Casual and Team Collaboration in High Performance Computing

Catalina Danis
Social Computing Group
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598 USA
danis@us.ibm.com

ABSTRACT

Successful collaborations are not only occasions for the accomplishment of shared goals, but also provide opportunities for individual collaborators to learn from each other. The process of knitting together different points of view which derive from differences in profession, culture and other factors is believed by many to be a critical foundation for successful collaboration. This paper examines the opportunities for such collaborative learning among users of High Performance Computing (HPC) systems. It contrasts opportunities for learning that arise naturally in long-term, team collaborative situations with the lesser opportunities in more casual, consultant-client relationships. The discussion explores how factors conducive to successful collaboration in longer, more tightly organized collaboration might be leveraged in more transient collaborative relationships.

1. INTRODUCTION

Researchers in Computer Supported Cooperative Work (CSCW) have long discussed the need for participants in cooperative work to develop a common understanding of the problems they are addressing. Concepts such as *common ground* [3, 7] and *common information spaces* [1, 2], though having very different theoretical underpinnings, articulate fundamental concepts around the development of shared understandings that take place and are believed to be required in successful collaborative interactions. In addition to accomplishing work through the process of developing a common understanding, multi-disciplinary collaborations by the very fact of developing shared understandings provide participants with occasions for learning about each other's areas of expertise. Long-term collaborations characterized by many opportunities for creating shared artifacts around which shared understandings are developed would be expected to provide ample opportunities for such cross-fertilization. However, there many other forms of collaboration [1] which are temporary and may occur among participants who do not know each other well [1, 5]. How much learning takes place under such circumstances? This paper explores how the *form* of the collaboration impacts the opportunity that participants have to learn from each other in the course of accomplishing a common goal. While it is obvious that shorter, less intense forms will provide lesser opportunities, the goal is to explore if and how the more fleeting collaborations might be shaped to improve opportunities for learning.

This discussion take place in the context of code development for High Performance Computing (HPC) systems where there is a severe shortage of the requisite computing skills among scientists whose work depends on the use of HPC systems. Investigations of users in HPC reveals that much of the successful coding for HPC systems is fundamentally a collaborative undertaking[1]. Teams typically involve one or more domain experts (i.e., scientists) and multiple computational experts. In addition to enabling coding work to be completed, a reasonable question is whether and, if so, how such collaborative relationships may provide an opportunity for scientists to learn computation skills from their more computationally skilled collaborators.

This paper explores how particular interactions in naturally occurring collaborative interactions might provide such opportunities for less computationally savvy users. It does this by exploring the quality of the collaboration that takes place in team collaborative relationships and seeks to understand factors that may either facilitate or obstruct collaborative learning. It then asks whether some of these factors might be applicable in the more causal collaboration which takes place between consultants and their clients.

2. HIGH PERFORMANCE COMPUTING

One defining characteristic of HPC is the requirement for parallel, as opposed to serial, coding. Computations may be distributed over up to tens of thousands of processors and the resulting processor output must be re-integrated to produce correct results [6]. Current HPC machines are of teraflop scale, meaning that they are capable of calculating one trillion floating point calculations per second, as measured by a benchmark task. These types of machines are sought after by scientists who need to do calculations that can not be done with conventional, serial machines. One of the computational experts interviewed for this project said with respect to

[1] A significant exception to collaborative coding is that done by "hero programmers." These are exceptionally gifted coders who work individually to develop entire modules or complete codes. However, this investigation and that of Halverson [9] suggests that the prevalence of this mode of working is overstated by HPC folklore.

Parallel machines are becoming more commonplace and their uses are spreading beyond the traditional military and government-driven applications to the commercial space, including applications in finance and entertainment [4]. As was noted above, a serious problem facing the HPC community is the shortage of domain experts who are literate in HPC [4, 6]. Consequently there are missed opportunities to drive the basic scientific and applied work which HPC technology enables.

