Architecture Framework for Modeling the Deployment of Parallel Applications on Parallel Computing Platforms

Bedir Tekinerdogan[1] and Ethem Arkin[2]
[1]*Information Technology Group, Wageningen University, Wageningen, Netherlands*
[2]*Aselsan A.Ş., Ankara, Turkey*

Keywords: Parallel Computing, Architecture Modeling, Architecture Viewpoint.

Abstract: To increase the computing performance the current trend is towards applying parallel computing in which the tasks are run in parallel on multiple nodes. Current approaches in parallel computing tend to focus on mapping parallel *algorithms* to parallel computing platforms. However, the complexity and variety of current software systems goes beyond the notion of algorithms only, and needs to consider the design from a broader *application* perspective that requires explicit design abstractions. For this purpose, we propose an architecture framework for modeling parallel applications to support the communication among the stakeholders, to reason about the design decisions and to support the analysis of the architectural design. The architecture framework consists of six coherent set of viewpoints which addresses different concerns in the design of parallel applications. The architecture framework is based on a metamodel that is derived after a thorough domain analysis on parallel computing. To support the architecture design process we have also developed the corresponding tool set that implements the architecture framework. The application of the architecture framework is illustrated for an order management application.

1 INTRODUCTION

It is now increasingly acknowledged that the processing power of a single processor has reached the physical limitations and likewise serial computing has reached its limits. To increase the performance of computing approaches the current trend is towards applying parallel computing on multiple nodes typically including many CPUs. In contrast to serial computing in which instructions are executed serially, in parallel computing multiple processing elements are used to execute the program instructions simultaneously. To benefit from the parallel computing power, usually parallel algorithms are defined that can be executed simultaneously on multiple nodes. As such, increasing the processing nodes will increase the performance of the parallel programs.

Different studies have been carried out on the design and analysis of parallel algorithms to support parallel computing (Amdahl, 2007) (Frank, 2002) (Pllana and Fahringer, 2002). These studies have provided useful results and further increased the performance of parallel computing. Several important challenges have been identified and tackled in parallel computing related to activities such as the analysis of the parallel algorithm, the definition of the logical configuration of the platform, and the mapping of the algorithm to the logical configuration platform. The research on parallel algorithms and its mapping to parallel computing platforms is still ongoing.

Together with the overall developments in parallel computing we can also observe the increasing complexity and variety of current software systems. Here the design problem goes beyond the notion of algorithms and data structures of the computation, and the design of the overall system structure of the parallel computing systems emerges as an important problem. In this context in particular the architecture design and modeling of parallel computing systems is important to support the communication among the stakeholders, to reason about the design decisions during the mapping process and to analyze the eventual design. In current parallel computing approaches, however, there do not seem to be architectural modeling approaches for supporting the design and analysis of parallel computing systems. Most approaches seem either to adopt conceptual modeling approaches in which the parallel computing elements are represented using idiosyncratic models or are generally low level and machine specific (Patyart et. al., 2012). A few approaches borrow for

Tekinerdogan B. and Arkin E..
Architecture Framework for Modeling the Deployment of Parallel Applications on Parallel Computing Platforms.
DOI: 10.5220/0005229301850192
In *Proceedings of the 3rd International Conference on Model-Driven Engineering and Software Development* (MODELSWARD-2015), pages 185-192
ISBN: 978-989-758-083-3
Copyright © 2015 SCITEPRESS (Science and Technology Publications, Lda.)

example models from embedded and real time systems and try to adapt these for parallel computing but do not provide dedicated modeling support for communication and analysis of the concerns of parallel computing architectures. The lack of a clear and precise modeling approach with first class abstractions for parallel computing obviously impedes the solutions for analyzing, designing and communicating the decisions on parallel computing.

Our focus in this paper is on architectural modeling in the context of parallel computing. In the architecture design community broad knowledge has now been gained on modeling the systemic structure and behavior of systems. An important practice is to model and document different architectural views for describing the architecture according to the stakeholders' concerns. An architectural view is a representation of a set of system elements and relations associated with them to support a particular concern. Having multiple views helps to separate the concerns and as such support the modeling, understanding, communication and analysis of the software architecture for different stakeholders.

