

INFORMATION SOCIETY TECHNOLOGIES
(IST)
PROGRAMME

Project IST-2001-37057 MKM-NET

Deliverable 4.3:
Conclusions and Research Plan for Formal MKM

Editor: Tudor Jebelean (RISC-Linz)
Authors: Anrzej Trybulec (University of Bialystok)
Andrew Adams (University of Reading)
Fairouz Kamaredine (Heriot Watt University)
Erica Melis (DFKI Saarbruecken)
Dieter Hutter Cristoph Benzmueller (University of the Saarland)
Renaud Rioboo (UPMC Paris 6)
Franz Lichtenberger (SCCH Hageneberg)
Bruno Buchberger, Tudor Jebelean (RISC-Linz)

Project Acronym: MKM-NET
Proposal/Contract no.: IST-2001-37057 MKM-NET

Contents

1 Basic Aspects of Formal MKM	4
2 Retrieval of Formal Mathematical Knowledge	6
2.1 Selection of the relevant knowledge	6
2.2 Mathematical Knowledge Retrieval as Proving	6
3 Research Activities and Objectives	9
3.1 Applications	9
3.2 Mathematical Theories	10
3.3 Knowledge Representation	10
3.4 Tools	11
3.5 Integrating Case Studies	11

Summary

The present document is the result of **Task 4.3: Conclusions and Research Plan for Formal MKM** of the **Work Package 4: Formal Tools in MKM**.

The document represents a synthesis of the suggestions presented by various partners in the project:

- Anrzej Trybulec, *University of Bialystok*
- Andrew Adams, *University of Reading*
- Fairouz Kamaredine, *Heriot Watt University*
- Erica Melis, *DFKI Saarbruecken*
- Dieter Hutter, Cristoph Benzmueller, *University of Saarlandes*
- Renaud Rioboo, *UPMC Paris 6*
- Franz Lichtenberger, *SCCH Hageneberg*
- Bruno Buchberger, Tudor Jebelean, *RISC-Linz*

The complete report has been integrated by Tudor Jebelean, RISC-Linz.

1 Basic Aspects of Formal MKM

Mathematics comprises a continuously growing body of knowledge which is useful for and used for various activities, that we will call “*applications*”. By Mathematical Knowledge Management (*MKM*) we understand the techniques for storage, retrieval, and presentation of this knowledge in such a way that it can be effectively and efficiently used in applications. (These applications include the activities directed to further growth of the mathematical knowledge, as well as to the further advancement of *MKM*).

The activities related to Mathematical Knowledge Management have various aspects, which are relevant to various fields in Mathematics and Computer Science. In this report we concentrate on the particular aspects related to the Management of Formal Mathematical Knowledge.

Informal Mathematical Knowledge. Mathematics is traditionally presented in textbooks and research papers, and these can be also considered Mathematical Knowledge Bases. The correct interpretation of such knowledge can be performed only by very well educated human experts (typically mathematicians and engineers). This is mainly due to incompleteness and ambiguities of the text and of the context. Informal texts do not specify completely the details (*e.g.* of proofs) and the contextual theory. The part of the text which is expressed in natural human language suffers from the ambiguities inherently present in this language. Both this part and the part which is expressed in mathematical notations (“*formulae*”) are correct only in the context of a certain mathematical theory. Human readers of informal mathematics are able (most of the time) to complete the text and to disambiguate it, by using their internal mathematical knowledge base (obtained through long and intensive education). However, both human writers and human readers of mathematical texts make their own errors when writing and reading mathematics.

Therefore, the management of mathematical knowledge in the traditional informal style is both slow (lack of efficiency) and error prone (lack of effectiveness). The automation of the Management of Formal Mathematical Knowledge aims at solving these problems, however it must not ignore the progress achieved during many centuries of mathematical development. We believe that an important source of “*inspiration*” for the representation and the techniques of formal *MKM* is the study of the ways in which mathematics is done by human experts.

Formal Mathematical Knowledge. In order to increase the efficiency and effectiveness of *MKM*, we advocate the use of computers, computer software, and advanced algorithms. More specifically, formal mathematical knowledge should be presented in such a way that it can effectively and efficiently be used by software tools, with minimal intervention from the human expert. This means that formal *MK* has to be complete and non-ambiguous, both with respect to the text (thus it has to be expressed in a *formal* language) and to the context (thus it has to have a highly organized structure). We do not advocate the complete elimination of the human factor from formal *MKM*, or from the use of *MK* in applications. We believe that by automating more and more of the *MKM* activities, the human experts will be more and more relieved from routine tasks, and will be able to use their creative and intelligent capabilities at a higher level, thus more efficiently.

