ABSTRACT. In this paper object-oriented information system (OOIS) is defined as an information system which employs object-oriented technologies in system design and implementation. This paper attempts to clarify the concept of OOIS and its implication, and to summarise research and practices in the OOIS area. At the top level, perspectives on foundations of object-orientation, information, information systems and OOIS are presented systematically. Then the domain of OOIS is analysed and a generic structure of OOIS as a branch of computer science is derived. Finally trends in OOIS technologies are analysed based on the review of OOIS’94-97 proceedings.

KEY WORDS: Information systems, object orientation, foundations, problem domain, generic structure, trends.

1. Introduction
In tracing the history of programming and system design methodologies, it can be seen that functional decomposition had been adopted in programming since the 1950s [MCD91]. In the 1970s the most significant progress in programming methodologies were structured programming [HOA72] and abstract data types (ADTs) [LIS74]. Since the 1980s object-oriented programming (OOP) [STR82/86/87, GOL83, SNY87] have been adopted in almost every software engineering branch.

Because object-orientation methodologies have taken the best ideas in structured programming and ADTs, and combined them with several new mechanisms such as encapsulation, inheritance, reusability and polymorphism, their applications have been naturally extended to information systems design and implementation in the 1990s. This progress has formed a new area of research named object-oriented information systems (OOISs).

This paper attempts to clarify the implication of OOIS, and to summarise research and practices in the OOIS area. At the top level, perspectives on foundations of object-orientation, information, information systems and OOIS are discussed systematically. Then the domain of OOIS is analysed and a generic structure of OOIS as a branch of computer science is derived. Finally trends in OOIS technologies are analysed based on the review of OOIS’94-97 proceedings [WAN98d].

2. Fundamentals of object-orientation

In computer science and software engineering, a set of fundamental conceptual tools have been developed to cope with the complexity of problem specification and solution. Some of the important methodologies are abstraction, information hiding, functional decomposition, modularization and reusability. Object-orientation (OO) technologies have inherited the merits of these fundamental approaches and represented them in well organised mechanisms such as encapsulation, inheritance, reusability and polymorphism.

Although OO is one of the broadly used concepts in software engineering and information systems, the literature presents few clear definition of OO, rather it is generally assumed that OO is a well known concept. This section tries to trace the history of object-orientation, describe the implication of object and OO, analyse the extension of the concept and categorise the technologies for OO.

2.1 What is object-orientation?

To enable the question of what is object and object-orientation to be answered, one needs to address the following issues: what is the implication of OO? what is the intention and extension of OO? Tracing back the history of programming methodologies, it can be concluded that object-orientation is a natural extension and combination of two main stream programming methodologies: the functional-
oriented programming and the data-oriented programming. Therefore, the definition of object-orientation would be based on the concept of object in programming and system design. A definition of object is given below:

**Definition 1.** Object is an abstract model of a real world entity and/or a computational module which is packaged by an integrated structure of interface and implementation, and is described by methods for its functions and by data structures (DSs) for its attributes.

In object-oriented programming languages, such as SmallTalk [GOL83] and C++ [STR86], objects can be very small like a user-defined record or fairly large like a whole hierarchy of classes. A generic structure of an object can be described in Fig.1.

![Fig.1. A generic structure of an object](image)

For a well packaged object, the only access means to it is via the interface of the object. The implementation of methods and related data structures is hidden inside the object, which enables the methods could be changed independently without affecting the interface of the object.

Object-oriented technologies were originally designed for programming. Therefore OO was initially an implementation tool rather than a design tool. However, as OO programming became broadly accepted, it was found that OO technologies could be used not only in programming, but also in system design and analysis. OO technologies are fairly generic and are applicable in almost every phase of the software development lifecycle. Based on this view, object-orientation is defined as below:

**Definition 2.** Object orientation is a kind of system design and/or implementation methodology which supports integrated functional-and-data-oriented programming and system development.
2.2 Basic attributes of object-orientation

The fundamental attributes that can be commonly identified in object-oriented technologies are encapsulation, inheritance, reusability and polymorphism. A set of formal descriptions of these basic attributes is given as follows:

**Definition 3.** Encapsulation is a basic attribute of object-oriented technologies by which the functions and data structures of an object is integrated into a package, and the object can only be accessed by calling its methods with specific messages.

**Definition 4.** Inheritance is a basic attribute of object-oriented technologies by which methods and related data structures modelled in an object can be inherited by derived objects as existing system functions and date types.

**Definition 5.** Reusability is a basic attribute of object-oriented technologies by which objects, classes and class hierarchy modelled in an object-oriented system can be reused by different applications as existing system resources.

