FMI and IP Protection of Models: A Survey of Use Cases and Support in the Standard

Erik Durling[*] Elias Palmkvist Maria Henningsson[*]

[*]Modelon AB, Sweden, Corresponding author: erik.durling@modelon.com

Abstract

FMI is increasingly being adopted as a standard for exchanging simulation models within and between organizations. Such models often represent significant investments for the model creator. There is thus a large interest in protecting intellectual property while collaborating and sharing simulation models in the form of FMUs. This paper presents a collection of use cases and issues related to IP protection of model contents, that have been identified in interviews with industrial representatives. The requirements in each use case are described, along with an investigation of how well the use cases can be managed within the current version of the FMI standard, including a proposed extension of the standard.

Keywords: *FMI, IP protection, model exchange*

1 Introduction

The promise of major benefits in model-based systems engineering and virtual development lies in reusing models in different contexts. To develop, parameterize, validate, and maintain models represent a significant investment, and to maximize the return the models need to be utilized as much as possible.

FMI is becoming the de facto industry standard for exchanging models between different tools. Two main directions in the FMI domain is currently integration and democratization. Integration means software, processes, and standards for co-simulation of multiple models from different tools or different organization. Democratization means effort to spread the usage of advanced simulation models for experts using expert tools to much larger groups of engineers to use for design space exploration, boundary conditions for other systems, or software development and testing.

Both these directions involve exposing models that often represent significant investments and contain sensitive data to a larger user base within and outside of the original organization. The question about protecting IP (Intellectual Property) is often raised in discussions about exchanging models between partners with commercial interests.

Although there exist solutions and best practices for sharing models with existing technologies, FMI is still a new standard, and there is a general need for knowledge about applying similar solutions with FMI (Köhler et al. 2016). One of the arguments for using FMI is that it allows protecting the internal contents of models. But it is important for the part sharing a model to understand what is exposed, and what measures that can be taken to protect what should not be shared.

The purpose of this study has been to make an inventory of use cases and concerns related to IP protection of FMUs, and to evaluate to what extent this is supported by the current standard. This overview can be of interest for users who need to understand the risks and mechanisms for exposing and protecting the content of their models.

The study also intends to raise the need for a standardized way of managing IP protection mechanisms of FMUs, or at least to provide information to the model importer about embedded mechanisms to restrict execution of the model.

The paper starts by outlining how the listed use cases were elicited. The list of use cases is presented in Section 3. An evaluation of how well the use cases are supported by the current (2.0) FMI standard is found in Section 4.

2 Methodology

The study was carried out in two phases. During the first phase, information was gathered about the needs that exist for protecting IP when sharing models within the general area of model based systems engineering.

Interviews were carried out with 16 engineers at Modelon and Volvo Car Group, with experience of sharing models within automotive, energy and aerospace industries. The interviews were typically with a single person at a time, and lasted in the range of 30 to 60 minutes. To obtain unbiased information, open questions were asked to let the stakeholder present their own view of the issues they found important. The questions concerned exchanging models in general, and not specific to the FMI standard.

In addition to the interviews, an anonymous online survey was sent out to additional external stakeholders. The purpose of this survey was mainly to obtain an impression of the importance and priorities among the identified use-cases. The survey also included specific questions to identify experience and concerns specific to the FMI standard.

During the second phase, the FMI-standard was evaluated in terms of each of the use cases that had been identified. The following questions were considered when evaluating the standard:

- Is the use case relevant to be considered within the scope of the standard?
- Is there any support for the use case within the standard?
- Are there any obstacles or gaps within the standard that prevents solutions for the use case from being implemented?

3 Use Cases: Needs for Protecting IP When Sharing Models

This section describes the different use-cases that have been identified in this study, following a short summary of the roles involved.

The question of IP protection is typically considered when models are to be shared between different organizations with commercial interests. The purpose is to protect valuable or sensitive knowledge or data from being accessed by someone who is not trusted. The need for protection generally comes from the part sharing (exporting) the model, but there are issues related to this that may affect the receiver (importer) of the model.