To provide dedicated support for parallel computing concerns we propose an architecture framework for supporting the modeling of parallel computing architectures. For this, we first provide the overall metamodel that defines the concepts required in modeling parallel computing system architectures. The metamodel is derived after a thorough domain analysis to parallel computing and the related problems. Based on this metamodel we introduce six coherent set of architecture viewpoints each of which focuses on a particular concern in parallel computing. The architecture framework is supported by the corresponding tool workbench that can be used by parallel computing architects to design the parallel computing system.

The remainder of the paper is organized as follows. In section 2, we describe the background on parallel computing and software architecture viewpoints, and describe the problem statement. Section 3 presents the metamodel for parallel applications. Section 4 describes the viewpoints and the approach for using these viewpoints. In section 5, we present the implementation and the toolset for the architecture framework. Section 6 presents the related work and finally we conclude the paper in section 7.

2 BACKGROUND AND PROBLEM STATEMENT

To increase the performance of computing the current trend is towards applying parallel computing on multiple nodes. Unlike serial computing in which instructions are executed serially, multiple processing elements are used to execute the program instructions simultaneously. To benefit from the parallel computing power usually parallel algorithms are defined that can be executed simultaneously on multiple nodes. Hereby, increasing the number of processing nodes usually increases the performance of the parallel programs (Amdahl, 2007)(Gustafson, 1988)(Hill and Marty, 2008). In general, a parallel algorithm can be mapped in different alternative ways to the processing nodes and research has been carried out to optimize the algorithm and the mapping process. This problem has gained even more attention with the dramatic increase of the processing nodes to tens and hundreds of thousands of nodes providing processing performance from petascale to exascale levels (Kogge et. al., 2008). Once a feasible mapping is selected the parallel algorithm needs to be transformed to the target parallel computing platform such as MPI, OpenMP, MPL, and CILK (Talia, 2001).

Despite of the interesting development the challenges in parallel computing are still active. In this paper we focus on the challenges with respect to software architecture modeling. A close analysis of parallel computing research shows that the well-defined concept of *algorithm* is prevailing and the broader consideration of software *application* and its mapping to parallel computing platform does not seem to have got much attention. To illustrate the problem we will use the Order Management Application architecture as an example. The Order Management application is typically a critical part of commercial systems including, for example, packages like Order Entry, Financial and Inventory. To increase the performance of such a system several modules need to be run in parallel. Here we can already observe that the parallel modules are not just well-defined algorithms but can also consist of domain-specific modules of the application. For example, it might be decided that modules such as Order Change and Order Validation in the example should run in parallel over multiple nodes. Obviously for modeling the mapping of parallel applications to parallel computing platforms we need to address more than the mapping of a parallel algorithm to a selected computer architecture. Given the current architectural modeling approaches no direct and integrated support is provided to model the above concerns. Some approaches focus on a particular concern but to the best of our knowledge none of the approaches provide an integrated approach for architectural modeling of the mapping of parallel applications to parallel computing platforms. In the subsequent sections we will describe the architecture framework for addressing each of these concerns.

3 ARCHITECTURE VIEWPOINTS

To explicitly address the concerns related to the design of parallel computing systems we present an architecture framework that defines a coherent set of viewpoints. An *architecture framework* organizes and structures the proposed architectural viewpoints. To define the viewpoints we will adopt the recommended standard for architecture description as it is defined in (ISO/IEC 42010:2011). Figure 1 shows the metamodel on which the architecture framework and the corresponding viewpoints will be based. Based on the metamodel of Figure 1 we define the architecture framework consisting of a coherent set of viewpoints for parallel computing systems. The viewpoints aim to address the concerns of parallel applications in particular.

