On Automatic Crash Model Translation

Edmondo Di Pasquale[1,2]

[1] SimTech, 2 rue Albert Samain, 75017 Paris, France, edmondo@simtech.fr
[2] ENSIAME, Université de Valenciennes, France

1 Summary

This paper discusses some issues relevant to automatic crash model translation (conversion), based on SimTech experience.
Automatic crash model conversion is in many ways similar to automatic language translation.
The first level of conversion is “literal” translation, where each entity in the source model is translated into a corresponding entity in the target model. This issue is already complicated because such a target entity may not exist or may not be uniquely defined. Such a literal translation may not run on the target code, exactly as a literal translation of a text can be incomprehensible.
The second level of conversion is therefore to obtain a model which runs, in the sense of not containing contradictory cards which would block the target code.
The third level of conversion seeks to obtain a model which gives the same result on the target code as on the source code.
The methodology described in the paper is implemented in an ENKIDOU application. We show the conversion of a relatively complex model and the comparison between the results of the original (source) and converted (target) model.

2 SimTech approach to model translation

2.1 Project approach

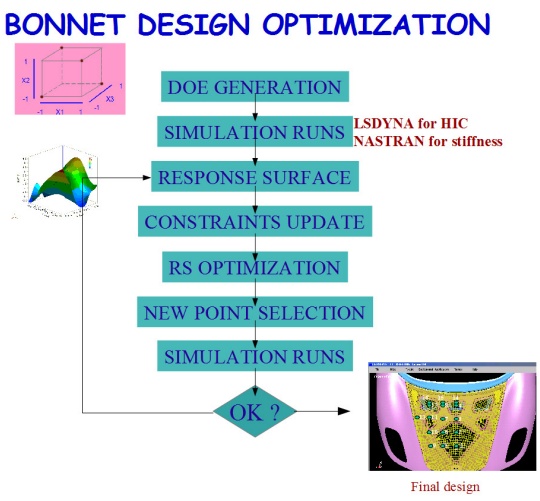
Translation between FE models corresponding to different simulation codes arises in several occasions during advanced engineering projects.

2.1.1 Product-process engineering

The first occasion where SimTech was confronted with such a problem arose in the context of product-process engineering involving sheet metal formed parts. Results from stamping simulation (in particular, using the one-step approach), were to be transferred to static solvers such as NASTRAN (for fatigue simulation, see [1]) or LSDYNA, PAMCRASH or RADIOSS for crash simulation. Technically speaking, this is not an issue of model translation, in the sense that (a) only part of the model is translated and (b) part of the translated data are the results of a simulation rather than the simulation model. Still, the problem was posed in the context of an automatic data translation, which is the topic of the present paper.

2.1.2 Automotive bonnet MDO design

Multi-disciplinary optimization is another topic where automatic model translation is very useful. In bonnet design (cfr. [2]), the definition of a bonnet as a LSDYNA mesh represent the design point. In order to evaluate the responses associated to this design point, we must
- run the head impact simulation as defined by the list of impact points
- run the static and vibration analysis to get the corresponding responses
This implies that, for each design point, we perform an automatic translation of the crash model (LSDYNA) into a static model (NASTRAN).
The challenge of this translation is that a crash model, as a general rule, allows for much more complex representation of model features, especially assembly links, so that some simplifying



assumptions must be made and implemented before the actual translation is carried out.

2.1.3 Structural optimization

A full report on the optimization of a composite rail car body has been given in [3] and [4]. In this project, two simulation process were running in parallel:
- a structural optimization using ALTAIR OPTISTRUCT
- a detailed analysis according to ALSTOM standard using ANSYS

Each version of the proposed car body, developed in the ANSYS environment, had to be converted into a NASTRAN/OPTISTRUCT model for optimization. Conversely, the composite layouts obtained with optimization had to be returned into ANSYS for verification.
Thanks to the automatic translation, the communication between the two formats was almost instantaneous and the optimization could be used effectively in the whole design process.