A common scenario is that a component supplier delivers a component model to an OEM (Original Equipment Manufactory). This component model is integrated by the OEM as part of a system model. The opposite also occur, where the OEM supplies a system model, to let the supplier test their component as part of a system environment.

There are also situations where there is a need to protect models that are shared inside the same organization. Reasons for this can be to maintain control over what models are being used in the organization. Another reason can be to prevent potential leaks by limiting access to sensitive information. This situation could also apply during projects with external partners, where the information is not secret to the people in the project, but there is a need to protect the information from being shared outside the project.

In general, the main concerns regard export of models. This mainly covers two main issues: hiding the model content, and controlling who can use the model. But protecting the models can also lead to challenges for the receiver of the model, in terms of usability, that need to be considered.

3.1 Use Case 1: Protect Model Contents

The basic use case is that the part who shares a model would like to hide what is inside from the receiver. The sharing part needs to export the model in such a way that the contents are protected. There are a number of aspects that may be valuable or sensitive and needed to be protected.

3.1.1 Model Structure

There is often a need to protect the structure or design of the model. This consists of equations and algorithms that describe the relationship between inputs and outputs.

The model may be implemented using unique methods for describing the specific component. Examples of this could be algorithms, or representations of equations, or clever ways to select dynamic states. This can make the model design valuable in itself.

The model could also represent unique knowledge about the component that is modeled, and could reveal sensitive information about the actual component design and properties.

Some models might be created to support multiple application. In this case, information about the other types of application that is supported might be sensitive. It could be that the receiver should only have access to information that concerns their specific application.

3.1.2 Internal Variables

Internal variables (or ”signals”) may reveal sensitive information about the inner workings of a model, and could facilitate reverse engineering.

The names of the internal variables could also be sensitive and reveal information about the model structure and design, or ways to apply the model that the receiver should not be aware of.

3.1.3 Parameters

Values of internal design parameters, boundary conditions and start values, may reveal information that would not be available to a user of the actual component or system that the model represents. This data may be the result of expensive research, and considered valuable knowhow that a supplier is reluctant to share.

Parameter names may also reveal information about model structure, in the same way as internal variables. It could also be that the parameter values are only sensitive with a specific parameter name. Generic parameter names may not reveal any useful IP.

3.1.4 Black Box or Grey Box

The simplest approach is to hide everything inside the model (black box), which may be sufficient in some cases. However, in many cases it is necessary to expose parts of the model contents (grey box), in order to make the model usable. In this case, the exporter typically would like to expose only the sub-set of the content that is necessary for the receiver to have access to.

For example, exposing part of the model structure or internal variables could aid in simulation debugging. And some parameters may need to be tweaked in order to use the same model for multiple scenarios.

3.1.5 External Dependencies

A model could contain external dependencies, for example parameter files or additional model libraries. These parts could contain IP that may not be covered by the protection applied to the main model, and may require specific consideration.

3.1.6 Reverse Engineering

Sharing a model always comes with a risk of reverse engineering, either of the model itself or the component that the model represents. The only way to be completely protected against this is to not share any model. The required level of protection against this depends on the value of the model contents and the risk of the contents being revealed. A common strategy of handling this is to make sure that the cost of reverse engineering is higher than the value of the contents.

3.2 Use Case 2: Limit Access to Users

A common scenario is that only a set of expected users should have access to a model, for example to maintain control or prevent reverse engineering. A model could contain information that should not fall into the wrong hands, or be used for applications other than the exporter’s intention.

3.2.1 Limit Access to Specific User(s)

There are many scenarios where a model is only meant to be shared with a limited number of users, or different users should have different level of accessibility to the model. It is common that the right to use a model is given to a single organization by a partner. In sensitive cases, some models may even be restricted to specific groups within an organization, to minimize the risk that it ends up in the hands of the wrong people, for example a competitor. There are also commercial scenarios. For example, a model library may be sold for use on a single computer only.