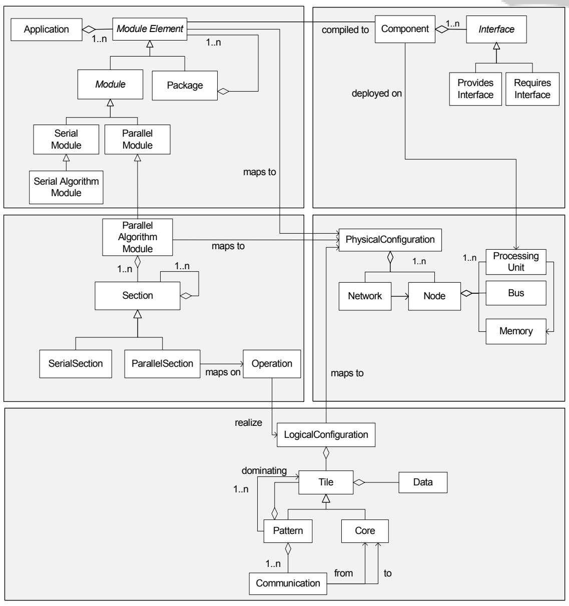
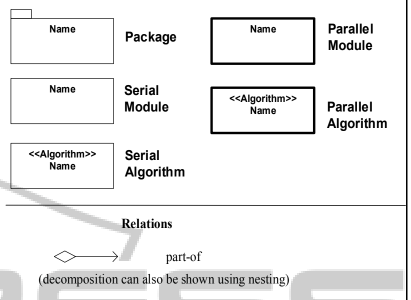


Figure 1: Metamodel for the parallel computing system.

3.1 Application Decomposition Viewpoint

The *application decomposition viewpoint* is shown in Table 1.

Table 1: Application Decomposition Viewpoint.

Section	Description
«Viewpoint Name»	Application Decomposition Viewpoint
«Overview»	The decomposition of the application to modules
«Concerns»	What is the decomposition of the application? How much of the application can be parallel?
«Typical Stakeholders»	Software Architect
«Constraints »	A module can be either serial or parallel
«Model types and notation»	 Name Package Name Parallel Module Name Serial Module <<Algorithm>> Name Parallel Algorithm <<Algorithm>> Name Serial Algorithm Relations part-of (decomposition can also be shown using nesting)

The viewpoint is used to indicate the modules from which the application is composed, and on the other hand defines the parallelism property for each module. In alignment with the metamodel the application can consist of modules or algorithms which can be serial or parallel. Each module can be either serial or parallel. Hence we have defined four different types of modules, *Serial Module, Parallel Module, Serial Algorithm,* and *Parallel Algorithm. Package* is the conventional grouping module for grouping a set of modules together. Based on the metamodel we have defined the Application Decomposition Viewpoint as shown in Table 1.

Using the viewpoint we can now model the application decomposition view for the given example. Figure 2 shows an example Order Management Application Decomposition view that includes three packages.

In the example, the *Order Entry* package consists of seven modules, a serial module for *Exception Management*, a parallel algorithm module *Shipping Calculations* and five other parallel modules *Initiate Order, Modify Order, Order Validation. Order Notification* and *Order Creation.* The Financial package has a parallel module *Payment Engine*, two serial modules *Account Management* and *Fraud Detection,* and an algorithm module *Tax Calculations*. The Inventory package has the serial modules *Inventory Management* and *Loss Management*, and a parallel module *Inventory Planning*.

Note that this is an example decomposition that is decided by the architect. In principle other decomposition might be possible based on different

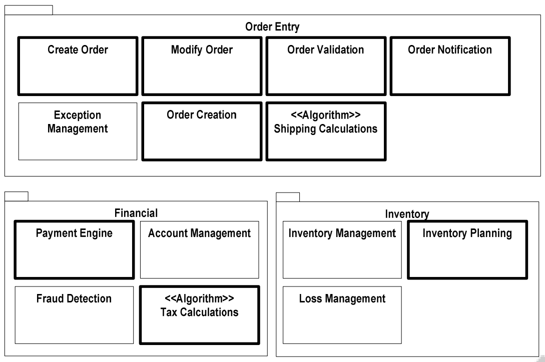


Figure 2: Order Management Application Decomposition View.

design heuristics as discussed in the parallel computing literature (Foster, 1995). Before decomposing the system it will be important to determine whether or not the considered modules can actually be parallelized and where most of the real work is being done. Hereby, the module that account for little computing power could be decided as serial. Using such different design heuristics other decompositions could be investigated to find feasible alternatives. In fact, by providing an explicit view for this we aim to support this design process.