Not only are there enormous barriers to entry for content experts, current computationally savvy users also have problems switching to new machines and new tools. These tasks of *porting* to a new machine and *tuning* or *optimizing* the code to take advantage of the machine's features, requires the highly specialized knowledge of computer engineering. While parallel coding courses are becoming commonplace in graduate Computer Science departments [6], there is a shortage of computationally expert teachers in science departments. Interviews conducted with computational experts with advanced scientific degrees who have made the transition to computational literacy, reveal a conflict between practicing both science and computation at expert levels. A typical situation is described by an elite code optimizer, trained as an electronic atomic physicist:

"… you learn Fortran, write a program … Because you don't know a lot about computing, your code does not run well, but you spend little time on making it work better because it interferes with the activities you need to survive as a scientist, like publishing…." To focus on optimization, "… you need to develop a body of knowledge about hardware, memory, protocols. Basically, you need to dig deeper, not in Computer Science, but in computer engineering. … I am no longer a practicing scientist." Consequently, domain experts typically learn on the job after completing their graduate training.

The examination of collaboration between scientists and computational experts in this paper is based on interviews with computational experts only. This is a shortcoming which I hope to redress with further data collection where the goal is to interview multiple participants engaged in collaborative relationships (e.g., scientists as well as consultants, multiple members of each team) and to collect communicative artifacts of the collaboration. The current data were collected as part of an initial, broad foundational investigation into work practices of users of HPC systems. It is part of a large, multi-disciplinary project aimed at understanding productivity bottlenecks amongst users of HPC systems in order to develop improved hardware, software and tools.

3. SITES AND STUDY METHODS

The examples in this paper are drawn from interviews with ten individuals who are highly skilled at computation for HPC systems. Five (one manager and four consultants) work at one of the five supercomputing centers in the United States that are funded by the National Science Foundation, a government science agency. The mandate of the supercomputing centers is to serve as a computational resource for scientists whose work requires the use of HPC systems but who do not have other access to HPC systems. Scientists gain access to the center's resources by writing a proposal requesting computer time. The criteria for selection at the supercomputing center where interviews were carried out includes demonstration that the scientist's application "can scale," that is, can make use of the terascale capabilities of the center's machines. Those whose proposals are accepted are given access to help from the center's consultant staff. All of the consultants have post-graduate degrees (2 physicists, a quantum chemist and a civil engineer) and had acquired computation skills, though not necessarily parallel computation skills, during their graduate work. They have pursued computation rather than science as their primary work after completing their studies. In addition to their consulting work they, as well as their manager, develop HPC codes in their areas of interest.

The other five interview participants are current or former employees of an international vendor of HPC hardware and software, which is also the author's employer. Three are current or former employees of the research division. Their work is in the development of tools and applications for experimental HPC systems. The other two are technical employees in an HPC sales organization. Their job responsibilities include improving the performance of clients' codes.

The data reported here were gathered primarily through semi-structured interviews. Each individual was interviewed at least once for between 60 and 90 minutes. Six of the initial interviews were done face-to-face at the individual's workplace; the others were done over the telephone. Nine of the informants were interviewed only by the author. A colleague of the author's participated in the interview of the final informant. The author made transcriptions of audiotaped records (5 of 10) or from handwritten notes made during the interview. The interviews were designed to elicit information about the interview participant (their educational and work training) and to understand their current work (its goals and practices and productivity challenges) in detail. The focus on collaboration emerged during analysis and served as the subject of briefer follow-up interviews. These were done by means of email, telephone or face-to-face with 5 of the original ten informants.

4. FORMS OF COLLABORATION IN HPC

Two forms of collaboration are discussed in this paper. The first form, designated as long-term, *team collaboration*, consists of a small team of core members that has worked together intermittently for a long-period of time (more than a decade in the case discussed here) on a single, evolving code project. The second form, designated as *casual collaboration*, consists of the short-term, sporadic interaction between one scientist who has written a code and one consultant who has been assigned to help him or her bring the code to a running state on one of the supercomputing center's machines. In both forms, the participants share the same overall goal of producing a working piece of code for an HPC system, although they differ in terms of their relationship to the codes. In the team collaboration situation discussed below, all the team members may be considered to own the code. However, in the causal collaboration situations discussed below, the scientist owns the code.

The two forms of collaboration also differ on other dimensions. In terms of the primary method of information exchange, the communication mode of participants in the causal form of collaboration consisted entirely of email or telephone exchanges. In contrast, the communication mode of the participants in the team collaboration varied over the course of their collaborative work. It included intense periods of closely coupled collaboration,

such as when the members needed to make decisions about how to approach a problem, needed to decide on the direction to take in their project or they had reached a point of code integration. Communication during these periods was carried out face-to-face. It also included long periods during which the work of the members was pursued relatively independently. During these periods, their communication consisted of email and telephone exchanges. It should be added, though, that the longer of the causal collaborations were characterized by similar *rhythms* of waning and waxing collaboration though they consisted entirely of email and telephone exchanges.

