Building a Consistent Taxonomy for Parallel Programming Models

Markus Nestmann[1]

Abstract: Parallel programming has been a challenge for developers and software engineers for over two decades now. To lower the complexity of parallel programs, a lot of different parallel programming models (like ACTORs) or supporting libraries (like MPI) have been introduced. These models and libraries support different features and have individual hardware requirements.
As part of a software development process a software engineer has to choose which programming model or library is used and fits best for a specific use case. Usually this discussion is made in an early design phase and depends on multiple factors.
To enable the software engineer to make a well-informed decision a taxonomy is required, that contains all necessary information. To find such a taxonomy we performed a **S**ystematic **L**iterature **R**eview, within we found five taxonomies. However, the found taxonomies are inconsistent regarding structure, terms and included models. This paper discusses the five found taxonomies and we will propose a new taxonomy, that overcomes their shortages and combines their features. With our proposed taxonomy software engineers are able to make well-informed decisions already in early phases of the software development process.

Keywords: Parallel Programming Models, Overview, Taxonomy

1 Introduction

Over a decade ago the validity of Moore's law seemed threaten, because single core processors reached a performance limit, due to various factors like overheating. At around the same time multicore CPU's (CPU's which contains multiple cores) became wildly used to overcome performance issues. Nowadays multicore CPU's can be found in all kind of devices (i.e., smartphones, desktop PC's, or Servers). To use these CPU's in an efficient way, it is necessary, that the software running on these devices is written in a way, that it supports parallel execution. However, developing parallel software is complex due to additional challenges (i.e., synchronisation, race conditions, etc.). Therefore, new parallel programing models and frameworks emerged, which abstract the level of parallelism to an extend, that it is easier to handle. Examples are: OpenMP [2] or Erlang [3]. These languages are based on parallel programming models, like the Parallel Random Access Machine (PRAM).

[1] Technische Universität Chemnitz, Fakultät für Informatik, Professur Softwaretechnik, Straße der Nationen 62, 09111 Chemnitz, nesma@hrz.tu-chemnitz.de
[2] http://openmp.org/wp/
[3] https://www.erlang.org/

doi:10.18420/in2017_245

However, choosing the right model or framework is not a trivial task. Every model has other features and drawbacks. Therefore, choosing a model is a design decision made by a software engineer in an early design phase and cannot be undone easily. This means further, the software engineer needs to gather founded information about different models as well as information about the hardware, the software should run on, to be enable to make a well-informed decision [FH16].

To support the software engineers decision making process an overview of the features and requirements for different models is necessary.

A **S**ystematic **L**iterature **R**eview(SLR)[Ne16] revealed five different taxonomies ([Li11], [Zh07], [DMN12], [Je11], [Be13]). However, when evaluating the found taxonomies it became clear that they also fail to provide a complete set of aspects desired for a taxonomy. In this paper we discuss and analyze the features and the shortcomings of the found papers. Based on the analyses we propose a combined taxonomy to overcome the problematic inconsistencies and flaws of the found taxonomies with the purpose to sufficiently differentiate between parallel programming models. Further we identify requirements for such a taxonomy by answering the following research question:

RQ: What is a consistent structure for parallel programming models and which criteria have to be used to sufficiently differentiate between the models?

As a result we present a taxonomy which possesses relevant criteria for differentiating between programming models. This taxonomy can now be used by software engineers to make well-informed decisions.

This paper begins with a short section about the performed SLR. Then the requirements of the desired taxonomy will be defined in the following section. After that we discuss the five found taxonomies regarding their underlying motivation, structure, and what requirements they miss. We conclude with the presentation of our proposed taxonomy.

2 Systematic Literature Review

To answer the research question a Systematic Literature Review was performed during a bachelor thesis [Ne16]. The goal was to find all relevant sources that deal with and present multiple parallel programming models. It was performed to get a list of currently used parallel programming models and to find differences between taxonomies. The thesis is a meta-survey, so especially surveys and overviews were interesting. The SLR was performed according to Kitchenham [KC07] using the meta search engine Google Scholar [4]. The following lists show the search terms (S), the inclusion criteria (I), and the exclusion

[4] https://scholar.google.de/

criteria (E). Synonyms for the search terms were *review*, *overview*, *parallel model*, *parallel computing*, and *model*.