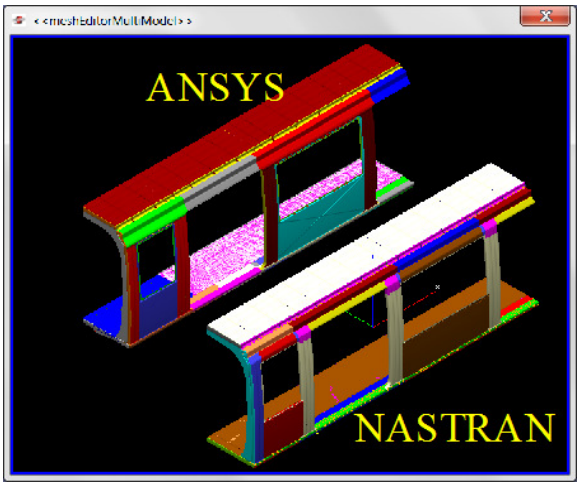


Fig.2: Car boby model conversion

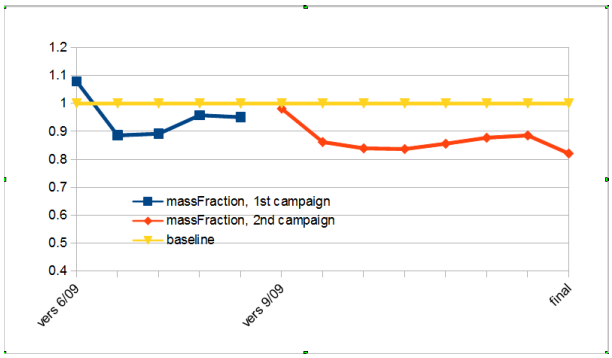


Fig.3: Evolution of car body optimal mass

2.2 Code development approach

SimTech technology for model translation is implemented in the ModelTranslator tool, based on ENKIDOU technology.
ModelTranslator lets the user performs the following tasks:
- import of one or more FE model in different formats (origin or source code)
- select a target format for a given FE model present in the task
- set up the translation options (as it will be discussed later)
- run the model translation

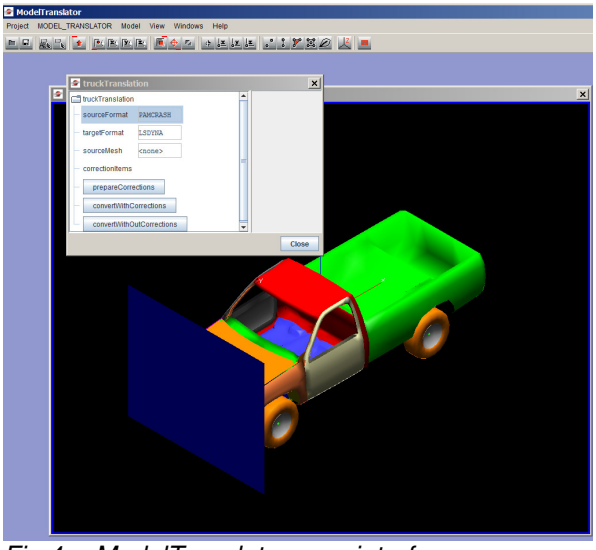


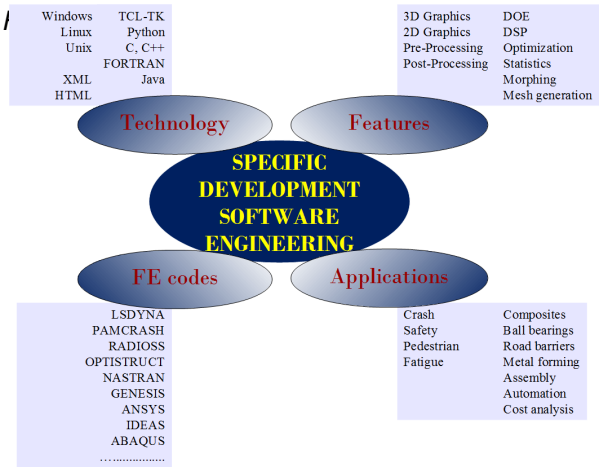
Fig.4: ModelTranslator user interface

ENKIDOU data structure is organized in user-defined tasks. For what concerns model translation, the most interesting feature is that to each task can be associated any number of heterogeneous FE modes. RADIOSS and LSDYNA models can thus be displayed, analyzed and treated in a single session of the ModelTranslator.
This is very useful in case of entity collapse or entity split, as it will be discussed later in more detail.