Models exported from some tools may be restricted to users who have a license for the exporting tool. Such limitations may represent a big obstacle for some scenarios of model sharing. It may not be feasible for users integrating models from many different sources to have a license for all the tools. This could also be a problem when an exported model need to be deployed to a large group of end-users, since the licensing fee would become unreasonable. Some OEMs have also expressed concerns that licensing solutions on exported models could lead to vendor lock-in.

3.2.2 Limit the Model Over Time

There are also scenarios where one would like to limit the model access to a specific time frame. A reason could be that the model could contain information or be used for applications that is only relevant during a limited time, and the use of the model may even be contracted between the two partners. This could for example be during the course of a specific project, or during a trial period of a commercial model library.

Having a time limitation on a model could also be a benefit when it is being developed and need to be maintained over time, since it reduces the risk that an old version of the model is used.

The time frame could differ depending on the use case, from a couple of days for a sales demonstration, a few months between model release versions, or during a project that last for years.

3.2.3 Information About the Protection

Models with limited access pose a challenge from the model receiver’s perspective. Without sufficient information about what type of protection is applied, debugging could be difficult when the user or the importing tool should identify that the model is not working due to this protection.

This is especially important for a user working with aggregates of models from multiple model suppliers, where there may be number of different types of access limitations that need to be managed. The workflow for these users are improved if it is possible to easily understand how each model is restricted and what is required for getting access to it.

3.3 Use Case 3: Provide Information to the Model Importer

When parts of a model are hidden, or protected, there is an increased need for information to keep the model useful.

3.3.1 Documentation

For a model with hidden contents, the user must rely on the documentation for information on how to use the model and what results to expect. It can be crucial to understand what aspects of the physical systems are modeled and at what degree of accuracy, especially when integrating the model as part of a larger system, or to understand simulation results. This could dictate what parts are needed outside the model and how to interpret the interface. It may also be difficult to determine what range of operating conditions the model is valid for, since it likely is not obvious what simplifications or assumptions have been made. In general, it is important that both parts have agreed on the interface of the model inputs and outputs.

3.3.2 Debugging

It can be very challenging to debug a model without knowledge about how it is constructed, without the ability to measure internal variables or to get usable error messages. This can be a challenge also when using the model as part of a larger system. The user may have to rely on support from the model supplier for solving the issues. This could also pose a challenge for OEMs that need to be able to trace issues found in simulation results back to the source model, many years after the results were produced.

3.3.3 Network Dependencies

Some protection solutions may require the model to communicate with remote network resources, for example to gain access to using the model or to exchange results with a simulation server. Information about these dependencies and adequate error messages can be important for helping the user identify any issues related to this.

3.4 Use Case 4: Binary Platform Support and Source Code

There is a conflict between protecting a model from reverse engineering while at the same time allowing the model to be used on multiple platforms (different operating systems or processor hardware). Exporting a model in a compiled binary format is a common way to protect the sensitive content. However, this will limit the model to the specific platform that the binary is compiled for. To support multiple platforms, the solution is often to export the model as code (commonly C-code) and let the receiver compile the model on the specific platform. While binary export is often considered sufficient protection against reverse engineering, c-code is generally not considered sufficient, since this is more easily interpreted by a human. Solutions for this could place requirements both on the exporting and importing tools.

It is important to note however, that the content of a binary also can be interpreted, while the effort to do so is generally much higher than doing this for higher level source code.

3.5 Knowledge Need

A general need for knowledge about the IP-risks specific to FMI was identified during this survey. When exporting a protected model that contains IP, it is important for the exporter to understand what is exposed when the model is exported, and what risks may need to be avoided. This will help making correct decisions about what measures need to be taken, but is also necessary for the exporter to feel trust in the solution used.

It is worth noting that new technologies have a start-up phase in general, where potential users will be naturally skeptical before information about the technology is widely known, and best practices have been established.

