a reader's interpretation could be not the same with the model creator's interpretation; -there are several separate models (e.g. static, dynamic, functional models) that are difficult to integrate and keep consistent; -the models are validated informally; -to verify program correctness, derive properties and make automatic optimization is not possible;

On the other hand, formal object-oriented model that are based on formal languages, such as TROLL [Jungclaus et al. 98], Object-Z [Smith 94], F-Logic [Kifer and Lausen 90], provide a high degree of semantic accuracy, but they can be applied only to a limited number of projects with specially trained personnel, because they are difficult to understand and to communicate to the user. In the waste majority of cases, domain experts, who are forced to use rigid formal notations, spend most of their time trying to find a way to express the model they had in mind with the required notation, rather than trying to better model the problem. This results in substantial growth of errors, rather than their reduction.

The need of integrating graphic semi-formal approaches close to the needs of the application domains with powerful formal analysis and verification techniques can be met by adding formal semantics to the graphic methods, so that a simple formal kernel model is added to a user-friendly notation. In this way specifications expressed in the user-friendly notation can be verified using formal techniques based on the formal kernel model. The basic idea for combining the advantages of mathematical and graphic approaches is to use mathematical notation in a transparent way. This approach has advantages over a purely graphic specification development as well as over a purely mathematical development because it introduces precision of specification into a software development practice while still ensuring acceptance and useability by current developers.

1.2 Contributions of this work

In this paper, we define a conceptual object-oriented model that is based on order-sorted dynamic logic with equality, following the ideas presented in [Wieringa et al. 94, Wieringa and Broersen 98]. This conceptual model formally represents the information acquired during object-oriented analysis and design. The principal difference between our model and other object-oriented formal models is that the former allows the representation of interconnections between model and meta-model entities. We use dynamic logic as the formal kernel language due to its simplicity, high expressive power and appropriateness for representation of behavioral and structural concepts of object-oriented systems.

To make it easy for engineers to handle the proposed formal conceptual model, we are developing a semi-automatic transformation method, which defines a set of rules to systematically create a single integrated dynamic logic model from the several separate elements that constitute a description of an object-oriented system expressed in Unified Modeling Language (UML) [Rational 97]. We have selected the UML because that it is a set of object-oriented modeling notations accepted by the OMG as an industrial standard for object-oriented analysis and design notation.

1.3 Different Levels of Specification: Model vs. Meta-model

A meta-model is a model for the information that can be expressed during software modeling. Basically, a metamodel is a model of models. It consists of entities defining the model language such as Class_diagrams, State_charts, Sequence_diagrams and so on.

On the other hand, a model describes the objects inherent to the application domain: for instance: Chess, Piece, Pawn and so on.

The ‘figure a’ illustrates the relationship between the two separate levels of specification. Spec_UML is a description of the meta-model elements, its semantics is the set of all the well-formed UML models. It contains constraints over meta-model entities, such as ‘generalization is acyclic’. On the other hand, Spec_SYS is a description of the objects in a particular system. It expresses constraints over objects, such as ‘after a checkmate the game must stop’. This separation of concepts leads to the following problem: if both specifications are separate (or expressed in different formalisms), to express relationships between entities belonging to different levels is not possible.
1.4 Related Work

A number of approaches to formalizing object-oriented modeling techniques have been proposed. We classify them in two different groups: model-based and meta-model-based approaches.

- In the model-based approaches (e.g. see [Moreira and Clark 94, France et al. 97(a), Goldsack and Kent 96, Waldoke et al.97]) a formal specification of the system is generated from a semi-formal graphical object oriented model. The key components of this approach are rules for mapping syntactic structures in the graphical modeling domain to artifacts in the formal modeling domain. In this way, specifications expressed using a user-friendly notation have a semantics in the formal kernel model. As the resulting specification is executable, prototyping can be used to find and correct omissions, contradictions and ambiguities early in the development process.

- In the meta-model-based approaches (e.g. see [France et al. 97 (b), Breu et al. 97, Klar and Geisler 97, Overgaard 98]), rather than generate formal specifications from each semi-formal model, the objective is to give a precise description of core concepts of the graphical modeling notation and provide rules for analyzing their properties. As a consequence of this precise description, the semi-formal models become formal and then amenable to rigorous analysis.

In [Pons 98] we analyze the principal differences between both approaches through an example. This analysis makes evident that the model-based approach is more appropriate for the specification of the information that is inherent to the application. On the other hand, the meta-model-based approach allows the representation of constraints over the meta-model entities in an adequate way. None of these approaches allows the specification of consistency constraints between entities belonging to different levels.

2. A two layers conceptual model

After studying the advantages and disadvantages of several approaches to formalizing object-oriented modeling techniques (see [Pons 98] for a deeper discussion about this topic) we arrived to the conclusion that an appropriate formalization of an object-oriented modeling technique