Search Terms:

S_1 *Survey AND "Parallel Programming Model"*

S_2 *Survey AND Models AND "Parallel Computation"*

Inclusion Criteria:

I_1 Results, that compare parallel programming models.

I_2 Results, that describe and present multiple parallel programming models.

I_3 Results, that introduce a taxonomy for parallel programming models.

Exclusion Criteria:

E_1 Results, that don't meet the search criteria.

E_2 Results, that were written in other languages than german or english.

E_3 Results, that are e.g. PowerPoint presentations.

E_4 Results, that are not available using the TU Chemnitz access.

E_5 Results, that don't go into parallel programming models.

E_6 Results, that don't cover multiple parallel programming models.

The search terms with their variations led on the 11.08.2016 to 20 results. Google Scholar filters were used to get results between 2006 and 2016 and the filter *intitle:* was used to search for the term Survey and it's synonyms in the title. The whole documentation of the SLR can be found in the bachelor thesis [Ne16]. After analyzing these 20 papers 9 of them were excluded due to exclusions criteria, so 11 papers were used in the bachelor thesis. All of them cover multiple parallel programming models but just 5 of them contain a taxonomy. These 5 papers ([Li11], [Zh07], [DMN12], [Je11], [Be13]) are now the foundation of our paper and we will discuss them later on.

3 Requirements of a Sufficient and Consistent Taxonomy

To find the desired taxonomy, it is first necessary to characterize the requirements for such a taxonomy. The requirements depent on the intended usage of the taxonomy, so we first have to define our motivation and then afterwards introduce the requirements. Our motivation is to have a taxonomy that can aid in the choosing process of a parallel programming model. For that it has to differentiate between the models using criteria that are interesting in the choosing process. To find these criteria and finally the requirements we used results from the SLR [Ne16]. Especially R_4 and R_5 were included after seeing the increasing popularity of these groups in the papers of the SLR (see also chapter 5). The first three requirements arose mainly from our motivation. R_2 was therefore included because it is important in the choosing process but also because we found this differentiation in the papers. More details can be seen in the thesis. In the following we present a well-considered list of requirements.

The taxonomy has to...

R_1 ...include all currently used parallel programming models, because most likely current models will be interesting for the software engineer in the choosing process.

R_2 ...give information about the memory architecture of the models, namely shared-, distributed- or hierarchical memory, so that the user can pick a model fitting to his hardware.

R_3 ...provide sufficient criteria to differentiate between all included models, so that it can aid the software engineer in the choosing process.

R_4 ...include GPGPU models (models that use also the GPU), because these models are gaining popularity and will be relevant to many developers today.

R_5 ...include hybrid models (they combine existing models, e.g. MPI + Pthreads), because some of them combine the advantages of existing models and allow a better runtime for some special applications. This gives the software engineer more possibilities.

4 Existing Taxonomies

In this section the mentioned five taxonomies will be discussed.

The taxonomy of Xin Li [Li11] includes just task-based parallel programming models, because the focus was on automatic analyzes in APIs for task-level compiler. The author also wanted to look on the efficiency of models and therefore chose a division of these task-based models in control- and data driven models. That is basically a division in different execution mechanisms. The taxonomy was created for a specific research and therefore doesn't include libraries that are currently popular and widely used, such as MPI.

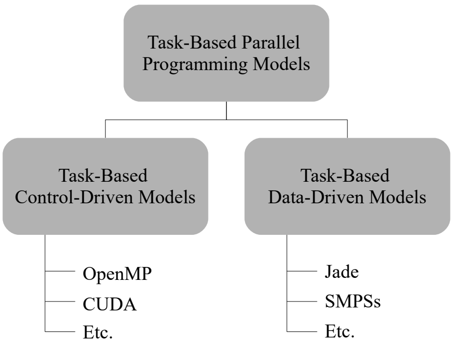
Task-Based Parallel Programming Models
Task-Based Control-Driven Models
Task-Based Data-Driven Models
OpenMP
CUDA
Etc.
Jade
SMPSs
Etc.