3.2 Algorithm Decomposition Viewpoint

Parallel modules are run in parallel as one unit on multiple nodes and as such will be integrated in a SIMD architecture. Since parallel algorithms are considered as consisting of multiple instructions it is important to analyze the algorithm first and define the serial and parallel sections. To support this we have defined the *Algorithm Decomposition Viewpoint* that is shown in Table 2.

In the Application Decomposition View of the Order Management application in Figure 2, we can identify two parallel algorithms *Tax Calculations* and *Shipping Calculations*. As such we can have two algorithm decomposition views for the application.

As an example we have defined the algorithm decomposition view for *Shipping Calculations* as shown in Figure 3. The algorithm is decomposed into two serial and two parallel sections. The first section is a serial section that initializes the cost parameters per city. The second section distributes the cost parameters to processing units to calculate concurrently. The third section serially calculates the cost per shipping on a processing unit. In the last section, the results for shipping calculations are retrieved from all processing units.

Table 2: Algorithm Decomposition Viewpoint.

Section	**Description**
«Viewpoint Name»	Algorithm Decomposition Viewpoint
«Overview»	The decomposition of the parallel algorithm
«Concerns»	What is the decomposition of the algorithm? Which section can be either serial or parallel?
«Typical Stakeholders»	Algorithm Analyst, System Engineer
«Constraints »	A section can be either serial or parallel
«Model types and notation»	*Index* | *Algorithm Section* | *Section Type* | *Operation* (with two empty rows)

Ind.	*Algorithm Section*	*Section Type*	*Operation*
1	Initialize cost parameters per city	Serial	-
2	Distribute cost parameters	Parallel	Scatter
3	Calculate cost per shipping	Serial	-
4	Get results for shipping calculations	Parallel	Gather

Figure 3: Shipping Calculations Algorithm Decomposition View.

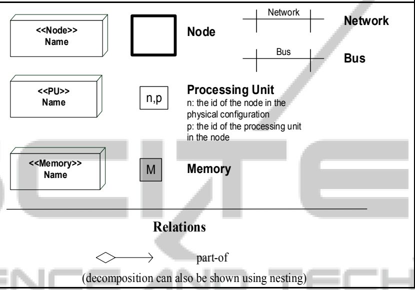
3.3 Physical Configuration Viewpoint

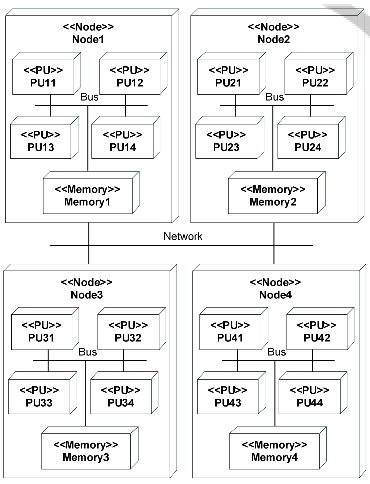
Table 3 shows the *Physical Configuration Viewpoint* for representing the physical parallel computing platform. The viewpoint defines explicit notations for *Node, Processing Unit, Network, Memory Bus* and Memory that are main physical structures of computer architecture.

In alignment with the Flynn's taxonomy (Flynn, 1972), the physical configuration can be defined as shared memory that has multiple processing units that use the same memory, distributed memory in which each node has its own memory, or hybrid memory that has also multiple nodes as distributed memory and each node has multiple processing units with a shared memory.