ENKIDOU is a SimTech library which we have used in a variety of tools, such as assembly design, optimization pre and post processing, digital signal processing, fatigue analysis, road restraining system optimisation and analysis, etc …

Most translation features available in commercial software systems treat the two meshes (source and target mesh) as two independent entities. Very often, the translation is carried out independently for each entity (card) of the source mesh.

Such an approach is obviously flawed and limited, given that, as we shall discuss later, very often there is no one-to-one relation between entities belonging to two different FE codes.



At the opposite side of the spectrum, there have been several attempts to define an archetype data structure which could contain all existing FE models, or at least the most commonly found. If such an archetype existed, model translation would be reduced simply to exporting the same mesh in different formats. To our knowledge, such an archetype is not available in the industry at present.

The approach used in ENKIDOU applications is a compromise between the two outlined above, which we hope is the most efficient.

First, all FE models present in an ENKIDOU task are extended from the same object (FeMesh). FeMesh. Basic entities such as nodes, elements, parts (in the LSDYNA sense) are shared by all the different meshes present in a task. This makes the translation of such entities immediate.

Specific features of each mesh (e.g. boundary conditions or contact definitions), although mesh-specific, refer to the same ENKIDOU objects, for instance node, elements or part groups (ENKIDOU collectors), tables (for curves and material data), etc … Even mesh-specific entities are thus, if partially, converted immediately.

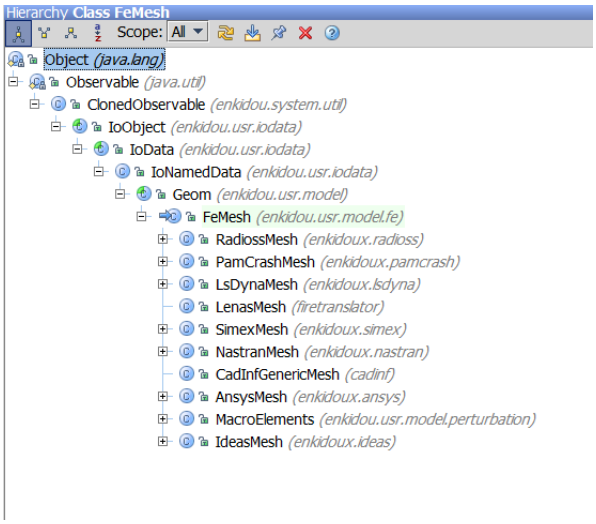


Fig.6: ENKIDOU FeMesh inheritance structure

3 The three levels of model translation

In a model translation we distinguish a source model and a target model. Translating one into another present the same challenges as automatic translation of a written text from a source to a target language.

The first automatic translators between a source language to a target language substituted words or groups of words by their equivalent according to some dictionaries. This we call “literal translation”. The resulting phrases very often did not make any sense. In FE model translation, a model that does not make sense will not run and generate error codes instead.

More advanced automatic translator introduced semantics into their treatment. As a result, they are able to select between translation alternatives so that meaningful sentences are produced. The equivalent in our context is to produce a target model which runs on the target code. This we call a “target consistent translation”.

Last, having meaningful sentences is often not enough. We need to convey the same meaning intended in the original text. For our problem, not only we want to see the target model running, but we want its behavior to be close to that of the source model. We call this an “effective translation”.

3.1 Literal translation

The first step in the translation is to generate, for each entity of the source model, an equivalent in the target model. What we ask in literal translation, as in the case of the translation of a human language text, is that the resulting target model is syntactically correct, i.e. that it corresponds to the entity description in the user manual.
In this connection, we can have three different types of conversion:

3.1.1 one-to-one conversion

In this case, there is one and only one equivalent for a source entity. This is the simplest possible case, even though some complication may arise in the treatment of LSDYNA OPTION cards. For instance, rigid body entities are converted in a *CONSTRAINED_RIGID_BODY, with or without the _INERTIA option depending on the existence of the mass and inertia parameters in the source entity.