Fig. 1: Figure based on taxonomy from [Li11]

The taxonomy of [Zh07] gives a historic view on parallel programming models. The authors showed that first there were models based on shared memory. After that the first models with distributed memory appeared and now the models with hierarchical memory are gaining popularity. That's why the taxonomy divides the parallel programming models in these three groups. Especially in the group of models with hierarchical memory a lot of current models are covered.

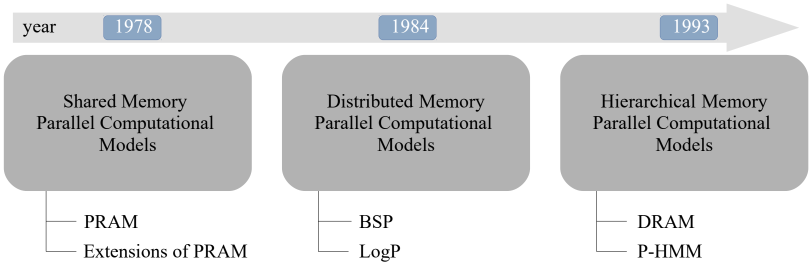
year
1978
1984
1993
Shared Memory Parallel Computational Models
Distributed Memory Parallel Computational Models
Hierarchical Memory Parallel Computational Models
PRAM
Extensions of PRAM
BSP
LogP
DRAM
P-HMM

Fig. 2: Figure based on taxonomy from [Zh07]

The authors Javier Diaz et al. [DMN12] introduce in their paper a taxonomy with four groups of parallel programming models. This allows a very broad view over the models and the authors also included currently used models. The paper of the authors discussed parallel programming models regarding their suitability for the High-Performance-Computing community. For that they chose one model for each target architecture: models for shared-/distributed architectures, GPGPU models and hybrid models (combine models of the other groups). The model PGAS forms a group of itself.

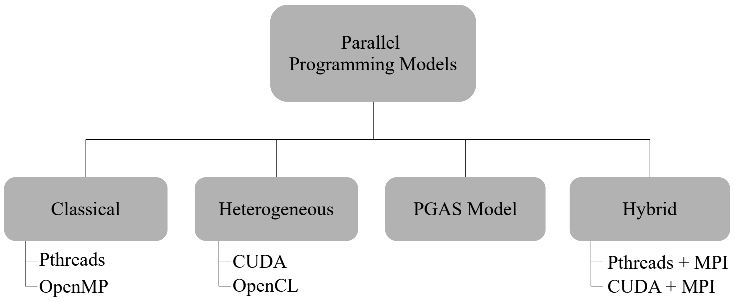

Fig. 3: Figure based on taxonomy from [DMN12]

The paper of Matevz Jekovec [Je11] introduces the most important models of computation. A model of computation is used as a term to describe sequential programming models and their counterpart parallel models which are covered in their paper. Each model belongs to one of the groups shared-, distributed- or hierarchical memory models, like the historic division form [Zh07]. But another criterion is the degree of abstraction. It shows for every model whether it is very abstract or realistic. For the author it was important to research technical bottlenecks to develop efficient algorithms. The memory architecture is an important aspect for this. That's why he structured the taxonomy as he did. In the figure only the parallel programming models (right side) are displayed.

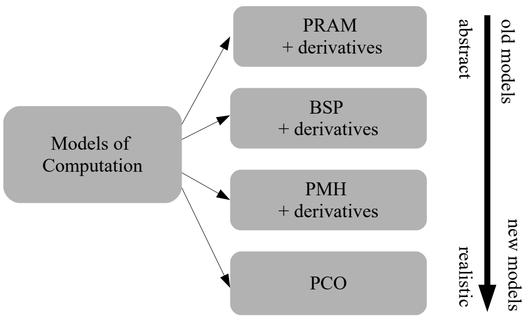

Fig. 4: Figure based on taxonomy from [Je11]

Evgenij Belikov et al. [Be13] have the largest overview in their paper compared to the other four. They use the term parallel programming models in another sense than the other authors. For example models like PRAM are not listed but a lot of parallel programming languages like JAVA or Erlang and also many libraries. Because of that they structured the models in many clusters that are also based on programming language properties. They differentiate for example between declarative and imperative languages. In the paper they focus on the efficiency of models and and support the opinion that the abstraction of Coordination and Computation has to be balanced for the best result. That's why this is included in their taxonomy. The software engineer can see how each of the models manages this abstraction.