**Definition 6.** Polymorphism is a basic attribute of object-oriented technologies to provide evolvability and tailorability for inheritance by which the inherited methods and related data structures of an object can be partially redefined or overloaded.

Within this set of basic attributes, encapsulation is a direct representation of the fundamental methodologies of abstraction, information hiding and modularization in objects. Inheritance and reusability are powerful features for improving productivity and quality in software and system development. Polymorphism is a supplement of flexibility to the other attributes of OO.

2.3 Classification of object-oriented technologies

In viewing OO technologies as generic system analysis, design, implementation and reengineering methodologies, a classification of existing OO methodologies can be presented as shown in Table 1.

According to the above classification, when an OO system is referred to, it implies that the system is designed, implemented and/or reengineered by OO technologies. This will be useful in defining what is an OO information system in the following section.

**Table 1. Classification of object-oriented methodologies**

<table>
<thead>
<tr>
<th>OO category</th>
<th>OO methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td>OO design / analysis</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OO requirement analysis</td>
</tr>
</tbody>
</table>
3. Foundations of object-oriented information systems

In Section 2 the implication of the concepts of object and object-orientation has been described in a historical view. Before formally defining the terminology of object-oriented information system (OOIS) in this section, implications of information and information systems and their philosophical foundations are investigated.

3.1 What is an information system?

It is commonly regarded that the classical information theory was founded by Shannon during 1948-1949 [SHA48/51/59, SHA49a/b/c], whilst the term ‘information’ was first adopted in Bell and Goldman’s work in 1953 [BEL53, GOL53]. According to Shannon’s information theory, information is defined as a probabilistic measure of the quantity of message that can be obtained from a message source. The classical information theory was flavoured by communications, signals analysis and coding theories.

In the domain of modern applied computer science and in the software and information industries, the term information has been referred to having a much more practical and concrete meaning that focuses on data and message presentation, storage and processing. With this orientation, information is regarded as an entity of messages, rather than a measurement or metric of messages as in the classical information theory. In this perspective on information, a definition can be derived below:

**Definition 7**: Information is a set of organised data that represents messages, knowledge and/or abstract real-world entities.

The storage forms of information are by means of data types, such as numeric, text, hypertext, audio, video or combined multimedia.
With above definition of information, an information system can be described as a system, which manipulates information as shown below:

**Definition 8:** An information system is a computer-based system for collecting, storing, processing (adding, deleting and updating), producing, presenting, searching and/or retrieving information.

What is represented in Definition 8 is a broad implication towards the concept of an information system. For instance, with this definition, an advanced word processing system in a PC can be regarded as a typical personal information system.

### 3.2 Philosophical foundations of information systems

Information system engineering is a unique discipline that relies on special philosophical foundations at the top level [WAN98e]. By contrasting the nature of information system with other science and engineering branches, the authors have found a number of interesting fundamental differences as described below.

#### 3.2.1 Information vs. matter

The knowledge of human being about the world can be categorised into two systems: the concrete and the abstract worlds. The former is formed with matter or natural entities; and the latter is represented by information – the human abstraction of the real world.

Based on this view, it can be considered that our world exists in two basic forms: information and matter. Therefore information science and technology are fundamental branches of science in the human knowledge structure which study theories and methodologies of information processing.

#### 3.2.2 Accumulability of information vs. conservation of matter

According to the natural law of conservation, matter can neither be reproduced nor destroyed. However, in contrast to the properties of matter, important attributes of information are that it can be reproduced, destroyed and accumulated. The accumulability of information is the most significant attribute of information which human beings relay on in evolution.

#### 3.2.3 Virtualization vs. realization

In conventional manufacturing engineering, the common approach moves from abstract to concrete and the final product is the physical realization of an abstract design. However, in information system engineering, the approach is reversed. It moves from concrete to abstract. The final software, database and knowledge-base are the virtualization (coding) and invisibility of an original design which describes
the real world problems. The only tangible part of an information system is the storage media or its run-time behaviours. As illustrated in Fig.2, this is a unique feature of information system engineering.

![Fig.2 Difference between information system engineering and other manufacturing engineering](image)

3.2.4 Infinite problem domain vs. limited problem domain

The problem domain of information systems encompasses almost all domains in the real world. It is infinite and broadly applicable compared with the specific and limited problem domains of the other engineering disciplines. This feature implies the natural complexity of information systems design and implementation.

3.3 Object-oriented information systems

Based on the discussion of implication of information systems in Section 3.1, the analysis of the nature of information in Section 3.2, and the formal description of object and object-orientation in Section 2, an OOIS can now be defined as follows:

**Definition 9:** An OOIS is an information system which employs object-oriented technologies in system design and implementation.