Knowledge Management. The notions which are essential to knowledge management are highlighted in the following scenario: An *application* generates a *query* to the knowledge base, whose *retrieval* mechanism produces an *answer* which is consumed by the application. Human intervention may occur in any of these activities, thus *user-interaction* facilities and *natural presentation* of the information are important. The knowledge is presented in a certain *language* and has a certain *organization* in the knowledge base.

Formal MKM in Applications. Since formal *MKM* addresses the use of mathematical knowledge in applications, it is important to specify the nature of the notions highlighted above in the context of concrete applications which need mathematical knowledge. That is, we need to investigate what kind of queries/answers are generated/consumed by such applications. Typical applications are: problem solving (retrieving algorithms or

properties of mathematical models), automated reasoning (proving theorems), formal verification of programs or systems, algorithm synthesis, computer aided teaching of mathematics.

From the experiments carried out in this project, it appears that both the queries and the answers are (abstractly) collections of mathematical formulae (or theories), while the answers may also contain mathematical proofs.

Contents and Representation of Formal MK. Mathematical knowledge consists of formulae grouped in theories, mathematical proofs, mathematical algorithms, algorithm schemata (such as *e.g.* “divide and conquer”). Note that “mathematical algorithms” may also include logical inference systems and proof methods, and also that “algorithm schemata” may come together with a proof method for the purpose of proving correctness. All these can be essentially presented as organized collections of formulae.

The historical mathematical experience, as well as the experience of the present project, suggest that the most appropriate representation of Mathematical knowledge is (abstractly) the language of mathematical logic (higher order predicate logic), and the organization of knowledge should follow a hierarchical approach (mathematical theories). This is consistent with the form of the queries and the answers, which are of the same type.

These considerations still leave open various questions regarding:

- The use of different logical systems and of concrete syntax for formulae,
- Concrete computer representation and translation between various formats,
- Concrete organization of theories and of links between them.

This questions are also influenced by the dynamical character of the mathematical knowledge.

Retrieval in Formal MK Bases. The mathematical experience and the work in the present project suggest that, essentially, retrieval from a knowledge base of formal mathematics is the process of finding the proof of a mathematical statement (formula). This is consistent both with the previously proposed content of queries and of answers.

This leaves open the matter of the logical system that is used, the proof method[s] and the particular proving tool[s] used, and also the problem of appropriately selecting the set of assumptions (theories) that the proof should use.

This also suggests that most of the techniques already developed for retrieval in other types of knowledge bases (or data-bases) are of little use in the context of formal MKM. However, some techniques (*e.g.* keyword search, pattern matching) may be applicable in a certain limited number of situations.

User-Interaction in Formal MKM. Although this research aims at automating MKM to a high degree, it will probably be impossible to obtain good solutions in “real-life” problems without the assistance of human experts in all phases of knowledge management: query formulation, knowledge retrieval, and interpretation of answers. Moreover, certain applications (*e.g.* computer aided learning) are essentially dependent on user interactions. This raises the problem of appropriate presentation (in a form which is easily understood by humans) of the mathematical knowledge and of the processes which take place during knowledge retrieval (*e.g.* proofs). This problem has a social aspect too: different groups of users have different levels of formal training.

2 Retrieval of Formal Mathematical Knowledge

From our experiments we conclude that retrieval of formal mathematical knowledge has two essential aspects:

- Selection of the appropriate elements of the knowledge base;
- Subsequent proof using the elements extracted.

The nature of these activities have important consequences for the contents and organization of mathematical knowledge bases.

2.1 Selection of the relevant knowledge

Traditional presentations of mathematical knowledge have a hierarchical structure. For instance, when proving the properties of convergent sequences, one uses properties of real numbers; when studying differentiation, one uses the knowledge about limits; etc. Similarly, when we study sorting algorithms, we use properties of lists and of integers.

Likewise, formal mathematical knowledge bases must be organized in such a way that each application can clearly specify what part of the knowledge base is most likely to be useful for solving the queries generated by that application. The work presented in [11, 10] shows how to organize a database of mathematical knowledge for use in a computer-aided learning application, which has specific requirements about the retrieval process.