The two forms of collaborations also differed in the duration of the collaborative work. The team collaborations tended to be very long lived, with the team profiled below having been constituted for over a decade. Another of the team collaborations I learned about through my interviews had also been going on for over a decade, while the third had recently started. The long-lived character refers to the *core* team members, for during the collaborative work they were supported by "… a changing array of graduate students and post-docs." In contrast, the causal collaboration partners worked together for weeks or months.

4.1 Team Collaboration

All five of the employees of the supercomputing center develop their own HPC codes as part of their employment. I have explored team collaboration instances with three of the consultants and I describe one of these in some detail below.

This team works in the domain of earthquake simulation. The portion of their work discussed here has contributed to the porting of their codes, developed over several years, for simulating damage to buildings as a function of the underlying physical substrate to a terascale machine at the supercomputing center. The opportunity to use the supercomputing center's terascale machine enables the simulation of earthquake damage to include buildings in a large portion of the San Fernando Valley (80 kilometers square by 30 kilometers deep). The impetus for the team came from civil engineers who have an interest in predicting how particular buildings need to be built to withstand earthquakes based on the underlying science of the propagation of waves through ground materials and the faults that set off the earthquakes. The version of the code discussed here was completed in 2003 and was awarded the prestigious Gordon Bell prize for code performance, which recognized the team for the level of performance their codes achieved on an HPC machine.

The core collaboration team consisted of two civil engineers, a computer scientist and a code optimizer. The code optimizer was the only person on the team who has been interviewed for this study thus far. In this collaborative situation, all four of the core team members were computationally skilled, though their areas of expertise differed. The civil engineers had developed codes for calculating the propagation of waves through substrates ranging from loose soil through hard rock. The computer scientist is an expert in developing the complex, irregular meshes required to map the physical area of interest onto an appropriate data structure and in distributing the calculations across a large number of processors. The code optimizer brought knowledge of the terascale HPC machine that was required to run the large simulation (one hundred million sub-volumes) to address the scientific question. His role was to adapt theoretical solutions to the real world of a particular HPC machine. Through his expertise in areas such as communication bandwidth, latency, and I/O, he helped the team adapt the codes to the new machine.

From the standpoint of the optimization expert, the rhythm of the work was such that the three members with scientific expertise were able to work independently for periods of several months and then the team would meet face-to-face for periods of three to four weeks of intense closely coupled work. (Additional interviews with the scientific experts might reveal additional periods of tightly coupled work involving sub-teams.)

4.1.1 Learning From Each Other

During the face-to-face periods the various codes developed by the civil engineers and the algorithm expert would be tested on the HPC machine at the supercomputing center. Many test runs would be done to determine how various aspects of the algorithms performed on the target machine. As a result of the performance measurements, areas were identified where the algorithms would need to be modified in order for the codes to take advantage of the properties of the HPC machine. During these periods the team would draw heavily on the expertise of the optimization expert in order to learn how to modify their codes. The team would then disperse and work independently to incorporate their collaboratively acquired knowledge into their codes. The scientists credit the ability to use the center's machine efficiently to their success in being able to run such a large, scientifically interesting simulation. The algorithmic expert noted "we've benefited enormously from having this powerful system at <the supercomputing center> and we've developed algorithms to maximize out ability to use it well" [8]). These algorithmic changes were critical for enabling the scientists to test a big enough problem – mapping an area 80 kilometers square by 30 kilometers deep of the San Fernando Valley – in their simulation. Without integrating the knowledge held by the code optimizer, their scientific endeavor would have failed.

Learning from each other is important not only in enabling a better solution to the shared problem, but also in more subtle ways that enable more effective teaming through give and take on part of the members. The informant noted that learning about each other's domains was also critical for helping team members "avoid making unreasonable demands of each other," and conversely, for preventing "push-back" when the demands were reasonable. In any collaborative problem solving effort, conflicts can arise about how to do things which are particularly difficult to resolve if a solution will have a disproportionate impact on one member or a sub-group of the team. For example, a particular partitioning of the data across processors may be assumed by the data structure expert. However, this configuration may create problems for the optimizer who is aware of the engineering constraints on the network latency and the performance of I/O nodes. To the extent that accommodating another expert's needs creates problems for oneself, there is an opportunity for pushback. However, the informant in this group noted that having an appreciation for the reasons that led the other member to propose the particular solution made the others respond in a more open-minded manner.