It may also be important also for people that are only working indirectly with models understand how the risks are handled. For example, a lack of knowledge about the technology could represent an obstacle in and negotiations about sharing models between partners. A wider acceptance may be needed among all affected parts of an organization before it is regarded as safe.

3.6 Use Case 5: Authentication

Authentication concerns the need to ensure the integrity of a model. This question is not mainly about hiding content, but instead of protecting it from being changed. Although, it is sometimes discussed in relation to IP protection, since the challenges is somewhat related.

Some of the common needs are:

- Verifying that the model comes from the expected source.
- Verify that the model has not been altered after it was exported. This could be important in order to provide reliable support as a model supplier, or when the model is deployed in safety critical systems.
- Verify that the model is compatible with some external dependencies, for example that the specific version of a model is used together with the corresponding version of parameter data.

Authentication plays a role both as a sanity check to avoid mistakes, but also as a means of protecting against intentional intrusion.

4 Support for Protecting IP Within the FMI Standard

This section describes how and to what degree the use cases described in Section 3 are supported by the FMI-standard. Some examples are used to demonstrate how the standard supports certain use cases.

4.1 The Content of an FMU

An implementation following the FMI-standard is called an FMU. This section describes what parts of a model is exposed when being packaged as an FMU. An FMU is a zip-file, with a certain file structure, that contains the following parts:

- The model description XML-file:

  Contains meta information about the model that will be exposed to the simulation tool and user.
- Binaries:

  This is the actual implementation of the model, compiled for a specific (or multiple) target platform. This binary exposes the standard FMI API functions, for reading and writing variables and performing simulation time steps.
- Source code:

  C-source code for the model can be provided as an alternative, or in addition, to a model binary.
- Additional data/resources:

  This could be data stored in any format as a resource in the FMU. This would typically be parameter data. It is also possible for an FMU to access external resources outside of the FMU itself.

The FMI standard specifies the format of the model description XML-file, the API function interface of the binaries or source code, and the structure of the zip-file.

The FMI-standard allows two different type of models, one that contains a solver to simulate the model (Co-Simulation or CS-FMU) and one that requires an external solver to simulate (Model-Exchange or ME-FMU). The functions and exposed content differ somewhat between the two FMU flavors.

4.1.1 Content in the Model Description XML File

The model description XML-file contains necessary meta information to the user and simulation tool, in order to make the model useful.

The information required to be included is the type of FMU, name of the model, and a GUID (Global Unique Identifier).

In addition to this, XML-file is required to contain tags for model variables and model structure, which will contain a set of variables that are exposed. But there is no requirement from the standard that all model content, in terms of variable names or values, should be exposed in the model description XML-file.

In practice, at least the top level input and output variables are exposed. The model description could also contain references to all, or a subset, of the internal model variables and parameters. But it is up to the exporting tool whether all variables should be exposed, or none (black box) or a sub-set (grey box), and also what names to give the variables.

For the exporting user, it could be very helpful to get clear information from the exporting tool about what variables are exposed. Although this information is available in the xml-file, it can be very impractical to obtain the information by reading the file directly, especially for large models.

The variables defined in the model description file is a mapping between variable names and variable references. The variable reference (a number) is used to access the variable value with the FMI function calls in the binary or source. It is possible to include variables in the binary/source, that can be accessed by reference (a number), without having any mapping to a variable name in the model description file. This allows for “secret” variables. This also allows for defining “anonymous” variable names, that do not reveal any sensitive information about what the variables represent.

In order to avoid algebraic loops when using the FMU as part of a system, it could be necessary to provide a list of outputs and the variables that the outputs depend on to the importing tool.

Additional information can be included in the model description file that is typically not sensitive, like the exporting tool or experiment settings.

4.1.2 FMU Binaries

The FMU binaries typically represents a compiled implementation of the whole model. As discussed in Section 3.4, compiling a model as a binary is commonly considered sufficient protection of the model implementation. It is however up to the exporting tool to ensure that what is stored in the binary is not exposed in an open way.