An aspect which is specific to the organization of formal mathematical knowledge is heterarchy and multiple inheritance. For instance, reals form a set, as well as a group with respect to addition, *etc.*. Thus, when using knowledge about reals, we also need to use knowledge about sets, groups, *etc.*

In the FoC [9, 12] project, a large knowledge base about computer algebra algorithms and their properties is organized hierarchically, using a (multiple) inheritance mechanism. The MIZAR project developed a large collection of mathematical knowledge which is organized hierarchically starting from the simplest theories. However, currently there are very many conceptual and technical approaches to this problem (see *e.g.* [1] for a survey of various organizational styles for just one particular theory), thus an integration effort is necessary.

The experiments presented in [5] show that it is possible to organise formal mathematical knowledge using the mathematical formalism provided by category theory (functors). The advantages of this approach consist in having available the power of mathematical logic (inferences) also when working with hierarchically defined theories.

An important activity related to the overall organization of mathematical knowledge is the management of change. Since mathematical knowledge is highly integrated (many parts of it depend on other parts), any change in one part can trigger inconsistencies in other parts. A good MKM engine must be able to insure the coherence of the whole knowledge base, by following all the “links”. This is, in fact, one activity in which computer tools may bring significant improvements, since it is one of those “routine” types of work which is less attractive to and less appropriate for human experts. Mathematical knowledge is very dynamic, both at the global and the local level.

At the global level, new knowledge is produced, and older knowledge is often modified (or studied from different points of view). At the local level of each application (and in fact of each problem), the knowledge also changes (*e.g.* by debugging a system and its specification). Therefore, it is not realistic to conceive of a mathematical knowledge base as a frozen object (unlike *e.g.* a geographical data base). Rather, MKM engines must offer frames in which different views of mathematical theories can be represented.

The experiments described in [2, 3] show how the MAYA system manages change in a mathematical knowledge base dedicated to supporting formal verification of systems and software.

2.2 Mathematical Knowledge Retrieval as Proving

In most other types of knowledge bases (or data bases), retrieval is performed by using tables which show relations among different objects, and by using pattern matching queries (whose simplest version is keyword search). While these techniques may be useful for the selection of the relevant knowledge (see previous subsection), they cannot be used for full retrieval, because this needs logical inference.

Most of the time, the query (formulas to be “found” in the knowledge base) will not be present in the database in exactly the same form as the query.

This is due, *on the one hand*, to the fact that the same mathematical knowledge can be presented in different syntactic formats. The simplest example is the invariance of the mathematical formulae under the change of the names of the bound variables (e.g. $x + y = y + x$ is the same as $n + m = m + n$, because x, y, n, m are universally quantified). However, in typical situations more complicated cases of retrieval occur. A further problem in the concrete representation of mathematical facts is the use of different formalisms: some authors prefer the full encoding in predicate logic, while other incline towards a more “human like” encoding – as e.g. in our project [8].

On the other hand, most cases of retrieval involve a formula that does not have a syntactic equivalent in the knowledge base. In general, one needs to *prove* that the query is a logical consequence of the formulae present in the knowledge base. However, not all proving problems should be seen as mathematical knowledge retrieval. Rather, retrieval should be seen as performing “simple proofs”.

The difference between “retrieval” (as “simple proving”) and “proving” (as more “complex proving”) is not yet fully defined, one needs to perform more research and experiments. However, this difference can be exhibited in the context of some examples. Consider, for instance, that we use a knowledge base which contains the theory of rationals, and we are investigating the notion of “limit of sequences”, starting from the classical definition with alternating quantifiers. A proof of the theorem:

$$\lim(f + g) = \lim(f) + \lim(g)$$

starting from the definition involves complex inference steps consisting in “inventing” (or searching for) appropriate instantiations of some quantified formulae. It is probably unrealistic to expect a MKM engine to perform such kinds of proofs (for instance because the search space will be prohibitively large). However, if this theorem is added to the knowledge base and later we try to prove the lemma:

$$\lim(f + g + h) = \lim(f) + \lim(g) + \lim(h),$$

then this proof will be a straightforward application of some rewrite steps. This latter proof can be classified as “knowledge retrieval”. One may also see here some implications with respect to the contents of mathematical knowledge base:

- It is reasonable to store properties relevant to simple expressions like $\lim(f + g)$, $\lim(f * g)$, $\lim(f / g)$, . . . , but
- It is not reasonable to store properties referring to compound expressions like $\lim(f + g + h)$, $\lim(f * g + h)$, $\lim(h * f / g)$,

This is so because, by storing all the properties of the simple expressions (which can only be obtained by “complex” proving), one can later obtain “simple” proofs for any of the other expressions.