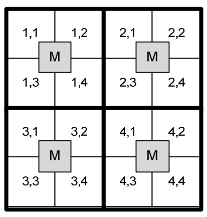
Figure 4 shows two alternative physical configuration view examples. In Figure 4a, the physical configuration is constructed with building blocks over a network. This presentation shows networks and buses explicitly, but for large scale physical configuration views it is hard to present this view. In Figure 4b the physical configuration is presented in a unified structure, that the processing units are represented by rectangles, nodes, buses and networks are represented implicitly in the presentation. This presentation alternative can be more suitable for very large scale parallel computing platforms.

Table 3: Physical Configuration Viewpoint.

Section	Description
«Viewpoint Name»	Physical Configuration Viewpoint
«Overview»	The physical structure of the parallel computing platform.
«Concerns»	What are the structures of the physical computing platform? The configuration is shared memory, distributed memory or hybrid?
«Typical Stakeholders»	System Engineer
«Constraints »	There exists only one Network in a Physical Configuration. There exists only one Bus and a Memory in a Node. If there is one Processing Unit in a Node, there is no need for a Bus.
«Model types and notation»	 <<Node>> Name; Node; Network; Bus; <<PU>> Name; n,p; Processing Unit; n: the id of the node in the physical configuration; p: the id of the processing unit in the node; <<Memory>> Name; M; Memory; Relations; part-of; (decomposition can also be shown using nesting)



(a)



(b)

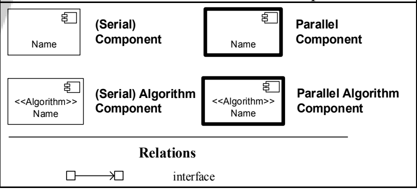
Figure 4: Physical Configuration Views using two different notations.

3.4 Component Viewpoint

The *Component Viewpoint* is shown in Table 4. The component viewport is used for defining the component structure of the parallel application. The components are compiled from the modules that are decomposed in application decomposition view. The components are classified as serial component, serial algorithm component, parallel component, and parallel algorithm component, based on the module that is compiled from. Each component can provide an interface for another component and each component can require an interface from another component. The interface relations are defined between the components.

Based on the application decomposition view for order management application, the component view is shown in Figure 5. Here the parallel and serial components are represented according to the modules defined in application decomposition view. The interface relations between the components are also represented in the view.

Table 4: Component Viewpoint.

Section	Description
«Viewpoint Name»	Component Viewpoint
«Overview»	The component structure for the parallel application
«Concerns»	What are the interface relations between components?
«Typical Stakeholders»	Software Architect, System Engineer
«Constraints »	Each module must have at least one component.
«Model types and notation»	 Name; (Serial) Component; Name; Parallel Component; <<Algorithm>> Name; (Serial) Algorithm Component; <<Algorithm>> Name; Parallel Algorithm Component; Relations; interface

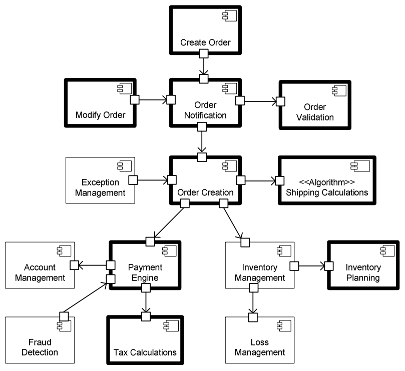


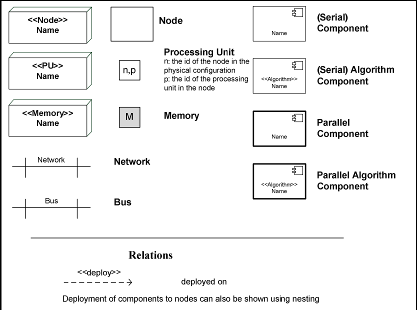
Figure 5: Order Management Component View.