In addition to the opportunities for learning from each other that time afforded the team members, additional credit was given to one of the co-principal investigators for his skill at assembling a "learning organization." He emphasized the need for all team

members, including the array of relatively transient graduate students and post-doctoral fellows, to actively learn about each other's work.

4.2 Casual Collaboration

The running codes that resulted from the collaborations between the scientists and the consultants were much smaller than codes developed under the condition of team collaboration. This stands to reason as many of these codes were developed by a single individual, namely the scientist, with various degrees of help from the consultant. The supercomputing center where these observations were done selects applicants based on the expected scalability of their code. The supercomputing center's administrators are interested in determining if "… they are big users (of computing resources) or do they have the potential to be a big user." Consequently, the consultants work with scientists at all levels of computational expertise. "Some start with parallel code, others with buggy parallel code. Some are still developing and want ideas on how to do what they need to do."

Scientists who wanted help initiated these casual collaborations through an email to the supercomputing center's "hotline" once their proposal for computer time was approved. The head of the support organization then assigned a consultant to work with the scientist, matching domain expertise whenever possible. The role of the consultant was to "…help the scientist fix any problems that prevented the code from achieving a production run." The amount and type of help the consultant provided depended in large part on the need of the scientist and the amount of knowledge the consultant had about the scientist's domain. The scientists who worked with the consultants whose data are discussed here varied considerably in their level of computational expertise. However, there was general agreement with the statement of one of the consultants that: "the majority of the users are at the lower skill levels."

4.2.1 Types of Users and Help Received

Those who required less intensive help included computationally capable users who needed support with operating in the machine and infrastructure environment available at the supercomputing center, but could code their applications themselves. The problems encountered by these types of users could nevertheless consume considerable amounts of the consultant's time; frequently on the order of two weeks. One example arose because of different implementations of the programming language MPI2[2] at the supercomputing center and the scientist's home machine. The consultant described the scientist involved in this example as "a good type of user who can isolate for you what he needed." The scientist was suspicious of the results he obtained from his initial runs of his debugged code on the supercomputing center's machine and developed a small one page test program to show the consultant what he intended to do. However, it took several rounds of emails to come to a shared understanding of the solution. The solution developed and debugged by the consultant would fail to produce the correct scientific results when the scientist ran it on his home machine which he was using for development and debugging. Eventually, after several rounds of emails, they were able to develop a joint understanding of the cause of the problem.

At the other end of the range were clients who needed help with their application code. Discussing one such user, one of the consultants noted "I had to literally work with his code. Had to work with his application to figure out the problems and then fix them." The types of problems addressed by the consultant might relate to fundamental concepts in parallel programming including the need to synchronize code operations through parallel constructs such as *locks* and *wait* functions. Occasionally, the consultants even had to solve serial programming problems such as memory leaks.

The type of help the consultant could provide depended to some extent on the match in the pair's scientific background. As was demonstrated, the consultants were able to provide value to the scientists even when they did not share their scientific background. Such "generic" computational help has some limitations though. For example, there are domains in which some familiarity on the consultant's part with the scientific domain is critical. Many Quantum Chemists, for example, do not write "raw code," but instead use one of the several "packages" that exist in this computationally mature area. The concept of packages in Quantum Chemistry is similar to statistical packages used by social scientists: standard types of analyses are encoded in pre-specified methods which are made available for the scientist to "plug-in" her data. The consultant needs to understand the models behind the various analyses in order to advise the scientist appropriately. <say something about Shaun's expertise>

Even in the case of codes written in MPI or other standard HPC "languages," the generic consultant is limited in the help he can provide. Recall the case of the computationally knowledgeable client who suspected his results though his code ran without error on the center's machines. One of the informants noted that in such cases as a "generic" consultant "You don't know really. The user does." The typically short-term mature of the consultant-client relationship means that the consultant does not have the opportunity to learn enough about the domain to provide domain-specific support. Unlike the team collaboration case described above where everyone learned a great deal about the earthquake domain as a result of the long collaboration, casual collaborations between scientists and computational experts who do not begin with a shared domain expertise is potentially a permanent liability.