This example suggests the following principles for the contents of and retrieval from mathematical knowledge bases:

- Retrieval consists in performing “simple proofs”;
- The knowledge base (theory) has to contain only those formulae which allow “simple” proving for all the other ones.

In fact, this is the way in which mathematical theories have traditionally been built.

The case studies realised in the **THEOREM** system and presented in Deliverable 4.2 and in more detail in [6, 7] give a systematic discussion and classification of the various situations in which retrieval and proving may overlap or not in the context of an algorithm retrieval scenario. The paper [4] presents the principles of systematic build-up of theories using the principle of “theory exploration”.

In the context of our project, the relationship between proving and knowledge retrieval is also studied in the experiments performed by the Ω MEGA group [13, 14], which is currently developing an engine for the interface between various provers and various mathematical knowledge bases.

The activity in the framework of work package 4 of the present MKM project has significantly developed our vision of what formal MKM is and how it can be done. However, the precise definition of the notion

of “retrieval of formal mathematical knowledge” as well as the precise recipe for building the contents of “formal mathematical knowledge bases”, if ever such precision will be possible, are yet to be found through more research and experiments.

References

- [1] A. Adams. General formalizations of real and complex analysis. Section of the MKM Net deliverable 4.2.
- [2] S. Autexier, D. Hutter, T. Mossakowski, and A. Schairer. The Development Graph Manager MAYA. In Proceedings 9th International Conference on Algebraic Methodology And Software Technology, AMAST2002, Springer-Verlag, LNCS 2422, pp.495–501, 2002.
- [3] S. Autexier and D. Hutter. Maintenance of Formal Software Developments by Stratified Verification. In Proceedings 9th International Conference on Logic for Programming Artificial Intelligence and Reasoning, Springer-Verlag, LNAI 2514, pp36–52, 2002.
- [4] B. Buchberger. Theory Exploration with Theorema. Analele Universitatii Din Timisoara, Ser. Matematica-Informatica, Vol. XXXVIII, Fasc.2, 2000, (Proceedings of SYNASC 2000, 2nd International Workshop on Symbolic and Numeric Algorithms in Scientific Computing, Oct. 4-6, 2000, Timisoara, Romania, T. Jebelean, V. Negru, A. Popovici eds.), pp. 9-32.
- [5] B. Buchberger. Groebner Rings in Theorema: A Case Study in Functors and Categories. Technical Report SFB 2003-26, Special Research Area ”Scientific Computing”, Research Institute for Symbolic Computation, Johannes Kepler University, Linz, Austria, November 2003.
- [6] B. Buchberger. Algorithm Retrieval: Concept Clarification and Case Study in Theorema. Technical Report SFB 2003-44, Special Research Area ”Scientific Computing”, Research Institute for Symbolic Computation, Johannes Kepler University, Linz, Austria, November 2003.
- [7] B. Buchberger. Algorithm Invention and Verification by Lazy Thinking. In: D. Petcu, V. Negru, D. Zaharie, T. Jebelean (eds), Proceedings of SYNASC 2003 (Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, October 1-4, 2003), Mirton Publishing, ISBN 973-661-104-3, pp. 2-26.
- [8] Fairouz Kamareddine, Manuel Maarek, and Joe Wells. MathLang: experience-driven development of a new mathematical language. , to appear in ENTCS 2004. Can also be downloaded from authors’ web pages.
- [9] The FoC project homepage: <http://www-spi.lip6.fr/foc/>
- [10] George Goguadze. Data Structure for Annotated Mathematical Documents. *MoWGLI Workshop*, December 16-17, 2002, Saarbruecken.
- [11] E. Melis, G. Goguadze, C. Ullrich, P. Cairns Problems and Solutions for Markup for Mathematical Examples and Exercises. In: A. Asperti, B. Buchberger, and J. H. Davenport, editors. *MKM03: Mathematical Knowledge Management*. Springer-Verlag LNCS 2594, pp80–92, 2003.
- [12] Renaud Rioboo. Towards faster real algebraic numbers. *Journal Of symbolic Computation*, 36:513–533, 2003.
- [13] Jörg Siekmann, Christoph Benzmüller, Armin Fiedler, Andreas Meier, Immanuel Normann, and Martin Pollet. Proof development in OMEGA: The irrationality of square root of 2. In Fairouz Kamareddine, editor, *Thirty Five Years of Automating Mathematics*, Kluwer Applied Logic series. Kluwer Academic Publishers, 2003. In Print.
- [14] Quoc Bao Vo, Christoph Benzmüller, and Serge Autexier. An approach to assertion application via generalized resolution. In *International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 1343–1344. IJCAI/Morgan Kaufmann, 2003.

