From automatic event detection to automatic cause correlation

Nouran Abdelhady[1], Dominik Borsotto[1], Vinay Krishnappa[1], Clemens-August Thole[1]

[1]SIDACT GmbH

1 Abstract

Reaching and fulfilling several design and crash criteria during the development process is what makes the engineer adapt and redesign the simulation model over and over again. Ideally resulting in new simulation runs with in best case improved performance, matching the intention of the applied changes. For the more demanding case of unforeseen results which do not necessarily fit to the expectations of the actual changes, methods and a workflow are being introduced here, which allow to identify the root cause of this behavior.

In a first instance every new simulation run is being added into an analysis database, which is continuously being used to compare new simulations against. Previous studies have already shown that this process can assist the engineer in automatically highlighting new behavior and pin pointing the engineer to the regions of interest. Rather than only highlighting the new behavior now a second phase is being triggered additionally.

In this second phase the previously detected event is being isolated and analyzed against the gathered data of the development history. The analysis methods used are based up on the Principal Component Analysis, a reduced order modelling technique. This allows not only identifying structures in the data but also correlating deformation patterns against each other. Especially the latter one is of interest for an automated process, as it allows automatically detecting and suggesting possible root causes to the engineer. As an outcome of this process the engineer receives a list of correlating parts, so that he can focus on deriving a better engineering solution to achieve a deterministic behavior, rather than searching for the root cause of the event.

To provide additional information about the type of cause, as for example failure or buckling, the identified parts are also forwarded to a classification prototype. This type of classification shall assist the engineer in deriving a possible design adaptation.

2 Event Detection

In a vehicle product development cycle, numerical simulations have vastly enhanced efficiency and pace of evaluating evolving designs over the course of development. One of the several challenges that remain in such a process is to be able to accurately and easily spot unwanted behaviors as a consequence of design changes, which other-wise is a laborious and time-intensive process and moreover many a times, important behaviors accidentally go unnoticed.

2.1 Femalyst

Femalyst addresses the aforementioned issue by automatically flagging newly found behaviors termed as Events for every new simulation that is imported into it's database. This is achieved through characterizing and learning deformation patters by employing principal component analysis, a popular unsupervised machine learning technique. Femalyst is able to extract and store rich insights in its database from simulation runs at a fraction of disk space, in comparison with original runs. In addition, the tool enables the Engineer to seamlessly search for similar deformation patters across all the runs in the database in an interactive manner.

2.2 Process and Technology

Simulation runs can be imported into the database in succession. The necessary data is extracted from the simulation run during the import, processed and stored in a dedicated Femalyst database. In

the analysis stage the simulation is analyzed against all its predecessors for unknown behaviors or outliers and assigned a score for each part (or a segment of part. See *Part fragmentation*) and state. By looking at the outlier score that ranges from [0,2], one can judge the degree of outlierness of the event. A score of 2 indicating a strong outlier, while 0 indicating a known event or an inlier. In addition to assigning a score, Femalyst **clusters** several sub-part events into a bigger meaningful clusters based on the neighborhood and other measures which could potentially help the Engineer in identifying and understanding the causal chain of the event and its propagation over time and space. Such an analysis is not just performed on geometry but also on other post-values. Thus highlighting and bringing Engineer's attention to outlier-like patterns in stresses, strains etc.

The tool also addresses two major challenges that arise while analyzing a set of simulations in the entire development branch through *Part fragmentation* and *Geometry mapping*

2.3 Part Fragmentation

In order to not lose identification of local effects, the bigger parts are fragmented into smaller sub-parts to facilitate analysis on smaller regions, thus being more sensitive to tiny local behavioral patterns. The granularity of such fragmentation is user- configurable as per the need.

2.4 Geometry Mapping

Another major challenge is to consolidate the data across simulations for analysis from the entire development branch where, several changes to the model are inescapable. Geometry mapping helps overcome this challenge. Mapping helps identifying geometry entities with each other in-space despite several inevitable changes in the model such as design alterations, re-meshing, addition of new parts, changes in the part ID etc. This helps in conditioning the data from across simulations in order to extract any possible insight and take them on board for analysis. A sophisticated node and element mapping techniques have been developed and incorporated to realize this. The user has the possibility to control the scope of mapping for various assembly levels internally called as components.