An FMI binary is only required to expose the FMI API functions. These will provide access to values of at least the variables defined in the model description file. For an ME FMU, it will also be possible to access the values of each of the internal (continuous time) state variables, their derivatives, and any event indicators. But the names of such internal state variables will not be exposed.

The FMI-standard provides support for logging, so that the FMU can generate messages for warnings and errors to the simulation environment. The message generated from the FMU, using the logging interface, could depend on hardcoded messages that might include internal model information (like variable names and values) that is not exposed in the model description XML-file. The standard allows using variable references when logging, which will avoid exposure of hidden names, but could still expose hidden value references and their values. It is up to the exporting tool to ensure that such internal messages are not exposing sensitive model content.

One way to support multiple platforms is to include multiple binaries in the same FMU. This may be an option if it is not possible to provide source code for the model (as described in section 4.1.3). This requires that the exporting tool is able to compile or package binaries supported by all different platforms. FMUs for multiple platforms could be supported through cross-compilation or with tools for merging multiple binaries into the same FMU. Note that the model description needs to match all of the binaries (including the GUID).

In some cases, the FMU binary just represents an FMI gateway, as an interface to an external application or interface (like another simulation tool or network sockets).

4.1.3 FMU Source Code

An FMU could include the source code for the model, in addition to, or instead of, the binaries. This is a way to allow the model to be compiled to a general target platform, to avoid supplying a binary for each platform where the FMU is to be used. Many suppliers are however reluctant to provide source code for their models, since this exposes the implementations of the models in a more open way than compiled binaries do. Depending on how the code has been generated, this may expose algorithms, parameters, and model equations that represent valuable IP.

A common way to deal with this is to apply code obfuscation, which makes the code very difficult to read. The effort of reverse-engineering would be similar to a compiled binary.

4.1.4 External Data in an FMU

An FMU may contain additional resources. This would typically be parameter files. This means that even if internal model parameters are not exposed through the FMI interface and model description, parameter data could still be openly readable through these resource files. To avoid exposing sensitive data, it could be necessary to apply encryption or some form of obfuscation of these resources.

An FMU can contain external dependencies for example to facilitate parameterization. The standard allows the FMU to contain additional data such as files with data tables. But the FMU is also allowed to access and use external files not included in the actual FMU.

The FMI-standard only specifies the communication interface between the model and the simulation tool. Access to external data is not covered by the standard. Handling of external data files needs to be considered separately, to avoid unintentional exposure of sensitive data.

4.2 Limit Access to the FMU

The purpose of limiting the access to the FMU is either to restrict the usage of the model or restrict the access to the content in the FMU. Reasons to protect the FMU is discussed in section 3.2.

There is nothing included in the FMI standard that either specifies or restricts how to limit the access to the model binary or source code. This means that it is possible to include any protection mechanism in the source code or binaries of the FMU.

4.2.1 Examples of Access Protection

Common examples of protection that could be applied are:

- Server Solutions: Only share access to the interface. The model content is protected on the server. The user can only access the FMI function calls. This type of solution will effectively protect the model files from unintended distribution and reverse engineering.
- Encryption: The main reason for encryption is to prevent the wrong user from accessing the model. In general, the model is exposed once it has been decrypted. There are many variations of workflows and encryption solutions, for example licensing of the decryption and password protected zip-file.
- Licenses: This can be applied to ensure that the model can only be used for a certain time, or to restrict the model to only be used by a given group of people. This licensing protection would be integrated into the binaries and will thus not protect the content of the XML-file.
- Limitation over time: The binaries can be generated to only work during a restricted timeframe. This is a way to protect the model from being executed. But it does not protect the content of the model description XML-file.

4.2.2 Information About Applied Protection

In section 3.3, the importance of the available information to the recipient is discussed. The FMI standard does not specify a way to provide information to the model receiver about the type of protection applied or requirements for accessing the model. It is up to the exporter to inform the receiver, either in or outside the FMU. The standard also does not define any requirements or interfaces for protecting the access to an FMU.