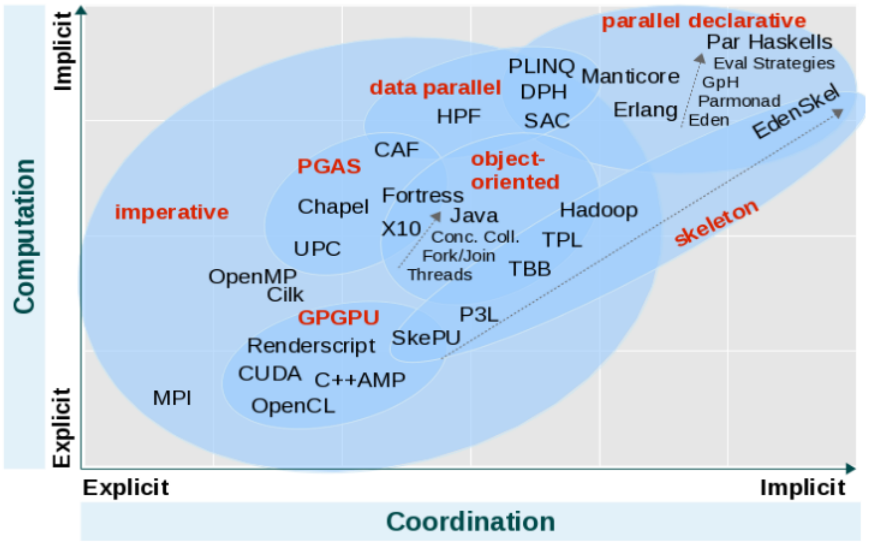

Fig. 5: Taxonomy from [Be13]

Comparison of the Taxonomies

In the following table 1 the fulfilled requirements are shown for each analyzed paper.

The paper [Li11] fulfills the third requirement, because for each model the execution mechanism is provided. So each model can be differentiated using that criteria. But it doesn't include GPGPU and hybrid models and it only covers task-based parallel programming models. The other requirements are therefore not fulfilled.

The paper [Zh07] fulfills the first two requirements, because all current models are covered in one of the three groups, that also differentiate based on the memory architecture, which is the second requirements. GPGPU models or hybrids are not mentioned however.

The paper [DMN12] covers all current models and includes GPGPU and hybrid models. Three requirements are therfore fulfilled. But there is no furhter criteria in the structure to differentiate between the many models that are covered in the first group (classical models). The taxonomy also gives no information about shared-, distributed or hierarchical models. So the second and the third requirement is not fulfilled.

The paper [Je11] gives information about each model regarding the degree of abstraction. The software engineer can differentiate between all models with this criteria. It also groups the models based on the three memory models. But the other requirements are not fulfilled because the authors only chose those models that show parallel structure to the sequential models in their paper.

The paper [Be13] lists a lot of parallel programming languages and libraries but many models are not covered and it also doesn't fulfill the second requirement. But with the inclusion of the GPGPU models the fourth requirement is fulfilled. It is also possible to differentiate between all models because they structured them in clusters and placed them in an diagram that shows how every model manages computation and coordination.

The table 1 shows that no taxonomy from the papers fulfills all requirements.

		taxonomies				
		[Li11]	[Zh07]	[DMN12]	[Je11]	[Be13]
requirements	R_1		✓	✓		
	R_2		✓		✓	
	R_3	✓			✓	✓
	R_4			✓		✓
	R_5			✓		

Tab. 1: Comparison of fulfilled requirements through the taxonomies

5 Proposed Taxonomy

Figure 6 shows our proposed taxonomy that was created to fit the requirements (see section 3).