2.5 Compressed Simulation Database

In order to make use of all simulation results during model development a compressed database has to be deployed which grants access to every result also at a later stage of development.

We benchmark a set of 518 A4DB full vehicle front crash simulations with ~ 5 million nodes each. The original size is 7615 GB. Applying an industry standard compression on each simulation result individually, we can reduce the data size to 363 GB. Incorporating also dependencies between simulation results, so that only differences are being stored, we achieve a compressed size of 70 GB. Thus exploiting the similarities between simulation results improves the compression efficiency by a factor of 5. For analysis purposes the data stored can be further reduced to a remaining 14 GB of data which makes it possible to analyze hundreds of simulations interactively and explore, see Fig. 1.

Making use of the similarities between the simulation results during the development phase therefore allows for a higher compression, which was the initial enabler for the development of algorithms to detect unknown behaviour.

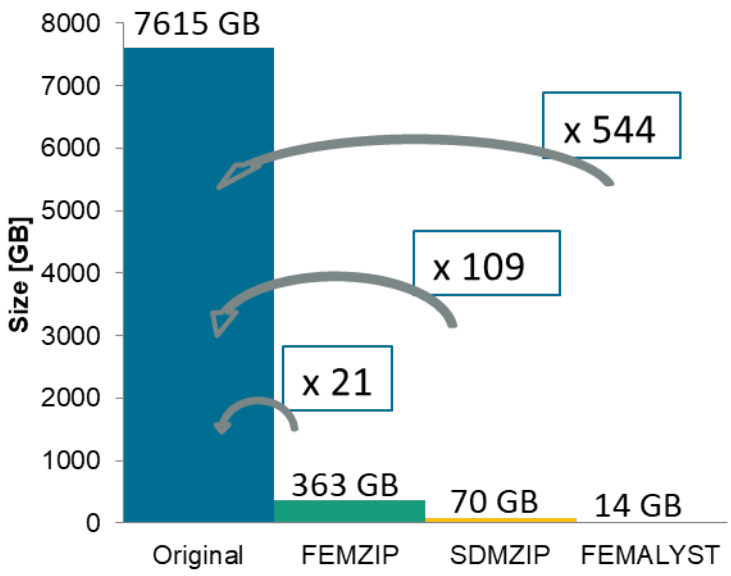

Fig.1: Compression ratio for 518 A4DB full-vehicle cases, ~5 Million nodes each, 64 Variables, 14 Variables analysed

3 Root Cause Detection

Depending on the event, the crash deviance does not always fit to the engineers expectation, based up on the changes applied to the model beforehand and might even pop up at regions out of scope. While this is one of the benefits of an automatic event detection, which does not focus on a limited area within the model, questions are prompted about the cause of this event.

3.1 PCA

To identify the possible causes for scatter in crash simulation results the Principal Component Analysis (PCA) has been introduced in the past and shown to be an effective method to analyze datasets by means of robustness and identification of root causes for scatter occurrence [1].

According to [3], principle component analysis (PCA) was introduced by Pearson in the context of biological phenomena [4] and by Karhunen in the context of stochastic processes [5].

In [6], PCA was applied to full crash simulation results. Let (p,) be the displacement of simulation run i out of n simulation runs at node p and time t. If $\bar{X}$ (p,t) is the mean of all simulation runs, the covariance matrix C can be defined as

$$C := [c_{ij}]_{1\le i,j\le n} \quad \text{and} \quad c_{ij} :=< X_i - \bar{X}, X_j - \bar{X} >_2$$

The eigenvectors v_i of C form a new basis (principle components) and the λ_i (square roots of the eigenvalues of C) provide a measure for the importance of each component.

If this method is applied to crash simulation results, n^2 scalar products between the simulations runs of length $3 * \#P * \#T$ have to be computed ($\#P$ number of points, $\#T$ number of time steps.)