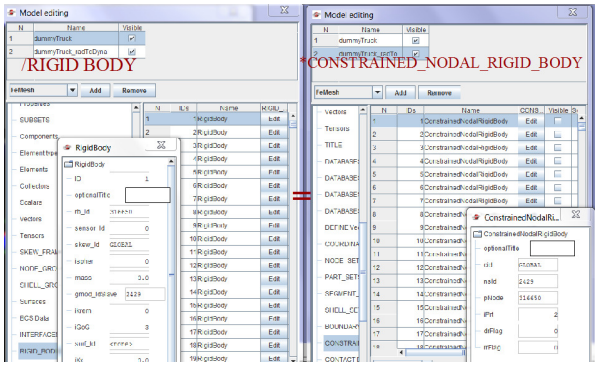


Fig.7: Example of one-to-one translation

3.1.2 Entity split

This case arises when a source entity gives rise to two or more entities in the target model. Most of the time, this happens in the treatment of the PART/MATERIAL/SECTION triads. If our target is LSDYNA, we should know that, if there are one section and one material for almost every type of LSDYNA part, this is not necessarily the case in the other crash code.

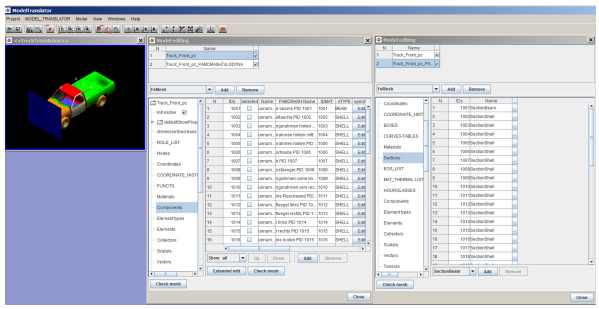


Fig.8: Example of entity split

3.1.3 Entity collapse

Obviously this transformation, where some entities disappear in the translation, is the opposite of the entity split.
It is worth noting that entity collapse appears also in different cases. For instance, the equivalent of LSDYNA *CONSTRAINED_SPOTWELD may be an link element and the corresponding part. The latter would disappear in the translation.

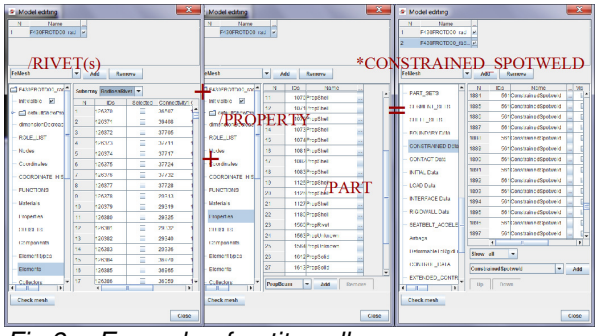


Fig.9: Example of entity collapse

Literal translation is complicated by two factors.
First, an equivalent entity may simply not exist. This happens mostly when the target and the source model treat a different physics, for instance with respect to the treatment of non-linearity. Equivalent of contact interfaces or of plastic material obviously does not exist if the target code is a linear code. Even when the physics are similar, some entities, say material models, may not have an equivalent.
Secondly, the equivalence may exist but may not be allowed for the model in question. A typical example is in the treatment of constraints in the translation of crash codes. Depending on the internal treatment of these conditions, certain combination of constraints (e.g. rigid bodies) are allowed in some code and not in others.
As a consequence, a literal translation may not be “understood” by the target code. The target code will produce error messages while running the translated model even if all the cards are syntactically correct.

3.2 Target consistent translation

A target consistent translation is a translation where the target code runs on the translated model, without producing any error message. This leads us to choose between different translations of a given entity, for instance penalty based instead of constrained LSDYNA entities.
An issue arises whether this should be carried out automatically rather that upon user demand. At present such corrections are carried out without user input, but this may change in the future.
In the context of translation of crash models, this is not sufficient to consider the translation satisfactory. Several problems may arise,such as:

- the target model may run but not be stable
- the target model may run, be stable but give results which are qualitatively different than the source model

Fixing up the problems listed above gives rise to what we call an effective translation.

3.3 Effective translation

For the purpose of this paper, we shall say that two model are qualitatively equivalent when the different entities translated work in the same way during the simulation using the two codes. In other words, we do not ask to have “the same results” in order to pronounce the translation to be effective, but we want that all the rigid bodies, the initial and boundary conditions, the contacts etc … work properly in the target model, as they do in the source model.
In ModelTranslator, the corrections needed to achieve this result are not automatic but are prompted by user actions.
The user has two ways to interact with the translation:

3.3.1 Dictionaries

Dictionaries are tables of equivalence for numerical parameters which we find in the source and in the target model. Such parameters have the same physical meaning but the numerical equivalence is not obvious or it may different from one user to another.
In the present version of ModelTranslator, dictionaries are present for element formulation parameters, such as LSDYNA ELFORMs.
The user can extend and modify the dictionaries according to the convention of his/her company.