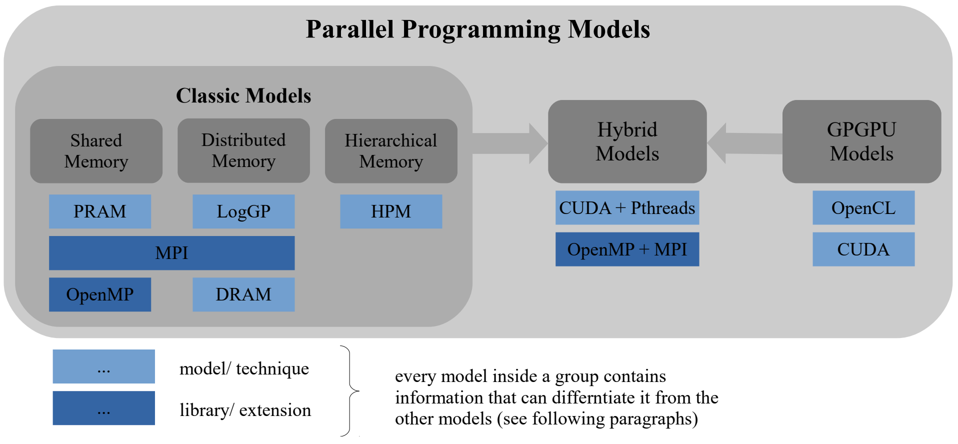

Fig. 6: Own approach for a taxonomy.

Many aspects of the analyzed taxonomies were taken to built a taxonomy that is relevant and includes the most important parts. The taxonomy is vertically and horizontally structured. Horizontally there are three groups. The classic models, the GPGPU models and the hybrid models. First it became clear that a division of the parallel programming models based on the shared-, distributed- and hierarchical memory models was often used ([Je11], [DMN12], [Zh07]) and therefore should be included. That is why the classic models are divided in these three groups. GPGPU models also appeared in many papers with increasing popularity ([Be13], [Li11], [DMN12]). A third group are hybrid parallel programming models which were an example used by Javier Diaz [DMN12] who introduced in his paper many possibilities for combining different parallel programming models.

The mentioned groups will be the horizontal structure but the models inside these groups have to be structured (vertically) too. First it has to be clear what type of models are listed. In the analyzed papers some listed exclusively models and techniques like PRAM, BSP or PGAS. Others listed libraries like OpenMP, MPI or IntelTBB. Both parties described them as parallel programming models. There are many aspects of models that can be used to differentiate them among themselves. Henry Kasim et al. introduced a list of such aspects in their paper [Ka08]. They ask the following questions to know whether these things are done explicitly by the programmer or implicitly by the program itself. This list is sufficient for our use case, so the vertical structure is based on these points:

1. system architecture (shared- or distributed memory?) – already included in horizontal structure
2. programming methodologies (API, new specifications)
3. worker management (creation of threads/worker/processors?)
4. workload partitioning scheme
5. task-to-worker mapping
6. synchronization (when can worker access shared memory?)
7. communication model

The representation of the taxonomy as a table could look like Tab. 2. (The table is just an example to get another perspective and is not complete.)

	Pr. model/language	Type	Sytem Architecture	Data-/Task Parallelism	Implementations	...
Classic	PRAM	model	SM	data	OpenMP, CUDA	
	ArBB	library	SM	data	N/A	
	...					
	BSP	model	DM	N/A	BSPlib	
	LogP	model	DM	N/A	N/A	
	...					
	P-HMM	model	HM	N/A	N/A	
	...					
Hybrid	Pthreads&MPI	libraries	SM	task/data	MPICH2	
	...					
	...					
GPGPU	OpenCL	library	HM	data	OpenCL C	
	...					
	...					

Tab. 2: This is an example for the representation of the taxonomy as a table. All the information for the columns were taken form the analyzed papers. The N/A shows, that not every detail was found in the papers. SM, DM and HM mean shared-, distributed- and hierarchical memory. The three dots indicate that the content of this example table is not complete horizontally and vertically.

6 Evaluation

Now we have to evaluate whether the proposed taxonomy is an improvement to the found taxonomies or not. For this we have to compare the taxonomy to the other ones. The proposed taxonomy includes all current groups of parallel programming models, so all currently used models can be found. The taxonomy gives information about the memory architecture as seen in the group for classical models. With the criteria from Kasim [Ka08] it is also possible to provide sufficient criteria for the differentiation between the models. The taxonomy includes GPGPU models and hybrid models. The summarized comparison can be seen in table 3:

		taxonomies					
		[Li11]	[Zh07]	[DMN12]	[Je11]	[Be13]	new
requirements	R_1		✓	✓		✓	✓
	R_2		✓		✓		✓
	R_3	✓			✓	✓	✓
	R_4			✓		✓	✓
	R_5			✓			✓

Tab. 3: Comparison table (see Tab. 1) with added row for the proposed taxonomy

As seen in the comparison table 3 the proposed taxonomy fulfills the requirements we initially gathered. Therefore we could provide an improved taxonomy that overcomes most identified drawbacks. However, also this taxonomy has some hitches we want to address in the following enumeration. The bachelor thesis [Ne16] and the taxonomy was discussed in a round of experts.