This definition indicates that an information system can be classified as an OOIS if its analysis, design, implementation and testing adopt object-oriented technologies. Definition 9 also shows that reengineering of legacy information systems into object-orientation can be achieved by means of both OO design and implementation technologies.

4. Problem domain of OOIS

In this section, by reviewing the research reported in OOIS’94-97, entities and problem domain of OOIS are identified. Based on this, a generic structure of OOISs is derived systematically.

**4.1 Domain of OOIS**
Generally, an information system contains a wide range of entities, such as hardware, software, people, organisational infrastructure, networking, communications, processes, incoming data, outgoing data, and other resources.

A review of the subject areas in the past events of OOIS’94-97 [PAT95/97, MUR96, ORL98], as shown in Table 2, is helpful to identify the domain of OOISs. The values in Table 2 show the numbers of papers addressed on specific subject areas.

Corresponding to Table 2, a distribution of the OOIS topic areas is derived as shown in Fig. 3. Analysing Fig.3, it can be found that the areas of increasing interest in OOIS included OO methodologies, OO reusability, applications of OO methods, OO software engineering, and OO Web and hypermedia. The areas of declining interest included OO modelling, OO environment/tools, OO programming languages, OO formal methods, OO metrics, and OO knowledge-bases. However the declination of research in certain areas is by no means to show that those areas were no longer important in OOIS.

Table 2. Review of subject areas in OOIS’94-97

<table>
<thead>
<tr>
<th>No.</th>
<th>Category</th>
<th>Subject area</th>
<th>OOIS’94 (London)</th>
<th>OOIS’95 (Dublin)</th>
<th>OOIS’96 (London)</th>
<th>OOIS’97 (Brisbane)</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OO design and analysis</td>
<td></td>
<td>20</td>
<td>20</td>
<td>16</td>
<td>22</td>
<td>78</td>
</tr>
<tr>
<td>1.1</td>
<td>OO methodologies</td>
<td></td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>20</td>
</tr>
</tbody>
</table>
A relative distribution of the topic areas is shown in Figs. 4 and 5 respectively. On the aspect of OOE design, as shown in Fig. 4, research has mainly concentrated on OO modelling, followed by OO methodologies and OO environment/tools.
Fig.4. Distribution of topic areas on OOIS design and analysis

On the aspect of OOIS applications as shown in Fig.5, research and practices were relatively focused on OO DBMS, applications of OO methods, and OO software engineering. A potential trend in application of OOIS is in the area of the Internet and the World Wide Web.

Fig.5. Distribution of topic areas on OOIS applications

4.2 Generic structure of OOIS

With the fundamental studies on OOIS and the analysis of their domain coverage, a generic structure of the OOIS knowledge hierarchy can be derived in Fig.6.

Generally, with respect to design and analysis of OOIS, research has been relatively concentrated on OO modelling. With regard to applications of OOIS, practices have been dominantly focused on OO database systems. That is why some authors perceived that OOIS, in its narrow meaning, is equivalent to OO databases. However, in this paper, the authors intend to explore the generic implication of the OOIS concept as a branch of computer science. This has been formally addressed in Sections 2 and 3 especially by Definitions 8 and 9, and is summarised practically in this subsection.
Fig.6. A generic structure of OOIS

Based on Fig.6 and with the fundamental studies in Sections 2 and 3, it can be asserted that OOIS has formed an important branch in computer science with sound theoretical foundations and a wide range of applications.

Comparing the generic structure of OOIS described in Fig.6 with Table 2, it is clear that interests and work in OOIS have mainly focused on how to develop new systems. An increasingly important aspect on reengineering of legacy information systems by OO has relatively not been covered. Thus the authors suggest that research on methodologies, processes and case studies towards OO reengineering of a large number of legacy information systems would be a worthy area to explore. This will be discussed in details in the following section.

5. Progresses and trends in OOIS technologies

This section describes progresses and trends in OOIS technologies based on the review of OOIS’94-97 and the work of the authors. Main topics on OO reengineering of legacy information systems, integration of existing OO methodologies, development of formal OO methodologies, temporal OOIS technologies, development of OOIS tools, and built-in tests in OOIS will be addressed.

5.1 OO reengineering of legacy information systems

Reengineering of the legacy information systems using object-oriented technologies has been recognised as an important trend in recent years [WIR90, GLE97, SCH97, TRA97, WAN97a]. Migration from procedural implementations to
object-oriented solutions of information systems is considered vital for improving system quality, reusability and productivity. Two categories of problems have been identified for reengineering legacy information systems to object orientation: technical approaches and organisation methodologies. The former can be implemented by the work on OO patterns and frameworks [FAY97, KOS95, LUN96, SPA96, WEI88]; the latter has been addressed by a set of empirical processes of object-oriented reengineering, based on the summarisation of best practices in the software and information industry [WAN96, WAN97b/c, WAN98a].