3.3.2 Entity correction

The user can activate entity corrections to modify the target model so that its behavior is closer to that of the source model even though the entities are not translated literally.
For example, some crash codes allow for mixing rigid bodies and boundary conditions. When this situation arises, the user can choose to delete the boundary condition entity and generate an LSDYNA *CONSTRAINED_NODAL_RIGID_BODY_SPC.

4 Example of model translation

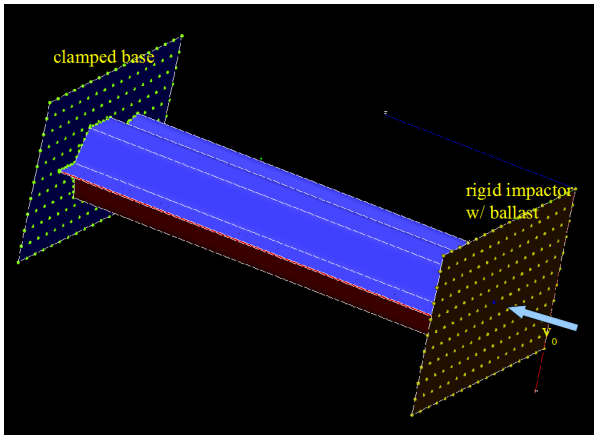
The ultimate test for a model translator is that, after automatic translation from a source to a target model, the results of the latter are the same as those of the former. In the following we show the translation procedure on a relatively complex RADIOSS model transformed into LSDYNA, along with the comparison of the crash results. For the sake of data protection, the model presented is slightly different than the actual model.

4.1 Model description

The model represent a crushed beam made up of:

- two stamped parts joined by spotwelds
- an impactor with ballast
- a base clamped to the reference frame

Spotwelds are modeled with RADIOSS springs, linked by tied interfaces.



4.2 Model translation

The translation is carried out automatically. Some relevant translation features are listed below.

4.2.1 Material conversion

In this model, one of the materials is a tabulated material, with a hardening function expressed by a set of curves for different strain rates. The LSDYNA equivalent calls for the generation of a new *TABLE entity, associated to a *MAT_PIECEWISE_LINEAR material.

4.2.2 Boundary conditions / rigid body conversion

This translation is user-defined. One of the rigid bodies in the original RADIOSS model is constrained by applying a boundary condition to one of the nodes. This is not acceptable in LSDYNA