3.5 Deployment Viewpoint

The components defined in the component view must be deployed on processing units defined in the physical configuration. Here, a serial module or a serial algorithm module can be deployed on a single

processing unit. A parallel module or a parallel algorithm module can be deployed on different processing units. The parallel module runs the same instruction sets on multiple processing units and provides output to other components. The parallel algorithm module runs different instruction sets and coordinate data between themselves to calculate a specific algorithm using different processing units. The *Deployment Viewpoint* is shown in Table 5.
Similar to the physical configuration view, the deployment view can also be defined in alternative representations. Figure 6 shows two alternative representation of the order management deployment view. In Figure 6(a), the deployment view is based on the physical configuration view of Figure 4(a), where the relations are shown using <<deploy>> relation. In Figure 6(b), the deployment view is based on the physical configuration view of Figure 4(b) and the relations are shown using nesting. Again, the second deployment view is more suitable for very large scale physical configurations, while the first deployment view represents networking structures explicitly.

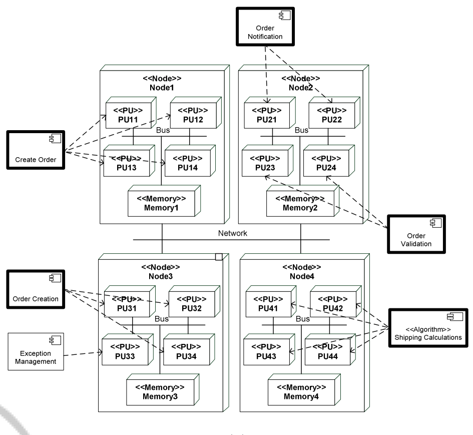
Table 5: Deployment Viewpoint.

Section	Description
«Viewpoint Name»	Deployment Viewpoint
«Overview»	The deployment for the modules of the parallel application
«Concerns»	Which component runs on which processing unit?
«Typical Stakeholders»	System Engineer
«Constraints »	Parallel component can be deployed on different processing units. Serial component can be deployed on a single processing unit.
«Model types and notation»	 <<Node>> Name Node <<PU>> Name n,p Processing Unit n: the id of the node in the physical configuration p: the id of the processing unit in the node <<Memory>> Name M Memory Network Bus Name (Serial) Component <<Algorithm>> Name (Serial) Algorithm Component Parallel Component Parallel Algorithm Component Relations <<deploy>> deployed on Deployment of components to nodes can also be shown using nesting

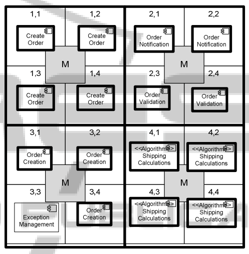
3.6 Logical Configuration Viewpoint

Table 6 shows the *Logical Configuration Viewpoint*, which presents the mapping of the parallel algorithm module to physical configuration together with the required communication links for the algorithm operations.

The previously introduced physical configuration defines the actual physical configuration of the system with the physical communication links among



(a)



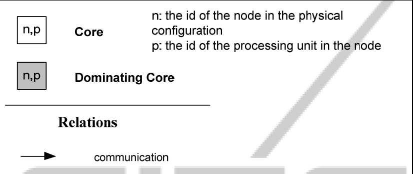
(b)

Figure 6: Order Management Deployment Views.

the processing units. The deployment view defines on which processing units the parallel algorithm module is deployed. The *logical configuration* is a view of the physical configuration that provides the logical communication structure among the physical nodes. Typically, for the same physical configuration we can have many different logical configurations. To represent the mapping of the parallel algorithm to the logical configuration, the cores are identified using the identification values of nodes and processing units. The number of cores should be equal to the selected processing units in the deployment view.

As stated before for each parallel algorithm a corresponding algorithm view is provided. In addition, a logical configuration view needs to be defined to present the communication structures of the physical nodes to realize the parallel algorithm. For example, the earlier introduced algorithm decomposition view for the *Shipping Calculation* algorithm included two parallel sections that implement the *Scatter* and *Gather* operations. The mapping for these operations is defined in the logical configuration view as communication relations between core elements on which the parallel algorithm is deployed. Figure 7 shows an example logical configuration view for both *scatter* and *gather* operations.

Table 6: Logical Configuration Viewpoint.