The work in [WAN96, WAN97b/c, WAN98a] reports organisational and technical processes in object-oriented reengineering of legacy information systems. The processes for object-oriented reengineering of legacy systems have been modelled in a software process reference model (SPRM) [WAN96, WAN97b/c]. A series of worldwide surveys of base process activities (BPAs) towards software process excellence [WAN98a] have been carried out to support the process model.

Findings and benchmarks of a subset of the survey on object-oriented reengineering processes for legacy systems are presented. Detailed and quantitative evaluation on 63 BPAs in seven object-oriented reengineering processes is provided. For each BPA, a set of practical attributes, such as the mean weighted importance in process and the ratios of significance, practice and effectiveness, are derived and benchmarked.

By comparing an organisation’s current practices in information system development with the best practices and benchmarks, recommendations can be given on which specific areas need to have processes established first, and which areas should be highest priority for process improvement in organising an object-oriented reengineering project of legacy information systems.

5.2 Integration of existing OO methodologies

Coad and Yourdon’s OOA is a useful method for analysing the static properties of the problem domain but has insufficient focus on dynamic properties. On the other hand, Jackson’s JSD, which uses event patterns, is useful for modelling the dynamic properties but not so good at modelling static properties. A combination of the OOA static properties and the JSD dynamic advantages may provide a comprehensive analysis method for OOIS design [MAT94].

It is commented in [BER95] that object orientation is a very powerful and useful tool at the technological level of OOIS development but not so successful at the conceptual level. Object-orientation principles have not provided the expected benefits at the conceptual level of OOIS development, especially at the crucial business requirement analysis stage. This is due to the fuzziness of the real-world objects. The most appropriate areas where reuse by inheritance and polymorphism can occur were identified at the programming stage in OOIS development.
5.3 Development of formal OO methodologies

A description is provided in [HIG95] for a formal representation of object-oriented query optimisation, which is considered important to the development of optimisation strategies. The work presents a formalised algebra, an optimisation method and an implementation converter, which are suitable for DAPLEX.

A new comprehensive notation for specifying object-oriented systems is developed in [MAU94]. Its representative diagrams are different from those of existing ones in the ways that they are better expressed and there is less chance for implicit assumptions. Examples of some of the modelling constructs including those of representing types, functions, preconditions, axioms, explicit multiplicity, reflection, views and temporality.

5.4 Temporal OOIS technologies

A temporal object model designed as a generic collection of classes or chronicle classes based on ordered collection of time points has been developed [SCH95]. Problems of expressing temporal queries have also been analysed.

A new distributed systems programming model based on events, constraints and objects is presented in [STA95]. Concurrency, synchronisation, and timing properties are expressed in a unified way using constraints, which may be associated with objects and events in temporal OOIS design.

5.5 Development of OOIS tools

A new intelligent and methodology-independent CASE tool is developed in [MEH94] to support information systems development. This customisable CASE tool supports the building of integrated information systems, spans the entire object-oriented lifecycle, utilises advanced data presentation techniques and exhibits intelligent behaviour.

A method to transform a relational schema into an equivalent object-oriented one is presented in [SOU94]. Using a common data model, a relational schema can be translated in three stages: deduction of the basic types, deduction of edges and functions, and deduction of fragments.

5.6 Built-in tests in OOIS

Built-in tests (BITs) [WAN97d, WAN98b/c] into objects and OO systems are a new philosophy contributing towards OO methodologies. BIT is defined as a new kind of software testing which is explicitly described in OO software and system source code as special methods. Software with BITs has two modes of operating:
the normal mode for program execution and the test mode for debugging, testing and maintenance. The BITs are inactive in the normal mode and can be activated in the test/maintenance mode.

By extending the inheritable structure and the reusability of OO system from code to tests via the BIT methods, the capability of object-oriented mechanisms, such as the encapsulation, inheritance and reusability, can be enhanced to the maximum. OO supplemented by BITs is useful for creating test reusable, easily maintainable and robust code in an object-oriented environment.

In [WAN97d, WAN98b/c] the concept of BITs is introduced. The standard structures, which incorporate the BITs into conventional OO software, are analysed. Reuse methodologies for BITs in OO software are developed at object and system levels. A case study is provided for showing how to create BIT and how to inherit and reuse the BITs in OO design and programming. The applications of BITs in test-reusable and easy maintainable OO systems development, control of dynamic inconsistency of OO software, and OO reengineering of the enormous legacy software systems have been investigated.

