgement through successive approximation and multiple regression analysis (Maddala, 2001). Let us use the definition of “shaping” (Skinner, 1953)to explain successive approximations. Here the definition of "shaping" is: "a behavioural term that refers to gradually moulding or training a method or technique to perform a specific response by reinforcing any responses that come close to the desired response. Successive approximation is used for determining the range where the solution can possible exists instead of finding the exact solution.

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 10| Issue: 6| June 2024|| Journal DOI: 10.36713/epra2013 || SJIF Impact Factor 2024: 8.402 || ISI Value: 1.188

The essence of this modelling is that the state areas can be pulled at the status of the centre areas when plans are formulated through continuous interface between the development plans and development outcomes. This interface planning method is unique in the discourses of development planning and will open new vistas of development planning ideas as well as methods through the application of computational techniques.

FINDINGS

The model delivered above by OOP here is finally tested over a set of 320 households. It is observed that bottom up approach of decentralised planning can be executed successfully in its true sense by using OOP through minimising incentive incompatibility and excluding centralised components.

CONCLUSION

This work wants to develop a model where the decentralised plan can be executed successfully. So far, this decentralised plan is failure due to the presence of abstract relationship which exist between State child class and the Centre parent class. Actually, this abstract relationship is failure due to improper formulation of polymorphic method. These polymorphic methods are actually nothing but the plans which are sent by the abstract Centre class to the State child class. For this purpose, a detailed algebraic model has been developed with the help of successive approximation and normative value judgement. The whole model is implemented through object oriented paradigm in an interface area. In this model the beneficiary agents are came from State child class and plans will come from the Centre parent class and they interact with each other and make the proper method with respect to beneficiary requirement. To implement the model and to test the outcomes of the algebraic model primary level data is collected through door-to-door questionnaire-based survey.

REFERENCES

1. *Abrahams, D. (2003). Building Hybrid Systems with Boost.Python.*
2. *Blunch, N.-H., & Verner, D. (1999). Sector growth and the dual economy model - evidence from Cote d'Ivoire, Ghana, and Zimbabwe. World Bank. Retrieved DEC 08, 2017, from http://documents.worldbank.org/curated/en/343991468749746215/Sector-growth-and-the-dual-economy-model-evidence-from-Cote-dIvoire-Ghana-and-Zimbabwe*
3. *Boeke, J. H. (1953). Economics and Economic Policy of Dual Societes as Exemplefied by Indonesia. New York: Ams Press Inc.*
4. *Das, N. G. (2010). Statistical Methods. New Delhi: Tata McGraw-Hills.*
5. *Egenhofer, M. J., & Frank, A. U. (1992). Object-Oriented Modeling for GIS. URISA Journal 4, 3-19.*
6. *Graf, S., & Saidi, H. (1997). Construction of Abstraction with PVS.IN CAV 97: Computer aided verified. Lecture Notes on Computer Science, 1254, 72-83.*
7. *Hitz, M., & Montazeri, B. (1995). Measuring Coupling and Cohesion In Object-Oriented Systems. 25-27.*
8. *Iserles, A. (1996). A First Course in the Numerical Analysis of Differential Equations. Cambridge: Cambridge University Press.*
9. *Jerke, N. (1999). Visual Basic 6: The Complete Reference. New Delhi: Tata McGraw-Hill Education.*
10. *Lafore, R. (1999). Object Oriented Programming in C++. New Delhi: Galgotia Publications Pvt. Ltd.*
11. *Ledyard, J. O. (1989). Incentive Compatibility. In J. Eatwell, M. Milgate, & P. Newman, Allocation, Information and Markets (pp. 141 - 151). London: The New Palgrave Macmillan.*
12. *Lee, D. G., & Kafura, H. K. (1989). Inheritance in Actor Based Concurrent Object-Oriented Languages. The Computer Journal, 297-304.*
13. *Lewis, J. A., Henry, S. M., Kafura, D. G., & Schulman, R. S. (1991). An empirical study of the object-oriented paradigm and software reuse. 91 Conference proceedings on Object-oriented programming systems, languages, and applications (pp. 184-196). New York: ACM SIGPLAN.*
14. *Lieberherr, Karl, Holland, I., & Riel, A. (1988). Object-oriented programming: An objective sense of style. ACM SIGPLAN Notices 23, 232-234.*
15. *Maddala, G. S. (2001). Introduction to Econometrics. New York: John Wiley & Sons, Ltd.*
16. *Mat Marcus, J. J. (2007). Runtime Polymorphic Generic Programming – Mixing Objects and Concepts in ConceptC++. GPCE '07 Proceedings of the 6th international conference on Generative programming and component engineering , (pp. 73-82). New York.*
17. *Pal, M. (2008). Decentralised Planning and Development in India (1st ed.). New Delhi: Mittal Publications.*
18. *Rao, C. H. (1989, February 29). Decentralised Planning: An overview of Experience and Prospects. Economic and Political Weekly.*
19. *Runjhun Tandon, N. T. (2016). Effect of seeding on sucrose polymorphism. Journal of Chemical and Pharmaceutical Research, 531-546.*
20. *Sarang, P. C. (2012). Java 7 Programming . London: Oracle Press.*
21. *Schildt, H. (2002). The Complete Reference C++. New York: McGraw-Hill Education.*
22. *Schildt, H. (2017). Java: The Complete Reference. New York: McGraw-Hill Education.*
23. *Siek, J. G. (1995). Gradual Typing for Functional Languages. Journal of functional programming, 111-130.*
24. *Skinner, B. F. (1953). Science and Human Behaviour. New York: Macmillan.*
25. *Smith, J. M. (2002). An Elemental Design Pattern Catalog.*
26. *Wooldridge, J. M. (2013). Intoductory Econometrics: A Modern Approach. Mason: South-Western Cengage Learning. Retrieved from http://economics.ut.ac.ir/documents/3030266/14100645/Jeffrey_M._Wooldridge_Introductory_Econometrics_A_Modern_Approach__2012.pdf*