From

$$\hat{X}(a) := \sum_{i=1}^{n} a_i X_i \quad ,$$

follows that

$$\lambda_i = \left\| \hat{X}(v) \right\|_2 \quad .$$

The $\hat{X}(v_i)$ show the major trends of the differences between the simulation results. The coefficients of the eigenvectors v_i correspond to the contribution of $\hat{X}(v_i)$ to $X_i - \bar{X}_i$ and can be used for cluster analysis and correlation with input parameters. If input parameters have been changed between the different simulation runs, the correlation analysis will indicate how certain trends can be avoided or increased by changing these inputs (e.g. thicknesses of parts) (c.f.[2], Chapter 2.4] for the properties of PCA analysis in general).

Principle Component Analysis is a mathematical method which determines mathematical trends in contrast to physical trends. To be more specific: λ, the square of the maximal eigenvalue of C, can be determined by

$$\lambda = \left(max_v \left\| \hat{X}(v) \right\| \middle| \|v\| = 1\right)$$

and therefore will be in general a mixture out of several physical effects, like buckling. For further insights please refer to [1].

To now find root causes of scatter occurrence a process is iteratively conducted by the engineer, including the interactive selection of the area of interest and its possible causes. During this process the principle variation modes of the scatter definition are computed using PCA, as described above. The correlation of the scatter modes to a possible source definition is evaluated using a tailored DPCA approach [1]. During such a process the engineer can freely choose the parts, as well as node-sets of interest, which is why the process has always been user driven and not fully automated in the past.

Given the fact that for a robustness analysis as described in this manuscript a set of 30 or more simulation runs is analysed, the use of a dimensional reduction method is beneficial. In our case the Principal Component Analysis is used to easier extract the essence of the crash behaviour for sets of simulations.

3.2 Part Fragmentation

The above mentioned PCA/DPCA analysis allows detecting any and all behaviours of a possible cause which are correlated to the behaviour of the target area of interest. To automate such a process, it is important to have an alternative to the engineer's interactive selection of local sets of nodes, to isolate distinct scatter sources. For this purpose, a part fragmentation algorithm was implemented. To detect effects on/of fragments rather than a whole part, parts are automatically split up during the initial simulation import.

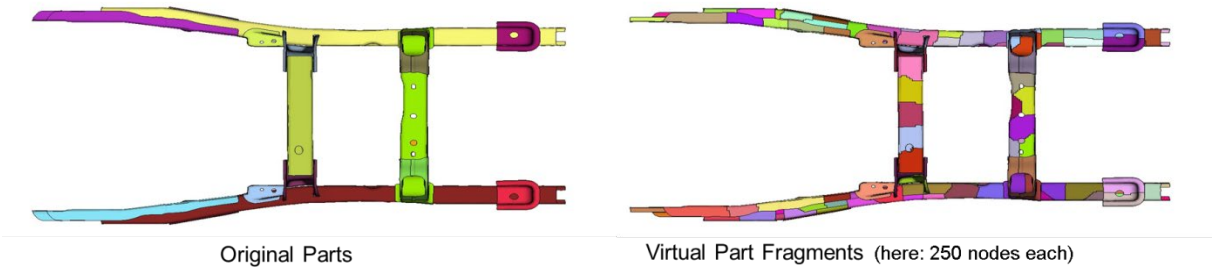

Fig.2: Automatic part fragmentation

3.3 Automation

The iterative detection of a deviation's scatter cause, described in 4.1, is error prone, and would require expert user knowledge and experience. To automate the scatter source detection process, the correlation of a deviation definition to every other fragment behaviour in the full domain at every time step is computed. The relative correlation is visualized to readily identify the biggest cause for the identified scatter.
The relevant source modes are automatically adapted for each source definition. Source modes that are less than a certain threshold of the principle source mode are neglected.

Correlation noise is automatically filtered out. High correlation against source definitions which have negligible scatter is unwanted behaviour, i.e. false negative. To filter correlation noise, the DPCA analysis of a source definition is only computed if the principle mode is above a certain threshold.