Section	Description
«Viewpoint Name»	Logical Configuration Viewpoint
«Overview»	The logical structure of the parallel computing platform.
«Concerns»	What are the primitive tiles to map on physical configuration? What are the communication patterns to use?
«Typical Stakeholders»	Algorithm Analyst, System Engineer
«Constraints »	The number of cores should be equal to the processing units in the deployment view. The numbering of the cores should match the numbering in the physical configuration.
«Model types and notation»	 n,p Core n: the id of the node in the physical configuration p: the id of the processing unit in the node n,p Dominating Core Relations communication

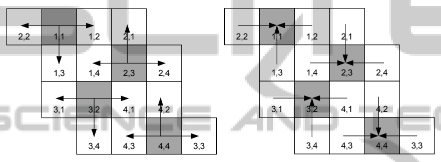


Figure 7: Shipping Algorithm Logical Configuration View. (a) Scatter operation, (b) Gather operation.

3.7 Approach for using Viewpoints

In the previous section we have provided the architecture framework consisting of a coherent set of viewpoints for analysis and design of parallel computing systems. Figure 8 shows the UML activity diagram that represents the process for applying the viewpoints. The process starts initially with the definition of application decomposition and physical configuration views. In principle, for each parallel algorithm module an algorithm decomposition view can be provided. The component view is defined according to the modules as presented in the application decomposition view. The deployment

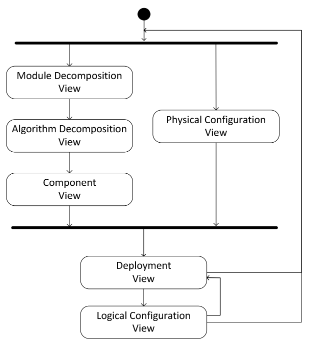

Module Decomposition View
Algorithm Decomposition View
Physical Configuration View
Component View
Deployment View
Logical Configuration View

Figure 8: Approach for Design and Analysis of Parallel Computing System.

view is defined by using both the component view and the physical configuration view. Finally for each algorithm decomposition view, the logical configuration view is defined according to the deployment view. Iterations to the start of the process take place from the deployment view and logical configuration view. Also there are iterations between logical configuration view and deployment view.

4 RELATED WORK

Applying viewpoints to manage parallelism has been proposed by several studies. Rozanski and Woods (2011) propose the *concurrency viewpoint* for describing concerns related to the communication and synchronization mechanisms of concurrent systems. Further they also propose the *deployment viewpoint*, which addresses how to describe the environment into which the system will be deployed. The viewpoints that we have described could be considered as a more domain-specific extension to these viewpoints.

Muhammed et al. (2011) propose the *parallelism viewpoint* to analyze parallelism related overheads in existing parallelism-intensive software systems. The targeted overheads include excessive context switches, uneven distribution of read/write operations and complex thread management structure. The authors propose one viewpoint with five different model kinds including, *Time Distribution*, *Task Distribution*, *Task Type*, *Thread Behaviour* and *Thread Management*. In our approach we have provided an architecture framework consisting of a coherent set of six viewpoints each of which has one or two different kind of notations. We did not directly focus on analyzing parallelism related overheads but focused on the deployment of the modules and algorithms to a parallel computing platform in a parallel application. Arias et al. (2011) focus on the runtime behavior and structure of a software-intensive systems. For this they propose an approach for defining, validating and documenting a set of *execution viewpoints* to support the construction and use of execution views for large software-intensive systems. Ortega-Arjona (2006) defined a parallel application as a specification of a set of sequential process and communication among themselves. According to this definition they propose a performance model and architectural patterns for parallel application.In the literature of parallel computing the particular focus seems to have been on parallel programming models such as MPI, OpenMP, CILK etc. (Talia, 2001) but the design and the modeling got less attention. Several papers have focused in particular on higher level design abstractions in parallel computing and the adoption of model-driven development.