4.2.2 Division of Labor

Another potential area for problems that was identified in the casual collaboration related to the expected division of labor between the consultants and their clients. On the surface, the division of labor seems straightforward. One consultant described his job of a *computational science consultant* thus: "What it is, we have researchers who know their work, but do not have the intricate details to program the machines. So we interact with them; we know the machines, they know their work. (We) get the programs to run well." However, cases described by the consultants revealed that the boundary between the roles is open to interpretation. One case concerns a consultant working with a user of one of the Quantum Chemistry packages. The client had little knowledge of the particular package she needed to use, though she was well versed in another of the Quantum Chemistry

[2] MPI is the Message Passing Interface, a library of parallel constructs added to C or Fortran to create parallel code. The number two refers to a newer version of the of the library that is being phased in at the center.

packages. Her code had several problems. The consultant diagnosed one of these as being due to a defect in the version of the package installed at the supercomputing center and he devised a work-around for his client. Based on his expertise in Quantum Chemistry and the package the client was using, the consultant concluded that the other problem was due to the client's lack of understanding of how the package worked. Given her knowledge of another package, he expected her to be able to form a correct understanding of the current package. However, she pushed back, giving him the impression that "she wanted me to do it for her."

Some of the clients seemed to have been setting clear limits with respect to how much computation they were willing to learn. For example, one of the consultants noted that many users are only interested in getting their code to run, and are unwilling to work to make it run *well*. He noted that there are "all kinds of users who are capable but it takes quite a time commitment to do that; but most users are not interested in it. " This sentiment is reminiscent of the feelings expressed by the atomic physicist turned optimizer quoted previously: computation can seem like a distraction from the main work of the scientist.

4.2.3 The Consultant's Influence

Since many of the scientists learn to code through informal means ("on the job"), they may not necessarily learn good programming practices. One consultant noted that "Users are stuck in old practices. One of the problems is that they are stuck in old models." They often fail to learn fundamentals of programming and they fail to take advantage of newer tools that can make them more efficient (like IDE's and newer, higher-level languages). This consultant gave an example concerning the practice of declaring all of one's variables in a single place in one's code. While this is not required by some programming languages (e.g., Fortran), it is a good practice that is especially useful in making it easier in consultant-client collaborations when one has to read the code written by another person. This particular consultant has encountered resistance from some of his clients to this optional practice. Consequently, he started giving his scientific collaborators an ultimatum around declaring all variables in a single place even though (the language they were using in combination with MPI) Fortran does not require it. "I tell users that if you don't use it you are on your own. I keep telling him 'hey you have to put this in; it's good practice."

One interpretation of this example is that it may be possible for the consultants to exert some degree of influence on their clients. Even though the relationship is short-term and thus limited in the amount of cross-influence that can occur, it may be possible for the consultant to have some leverage with the client and thereby have the opportunity to teach him some minimal aspects of good programming practice.

While many of the consultant-client collaborations provide little opportunity for mutual influence because of their brevity, there are some exceptions. Once the codes are completely debugged, the casual collaborations between consultants and their clients are disbanded and the scientist and his code are passed to the Runtime Support group. The Runtime Support group is responsible for ensuring that the codes complete their production runs. As these may take several weeks to months (most codes share the HPC machine at the center with many other users), there is ample opportunity for problems to occur (e.g., a failing file system, defective CPU nodes). The members of the Runtime Support organization are primarily responsible for helping the scientist maneuver these problems.

However, the consultants reported that occasionally the scientists reverted to asking them for help after responsibility for their codes had been transitioned to the Runtime Support staff. One consultant hypothesized that this might indicate that the scientist had gotten comfortable with him and perhaps felt he could rely on him. Especially in the case of the less technically experienced scientists, this could also indicate a reluctance to attempt communication with a person who may not "speak their language." Developing relationships could provide occasions for the consultant to influence the client. This could be potentially fruitful if scientists are repeat users of the services of the supercomputing center. It is the practice of the center to reunite previous consultant-client pairs should the scientist return to the center for another run of their code. While this is primarily motivated by a goal of efficiency, it also provides for continuity in the relationship between the consultant and the scientist.