3 Research Activities and Objectives

3.1 Applications

Since the main purpose of the management of formal mathematical knowledge is to make mathematical knowledge more usable for human activities, it is very important that all aspects of formal MKM are studied in the context of concrete applications.

During our research in the MKM project, we identified the following main applications which appear to benefit most from the automatic use of formal MK:

- Automated reasoning (proving theorems).
- Algorithm retrieval and synthesis.
- Formal verification of programs or systems.
- Computer aided teaching of mathematics.

Automated reasoning (proving theorems). The traditional approach to automated reasoning is to attempt to prove a goal formula on the basis of several assumed formulae, when it is *known* that the goal is a logical consequence of the assumptions. Human experts, however, do not proceed like this. Rather, they “discover” the necessary assumptions during the development of the proof. “Discovery” means in fact that the necessary assumptions are retrieved from the “internal” mathematical knowledge base of the human expert, and maybe from other external sources (textbooks, etc.).

A proving system assisted by a knowledge base may proceed by sending a query to the retrieval engine whenever it has to prove a goal. If the goal is not “retrieved” (that is: it is not “easily” provable from the knowledge base), then the prover has to decompose the goal further into simpler subgoals, which again are submitted to the same procedure.

Algorithm retrieval and synthesis. The problem of algorithm retrieval/synthesis essentially reduces to finding a function which satisfies a given specification (which is a logical formula).

If the definition of the function exists in the knowledge base then we can say that the algorithm is “retrieved”. A good knowledge base would also contain in this case enough properties about the retrieved function such that the proof of the specification is “easy”, and thus is provided by the retrieval engine. Otherwise, one must involve an appropriate prover in order to prove the specification: this is program verification (see next item).

If the knowledge base does not contain any function which satisfies the specification, then we are in a situation of “algorithm synthesis”: the algorithm synthesis method will have to refine (decompose) the definition of the function into further sub-functions (for instance using an appropriate algorithm “schemata”), and then it will try again to retrieve these new sub-functions from the knowledge base.

Formal verification of programs or systems. This application builds on top of the previous two applications. Namely, program verification is the attempt to prove that a given function satisfies its specification (a logical formula), thus it is basically theorem proving. However, in the context of a concrete engineering activity, proving in this case has a special flavour, because both programs and specifications are not in their final form when we attempt to verify them. Thus, one cannot expect that the proof will actually work from the very beginning. Rather one has to organize the whole application in such a way that failing proofs provide the human expert with useful information for debugging both the program and the specification.

This application is very important for the future use mathematical knowledge in human activities, because formal verification becomes an essential part of engineering activity in the context of realization of more and more complex technological systems.

Computer aided teaching of mathematics. This application area consists in intelligent tutoring in the field of mathematics, and it relates mainly to automated reasoning (presenting proofs to students and checking student proofs), but also to algorithmic knowledge (presenting proof methods and algorithms, checking their

correct application by students). A particular feature of this application is the necessity for the presentation of mathematical knowledge in a tutorial fashion, which has important consequences for the way in which the formal mathematical knowledge is represented and organized, as well as on the requirements for the retrieval mechanism. For instance, the knowledge which is used at a certain moment of the learning process has to be part of the knowledge which is assumed to have been mastered by the student at that time.

The research should develop case studies of significant size, in which formal mathematical knowledge is used in (possibly) real-life applications.

3.2 Mathematical Theories

Ideally “all” mathematical knowledge should be integrated in one or more mathematical knowledge base(s), however this is probably unrealistic now from the point of view of the effort, and even unrealistic at any time in the future, since global mathematical knowledge is very dynamic.