The model description xml of FMI 2.0 may contain an optional flag that describes information about the intellectual property licensing. This provides information about how the FMU may be used, but not how it is protected in terms of technical licensing.

One proposal is to extend the standard with information about the type of technical licensing that is applied to an FMU. This could be added in the form of new attributes in the model description XML: "protection-type" and "protection-trigger", and an additional function in the header file "fmi2checkProtection". The "protection-type" attribute should contain information about what type of protection the FMU has, like "license-file", "time" etc. The "protection-trigger" contains information about how the protection is triggered, like "instantiation", "initialization", "2017-01-01" (for protection over time). To check if the FMU can be used at the current state, the function fmi2checkProtection can be called to perform a "validation check".

4.3 Authentication

Use cases concerning authentication were discussed in section 3.6. Authentication is not covered specifically in the current standard. However, implementation of authentication solutions in the model (binaries/sources) does not necessarily require any specific support from the standard. Two examples are given to demonstrate how the use cases can be implemented without specific support from the standard:

- **Verify the source of the FMU:** To verify that the FMU comes from the correct source, the checksum of the FMU could be digitally signed by the exporter, and provided in addition to the FMU. The signed checksum could then be used by importing tool to verify the integrity of the FMU.
- **Verify that the XML has not been altered:** The modelDescription.xml is most likely part of an FMU to be altered. It would be possible to include a function in the model binary that calculates and verifies the hash of the XML, and prevents the model from running if this is different from expected.

5 Conclusions

We have presented a survey of common use cases and concerns regarding IP protection when sharing models, and we have discussed to what extent this can be addressed within the current FMI standard.

The most common use cases concern export of models, mainly in terms of having control and information of what is exposed of the model content, as well as limiting access to unintended users. But there are aspects of this that also affect the importer, mainly in terms of usability and platform support.

Furthermore, a general need for knowledge dissemination was identified, regarding the risks and mechanisms of protecting the model content, specific to the FMI standard. One purpose of this article has been to address this need.

No obstacles were identified within the standard. All of the use cases described can be managed within the standard. Tools that export FMUs are free to include any conceivable solution for restricting the execution of the binaries, and are free to exclude all sensitive information from the model description file.

A risk for the model exporter is that sensitive information may be exposed in unintended ways, like through the logger, or through external dependencies not controlled by the standard at all.

For most use cases, it is more a question of support by the tool rather than support by the standard. The amount of information that is exposed depends a lot on the tool and specific export settings.

This leaves much freedom for tool vendors and model exporters, which can translate to challenges for model importers. The lack of standardized ways of imposing IP protection on models can make it difficult to deal with a multitude of different licensing or encryption mechanisms. Without a standardized interface it is hard to troubleshoot issues related to licensing issues. We therefore propose for a future version of the FMI standard to add an optional flag in the model description XML scheme to provide information about embedded protection that will limit execution.

Acknowledgements

This study was carried out within the research project Second Road Phase 2, coordinated by Volvo Cars Corporation and funded through the Swedish research agency VINNOVA. Time and input from all interviewees is gratefully acknowledged.

References

FMI for Model Exchange and Co-Simulation, Version 2.0: https://www.fmi-standard.org/

Köhler J., Heinkel H.-M., Mai P., Krasser J., Deppe M., Nagasawa M. Modelica-Association-Project "System Structure and Parameterization" – Early Insights. *The First Japanese Modelica Conferences, May 23-24, Tokyo, Japan, 2016.* doi: 10.3384/ecp1612435

Köhler J., King J., Kübler M. Simulation of Complete Systems at ZF using Modelica Standards, *The First Japanese Modelica Conferences, May 23-24, Tokyo, Japan, 2016.* doi: 10.3384/ecp1612424

"Smart Systems Engineering" project of the iViP Association: http://www.prostep.org/en/projects/smart-systems-engineering.html