RADIOSS LSDYNA

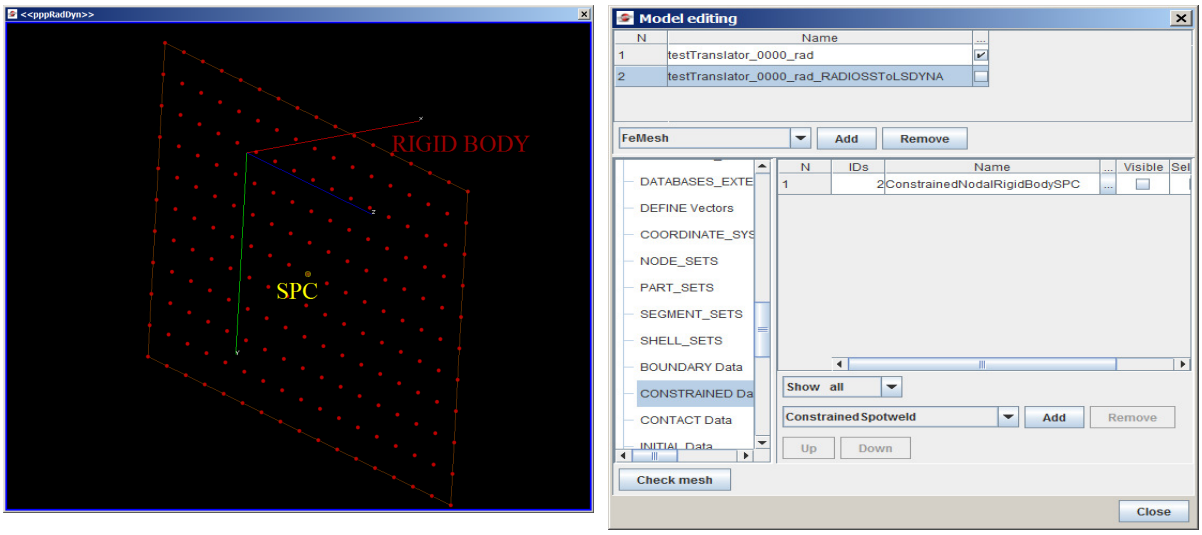


Fig.12: LSDYNA modified rigid body

4.2.3 Tria and Quad part conversion

This translation is user-defined. RADIOSS allows for different element formulation when tria and quad elements are present within the same part. In order to keep different formulation, in the translation we create a new part containing the 3-node elements.

RADIOSS

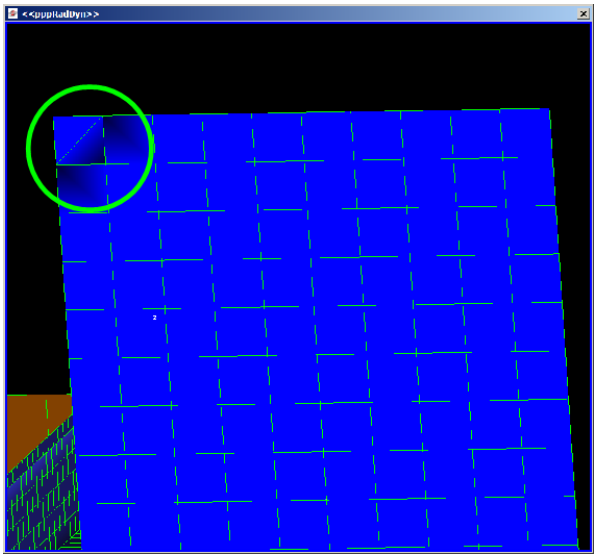


Fig.13: RADIOSS impactor with trias and quads

LSDYNA

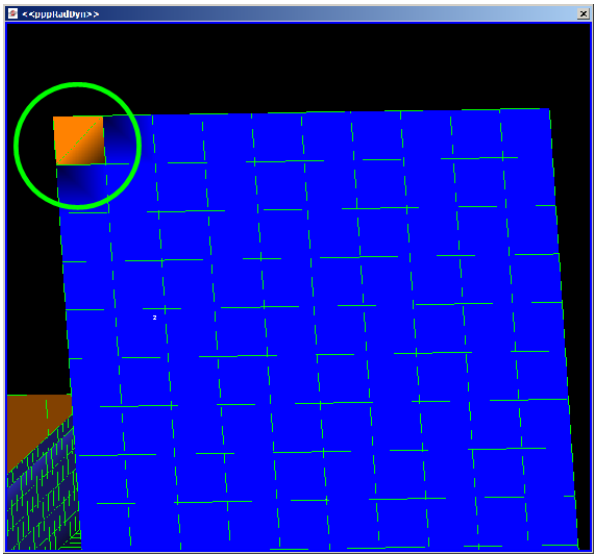


Fig.14: LSDYNA impactor with new part

4.2.4 Spotweld conversion

Spotwelds are modeled with non-linear springs, linked to each of the joined surface by a tied interface. In the conversion, we carry out the following automatic procedure:

- conversion of part and property data into corresponding part and section
- creation of a new material corresponding to the spring properties
- conversion of the tied interface into the corresponding LSDYNA contact

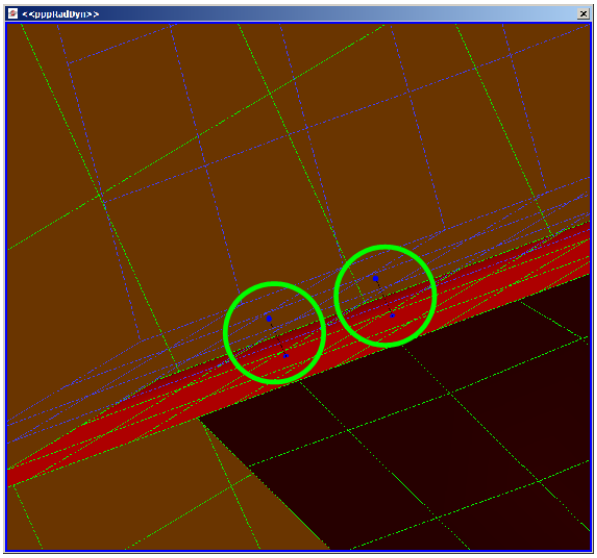


Fig.15: Detail of spotweld modeling

4.3 Crash results comparison

Results on RADIOSS and LSDYNA simulation on the crushed beam are compared in terms of overall shape, final impactor stroke and deformation energy.