Instead, the research should try to realize concrete knowledge bases for those area of mathematics which are most useful in applications, and on this basis refine and improve the concept of formal MKM and the necessary tools such that it will offer a better and better framework for future larger collections of mathematical knowledge.

The activity in the MKM project revealed the following areas as being most useful for further development and experimentation:

- Mathematical analysis (calculus), functional analysis, Hilbert space theory (for computed aided teaching, automated reasoning, development of numerical algorithms).
- Computer algebra – including Groebner Bases, subresultant computations (for computed aided teaching, automated reasoning, retrieval and verification of symbolic algorithms).
- Basic algorithmic domains – sets, lists, integers (for computed aided teaching, automated reasoning, retrieval and synthesis of algorithms).

The research should make use of existing repositories of such knowledge, by further developing and integrating them into coherent knowledge bases which are usable in applications.

3.3 Knowledge Representation

The representation of formal mathematical knowledge has two basic conceptual aspects:

- The representation of the individual items (logical formulae or formula patterns, algorithm schemata, etc.);
- The representation of the overall organization of the knowledge: structuring into theories and sub-theories, multiple inheritance, dependences between various parts, etc.

Current repositories of mathematical facts use numerous incompatible representation schemes. However, traditional mathematics is presented in the universal language of higher-order predicate logic, which has been used successfully over a long period of time in order to store and communicate mathematical knowledge.

Moreover, even systems which use similar conceptual mechanisms are significantly different in their concrete encoding of mathematical knowledge. During the last few years a significant effort has been made in order to standardise the syntactical representation of mathematics (MathML) and promote the use of a standard (OMDoc). However, the current standards do not yet offer a framework for the representation of complex logical formulae or mathematical theories.

The research in this area should address three main goals:

- Finding the most appropriate conceptual representations, from the point of view of their usability in concrete applications.
- Integrating the conceptual representations such that they are at least compatible (if not unique), in the sense that semantic-preserving translation between different representations is easily possible.
- Surveying, finding, adapting, and using the most appropriate standards for the encoding of mathematical information.

3.4 Tools

The management of formal mathematical knowledge is based on a few “core” tools:

- Retrieval (find proof of the goal in a certain theory),
- Theory selection (find the theory or theories relevant to the query),
- Database maintenance (integrity, consistency, dependencies),
- Input–output and import–export routines.

Moreover, the management of formal mathematical knowledge, especially if it is embedded in applications, needs the use of additional tools:

- Automatic theorem provers, in particular “natural style” provers;
- Tools for program (system) analysis, verification, and synthesis;
- Computer algebra programs;
- Editors for mathematical texts;
- Internet communication engines;
- Database engines;
- Translators between different representation formats;
- User-interface and presentation tools for mathematical information.

The development of such additional tools is the object of whole research areas in Computer Science and Mathematics, and thus should not be in the centre of the research done for formal MKM. However, the research and experiments performed for formal MKM should provide useful input and new research directions for these other areas.

The goals of the activity in formal MKM should be continuously to survey and influence the development of such tools, to find the ones which are most useful for MKM, and possibly to adapt the most appropriate ones in order to use them as “core” tools. For instance, the basic retrieval engine may be a “basic prover” which is able to perform only “simple” inference steps, such as the matching of ground formulae (and terms) against universally quantified ones, propositional inferences, and rewriting using equality.

An important goal of the MKM activity should be to achieve as much as possible compatibility, integration, standardisation and automatic cooperation between the various tools used for formal MKM, and also for the processing of mathematical information in general.

3.5 Integrating Case Studies

The activity in the current project consisted mainly in developing various case studies at different sites and using different concepts and tools. The main interaction was between the participating researchers and at the conceptual level, but there was little interaction between different systems, and also there was almost no overlapping in the subjects of the case studies. This makes it almost impossible to compare the advantages and disadvantages of the various concepts, approaches, systems, and tools.

In order to make further progress, in the next phase of the research it is necessary to develop:

- More comprehensive case studies by increased cooperation between different sites and different platforms,
- Case studies with similar goals but on conceptually and/or technically different platforms, exhibiting compatibility and translatability.

The goal of this research is to increase the possibility of cooperation between different groups, but also to reveal the advantages and disadvantages of certain conceptual and technical approaches.