6. Conclusions

This paper has reported on basic research in seeking the foundations of OOIS. Fundamental concepts of object, information, information system, object-orientation, OOIS and their relationship have been formally described. A generic structure of OOIS domain has been derived. Based on a review of OOIS’94-97, progresses and trends in OOIS research and development have been analysed.

It has been shown that information system is an important branch of computer science, and that OO technologies are a universal and powerful means not only for improving the design and implementation of new OOIS, but also for reengineering of huge number of legacy information systems.

Trends of OOIS technologies in reengineering, integration of existing OO methodologies, development of formal OO methods, temporal OOIS design, development of OOIS tools, and built-in tests in OOIS have been identified for future research.

Acknowledgements

The authors would like to thank the OOIS’94-98 editors and authors for their inspiration to this work. We would like to thank the valuable comments of the referees.

References


[STA95] Starovic, G., Cahill, V. and Tangney, B. [1995], An Event Based Model for Distributed Programming, in *OOIS’95*, pp.72-86.


Yingxu Wang is a professor in computer science and senior research fellow with the Centre for Software Engineering at IVF, Sweden. He received a PhD in software engineering from the Nottingham Trent University / Southampton Institute, UK. He was a visiting professor in the Computing Laboratory at Oxford University during 1995. He is a member of IEEE TCSE/SESC, ACM, BCS, and ISO/IEC JTC1/SC7. He is an editor of ASQ Journal of Software Professional, and a guest editor of the Annals of Software Engineering. He has headed 16 research projects as principal investigator and manager, and has 15 year research, teaching and consulting experience in computer science and software engineering. He has published more than 60 papers and won a number of research achievement and academic teaching awards.

Islam Choudhury is a PhD candidate in computer science at South Bank University. He graduated from the University of Wales, College of Cardiff with a BSc Honours in 1989 and a Postgraduate Diploma in Systems Engineering in 1991. He is a member of BCS. He has worked in both academia and industry. His research interests include object-oriented analysis, design and modelling, particularly in business modelling and identifying generic components that can be reused across many industrial sectors.

Dilip Patel is the Head of the Centre for Information and Organisation Studies and Principal Lecturer in Information Systems in the School of Computing, Information Systems and Mathematics, at South Bank University, London. He received a PhD at the same university. His teaching interests include Object Oriented Information Systems, Software Engineering, Systems Analysis and Design and Project Management. His research interests span Information and Business Modeling and Databases. He is a founder member of the Object Oriented Information Systems conference series, and annually organizes a Workshop on Business Objects at the OOPSLA conference.

Shushma Patel is a senior lecturer in Information Systems at the School of Computing, Information Systems and Mathematics, South Bank University, London. She received a PhD from one of the Royal Postgraduate Medical Schools, University of London in 1992. Her teaching is mainly in information systems, research methods and project management. The main core of research is in information systems and medical informatics.
**Alec Dorling** is the Deputy Divisional Manager at IVF’s Centre for Software Engineering in Sweden. He received a BSc in Computer Science from Leicester Polytechnic Univ. in 1973. He is the international project leader of SPICE and convenor of the JTC1/SC7 WG10 developing the 15504 standard on software process assessment. He is also the project manager for several EC projects. Prior to joining IVF, he was the International and Strategic Projects Manager at the European Software Institute in Spain. He is a chartered engineer with 25 years experience in the software industry gained both in real-time and commercial systems environments. He is an internationally recognised expert in the field of Software Quality Management and Process Improvement.

**Håkan Wickberg** is the Manager of IVF’s Centre for Software Engineering in Sweden. He received a MSc from Chalmers Univ. in Gothenburg, Sweden. He has spent great effort in the last six years to build up the Centre which is now widely recognised and includes top class researchers not only from Sweden but also from all over the world. Before joining IVF he was working with real time systems in information and communication technology as well as industrial systems. He is a recognised expert in the European software process assessment and improvement networks. He is the national project leader of SPICE and manages the SPICE local trails centre in Sweden. In the systems and software procurement area He is currently running three EC SPRITE projects which involves the major procurement organisations in Europe.

**Graham King** is Head of Research for the Systems Engineering Faculty of Southampton Institute and Professor of Computer Systems Engineering. His first degree profile is computing and electronics and his MPhil award was for research into data compression for image processing. Subsequently his PhD was awarded for the study of high performance architectures for signal processing. This background has created an abiding interest in the cultural differences between the engineering of hardware and software and have led to a strong interest in all aspects of software quality which has become a major thrust of his research centre.