Final shape

RADIOSS (stroke 138.96 mm)

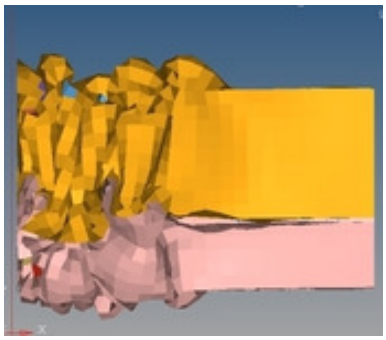


Fig.16: RADIOSS final shape of crushed beam

LSDYNA (stroke 139.81 mm)

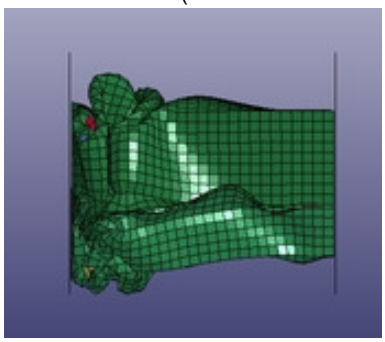


Fig.17: LSDYNA final shape of crushed beam

Energy time history

RADIOSS

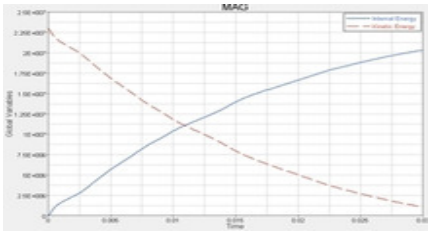


Fig.18: RADIOSS energy time history

LSDYNA

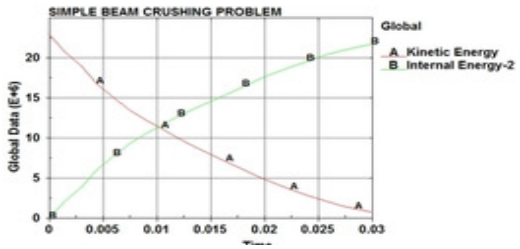


Fig.19: LSDYNA energy time history

We can point out that the pattern of deformation energy, and thus of impactor stroke and axial force, are almost identical.
However, final shape is quite different. In fact, final shape depends greatly on numerical parameters, in particular on shell element formulation. This consideration led us to the introduction of dictionaries in the ModelTranslator. The user can decide which formulation is, in his/her opinion, the best suited for his/her simulations.

5 Conclusions

Using the ENKIDOU base ModelTranslator, automatic conversion or translation of different crash models is feasible.
We have a theoretical framework and a software structure which can address all the translation problems which can be found in such situations.
Automatic translation can provide (target) models which can be run without any further modification, yielding results which are very close to the original (source) model. However, we should not forget that crash codes are different in their inner working and that, even for simple models, the behavior can be different when we look into the details.
Other translation features are available or under development, such as PAMCRASH to LSDYNA, LSDYNA to RADIOSS, LSDYNA to NASTRAN, ANSYS to NASTRAN.

6 References

[1] Y. Le Roch, J.L. Duval, E. Di Pasquale, " Coupled Sheet Metal Forming And Fatigue Simulation", Proc. IDDRG 98
[2] Di Pasquale, E., “An innovative approach to bonnet design for pedestrian safety”, 9th European LSDYNA Conf., 2013
[3] Di Pasquale, E. Gielczynski, G., "Multi-Disciplinary Optimization of Railway Systems", Research in Interactive Design, Vol. 3, 2010.
[4] Di Pasquale, E., Ghys, P., “Structural optimization in the design of a composite rail car body”, SIA, Journée d'étude « Optimisation des Composites », February 12, 2014

7 Acknowledgments

ENKIDOU and ModelTranslator are the property of SimTech. LSDYNA is the property of LSTC. OPTISTRUCT and RADIOSS are the property of ALTAIR Engineering. NASTRAN is the property of MSC Software. ANSYS is the property of ANSYS. Other products are the property of the respective owners.