In our earlier study (Arkin et. al., 2013) (Tekinerdogan and Arkin, 2013), we have proposed an architecture framework for mapping parallel algorithms to parallel computing platforms. In that study we only focused on parallel algorithms and did not consider the broader concept of application. Also we assumed a distributed memory model in which each node has its own memory unit and, as such, targeted the MISD architecture of the Flynn's taxonomy. The current approach focuses on software application and is more general in the sense that it supports both modules and algorithms, can represent different memory models, and supports modeling different computing architectures.

5 CONCLUSIONS

The current trend towards increased parallelization of software systems requires proper architecture design approaches for modeling and analysis of concerns related to parallel computing. In this paper we have primarily focused on the mapping of parallel applications on parallel computing platforms. In the current parallel computing literature the focus has been mainly on mapping algorithms to computing platform. Our approach adopts a broader perspective and considers the mapping of software application consisting of modules and algorithms to different computing platforms. Although various viewpoints exist in the software engineering community to cope with parallelism the architecture framework is novel since it provides a coherent and integrated set of viewpoints dedicated for mapping parallel applications to parallel computing platforms. In addition to the viewpoints we have also provided the corresponding approach that describes the logical order in defining the views. We have illustrated the approach for the Order Management case study. The architecture framework is useful in supporting the architecture design process of parallel applications and supports the communication among stakeholders, the guidance of the development of the system and the analysis of the system. In our future work we will further refine the tool support and elaborate on the design of parallel applications using the presented architecture framework.

REFERENCES

Amdahl, G.M., 2007. Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities, Re. fr. the AFIPS Conf. Proc., Vol. 30 (Atlantic City, N.J., Apr. 18–20), AFIPS Press, Reston, Va., 1967, pp. 483–485, *Solid-State Circuits Newsletter, IEEE*, vol.12, no.3, pp.19,20.

Arias, T.B.C., Avgeriou, P., America, P., 2011. Defining and documenting execution viewpoints for a large and complex software-intensive system. Elsevier Journal of Systems and Software 84 1447– 1461.

Arkin, E., Tekinerdogan, B., Imre. K., 2013. Model-Driven Approach for Supporting the Mapping of Parallel Algorithms to Parallel Computing Platforms. *Proc. of the ACM/IEEE 16th Int. Conf. on Model Driven Engineering Languages and System.*

Flynn, M., 1972. Some Computer Organizations and Their Effectiveness, *Computers, IEEE Transactions on*, vol. C-21, no.9, pp.948, 960.

Foster, I., 1995. Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering. *Addison-Wesley Longman Publishing Co.*, Boston, MA, USA.

Frank, M.P., 2002. The physical limits of computing, *Computing in Science & Engineering*, vol.4, no.3, pp.16, 26.

Gustafson, J.L., 1988. Reevaluating Amdahl's law, Communications of the ACM, v 31, n 5, p 532-533.

Hill, M.D., Marty, M.R., 2008. Amdahl's Law in the Multicore Era, *Computer*, vol.41, no.7, pp.33, 38.

[ISO/IEC 42010:2011], 2011. Systems and Software Engineering – Architecture Description.

Kogge, P., et al. 2008. *Exascale Computing Study: Technology Challenges in Achieving Exascale Systems*. DARPA.

Ortega-Arjona, J.L., 2006. Architectural Patterns for Parallel Programming: Models for Performance Estimation, Phd. Thesis, Dept. of Computer Science, University College London.

Pllana, S., Fahringer, T., 2002. UML based modeling of performance oriented parallel and distributed applications, *Simulation Conf., 2002. Proceedings of the Winter*, vol.1, no., pp.497, 505 vol.1, 8-11.

Rozanski, N., Woods, E., 2011. Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives, Second Edition, Addison-Wesley.

Talia, D., 2001. Models and Trends in Parallel Programming. *Parallel Algorithms and Applications* 16, no. 2: 145-180.

Tekinerdogan, B., Arkin, E., 2013. Architecture Framework for Mapping Parallel Algorithms to Parallel Computing Platforms, *Proc. of the 2nd Int. Workshop on Model-Driven Engineering for High Performance and CLoud computing., Miami.*