1. The term parallel programming model includes also libraries even though there is a distinction between libraries and models.
2. It is not clear where MPI can be included. Due to he message passing model it can be based on shared- or distributed memory.

7 Application of the Taxonomy

With the example presentation in table 2 the developer get's a helpful overview for the parallel programming models. He can look for GPGPU models or choose a classic model. If the developer looks into the classic models, he can choose a category that fits his environment. If he wants to work with shared memory, he could for example choose the PRAM model. This decision could be based on different reasons. For example preferences in explicit or implicit handling of the worker management or synchronization. Implementations of the chosen model can be found in the last column. It is furthermore possible that the developer chooses a hybrid model that combines his chosen model with another one. A good combination of the advantages from two models can lead to a better performance on special systems (e.g. Pthreads and MPI [Wr06]).

8 Conclusion

Through the analysis of multiple papers that introduce taxonomies for parallel programming models we observed that there are many differences between them. A taxonomy has to fulfill some requirements so that it can aid the developer in his choosing process of a suitable model. We were able to define these requirements and then we discussed the individual taxonomies of the SLR. We were then able to introduce a new approach for a taxonomy that fulfills these requirements. Everyone who wants to choose a parallel programming model and is new to the subject could benefit from this taxonomy. Serving as an overview the developer can get information from the taxonomy about the differences between the current models. This could aid him so that he can make a well-informed decision. We could also show that our proposed taxonomy is an improvement to the taxonomies found in the SLR, nevertheless we are aware of hitches in the taxonomy and open for discussion and further improvement. It is also necessary to adapt the taxonomy when new groups of parallel programming models appear in the future so that it still includes all current models.

References

[Be13] Belikov, Evgenij; Deligiannis, Pantazis; Totoo, Prabhat; Aljabri, Malak; Loidl, Hans-Wolfgang: A survey of high-level parallel programming models. Technical Report HW-MACS-TR-0103, Heriot-Watt University, 2013.

[DMN12] Diaz, Javier; Munoz, Camelia; Nino, Alfonso: A survey of parallel programming models and tools in the multi and many-core era. IEEE Transactions on parallel and distributed systems, pp. 1369–1386, 2012.

[FH16] Frank, Markus; Hilbrich, Marcus: Performance Prediction for Multicore Environments—An Experiment Report. In: Proceedings of the Symposium on Software Performance 2016, 7-9 November 2016, Kiel, Germany. 2016.

[Je11] Jekovec, Matevz: Survey of the sequential and parallel models of computation. 2011.

[Ka08] Kasim, Henry; March, Verdi; Zhang, Rita; See, Simon: Survey on parallel programming model. In: Network and Parallel Computing. Springer, pp. 266–275, 2008.

[KC07] Kitchenham, Barbara; Charters, Stuart: Guidelines for performing systematic literature reviews in software engineering. 2007.

[Li11] Li, Xin: A Survey of Task-Based Parallel Programming Models. In: International Conference on Information Technology and Computer Science, 3rd (ITCS 2011). ASME Press, 2011.

[Ne16] Nestmann, Markus: , Erstellung einer einheitlichen Taxonomie für die Programmiermodelle der parallelen Programmierung, 2016. http://nbn-resolving.de/urn:nbn:de:bsz:ch1-qucosa-224238.

[Wr06] Wright, Charles: Hybrid programming fun: Making bzip2 parallel with MPICH2 & Pthreads on the Cray XD1. In: CUG'06, 48th Cray User Group Conference. volume 1. Citeseer, 2006.

[Zh07] Zhang, Yunquan; Chen, Guoliang; Sun, Guangzhong; Miao, Qiankun: Models of parallel computation: a survey and classification. Frontiers of Computer Science in China, 1(2):156–165, 2007.