5. CONCLUSIONS AND FUTURE WORK

These data support the expected differences between the value provided in *team collaborations* and in *casual collaborations*. With longer-term, more closely bound collaboration, there is both more opportunity to develop shared understandings and more reason to make commitments to accommodate each other (2, 7). Thus the circumstances of the team type of collaboration make learning from each other possible. In the particular case of team collaboration that was described, the depth of the expertise sharing that took place enabled fundamental changes in the work of its members. The result of their work was fundamentally a shared creation.

While general programming help from a consultant can mean the difference between a scientist producing running code or not, the lack of shared knowledge places some significant restrictions on the value that can be provided the scientist. Lack of shared knowledge between consultant and client imposes a greater demand on the scientist as he is responsible (or she) for judging the scientific accuracy of his (or her) results. And the limited duration of the collaborative relationship means that opportunities to teach each other will also be limited. Additionally, the time commitment required of the scientist to improve his computational skills is at odds with the need to devote time to his science.

However, the picture for the more casual collaborations is not hopeless. There is some suggestion that the more computationally skilled scientists may be more willing to invest time to add to their knowledge. This was demonstrated by the scientist who instead of depending on the consultant when his code did not work as expected, developed a test program to isolate the problem and help the consultant with the problem diagnosis. The consultants may also have some leverage with their clients who are less inclined to undertake to develop computational expertise. At least in one case, a consultant was willing to set conditions for providing help. While it is not clear form these data what the effect of the attempt to shape the client's behavior was, it may provide an opportunity to encourage learning to take place.

This paper is based on pilot data: It is based on a small number of cases and preliminary discussions with the informants. This will be corrected as this work progresses. A more serious problem is

that only one participant in each collaborative relationship was interviewed. This of course makes it impossible to corroborate the interpretations of events, with the possibility that descriptions are incomplete or incorrect. While I expect that I will be able to correct this in the case of the team collaborations I may not be able to do so in the case of the consultant relationships, at least not with the current participants. The only solution may be find other cases of such relationships where access to the clients will be less restrictive (perhaps internally at my employer's).

6. ACKNOWLEDGEMENTS

This work was in part supported by the HPCS program sponsored by DARPA. I am indebted to the informants who shared their experiences with me.

7. REFERENCES

[1] Bannon, L.J. and Schmidt, K. CSCW: Four Characters in Search of a Context. In Bowers, J.M. and Benford, S.D. (Eds.) *Studies in Computer Supported Cooperative Work,*. Elsevier Science Publishers B.V. (North Holland), 1991.

[2] Bossen, C. The Parameters of Common Information Spaces: the Heterogeneity of Cooperative Work at a Hospital Ward. In *Computer Supported Cooperative Work* (*CSCW '02),* November 16-20, 2002, New Orleans, Louisiana, USA.

[3] Clark, H.H. and Brennan, S.E. Grounding in Communications. In Resnick, L.B., Levine, J.M. and Teasley, S.D. (Eds.) Perspectives on Socially Shared Cognition, APA, 1991, pp. 127-149.

[4] Graham, S.L., Snir, M. and Patterson, C. (Eds.), Getting Up to Speed: The Future of Supercomputing. Available at:

[5] Hindmarsh, J., Haeath, C., von Lehn, D., and Cleverly, J. Creating Assemblies: Aboard the *Ghost Ship*. In *Computer Supported Cooperative Work* (*CSCW '02),* November 16-20, 2002, New Orleans, Louisiana, USA.

[6] Pollock, L. and Jochen, M. Making Parallel Programming Accessible to Inexperienced Programmers through Cooperative Learning. In *ACM Special Interest Group on Computer Science Education (SIGCSE '01),* February, 21-25, 2001, Charlotte, NC.

[7] Olson, G. O. and Olson. J. S. Distance Matters. In Journal of Human Computer Interaction, 15 (2-3), 2000, pp. 139-178.

[8] Big City Shakedown: Inside Lemieux, the 1994 Northridge Earthquake Shakes Almost Like the Real Thing. http://www.psc.edu/science/2003/earthquake/big_city_shakedown.html.

[9] Halverson, C. Inside large scale parallel codes: it's not as different as we thought. (manuscript in preparation).