4 Cause Classification

In addition to the automatic detection of events together with an automatic root cause analysis it would be desirable to also have a classification for the type of behaviour, for which a prototype has been developed. After a deviating behaviour, compared to the previous simulation results, as well as behaviour that has been classified as undesirable by the user, is recognized, the cause is assigned to typical behaviour patterns by calculating notable variables such as the number and type of failing elements, cluster formation and/or the number of descriptive basic vectors. The recognized event classes are:

1. Failure: Class indicates failure driven scatter, and is detected based on the failure count.
2. Time-shift: Class indicates that a time lag drives the scatter, and is detected based on the derivative of the scatter variance magnitude.
3. Buckling (Spontaneous): Class indicates a behaviour bifurcation, and is detected based on the derivative of the scatter variance magnitude and direction

Not only the displacements but also other post variables such as plastic deformation may be considered. In general the event classification can be applied to all parts with respect to identifying the type of the underlying scatter occurrence and aims to provide further insides into the model and its variation in crash behaviour. This means even though in the shown example case the classification is prototypically applied to the root cause, it could also be applied to every event beforehand.

5 Test Scenario

A detailed robustness analysis [8] [9] of a set of simulation runs from the Chevrolet Silverado [7], based up on thickness variations in the range of [-3;3] % , showed a clear bifurcation. While for some simulation runs the break-booster hooked up to the suspension, being pushed into the dashboard, for others there was no such hook-up. Therefore the test scenario shown here consists of a set of simulation runs with similar behavior of no hook-up and a newly analyzed simulation containing the hook-up.

Fig.3: Comparison of hook-up event vs no hook up (previous simulations)

5.1 Automatic Event Detection

Incrementally adding the simulations to the analysis database, the simulation with the hook-up behavior throws a major event at the break booster. The fringe plot shows the outlier score computed for the complete simulation and indicates a different behavior, starting at 62 [ms]. Next to automatically getting the part which is showing a different behavior it also indicates when it starts to differ from previous simulation runs.

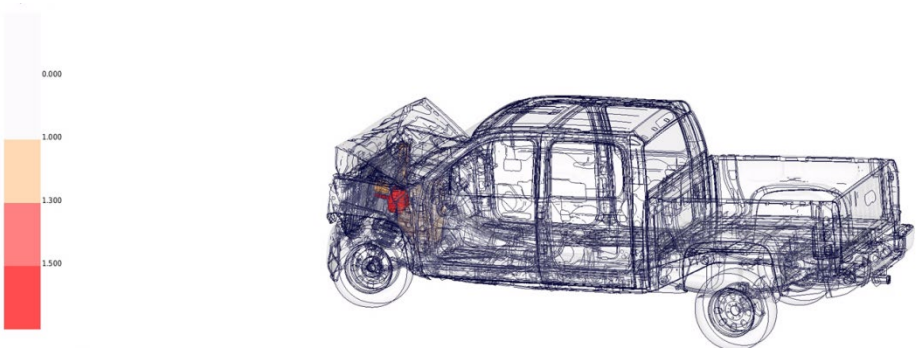

Fig.4: Event detection outlier score

5.2 Automatic Cause Identification

Based up on the results with respect to the event being detected at the dashboard/break booster, the dashboard has been forwarded to automatically identify the root cause of the hook-up. Next to obviously also highlighting the break-booster as a cause for the dashboard variation, additionally a fragment of the rail is being identified with a significant correlation, see Fig. 5.

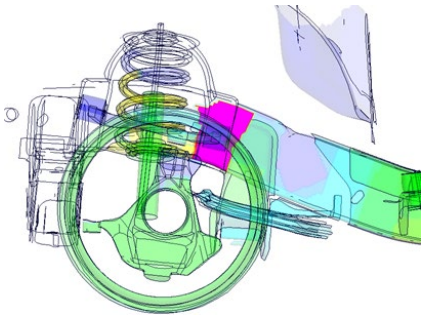

Fig.5: Automatic cause detection

While in previous studies [3] the same region could also be identified as a cause for the scatter, this process for the first time successfully combines the complete automatable chain from event detection to root cause analysis.

5.3 Automatic Classification

The prototypical classification taking part at the end of the analysis process is being applied to see if any of the behavior can be classified by the three types Failure, Time-Shift or Buckling.

While failure isn't present in the model and the relevant parts of interest do not show a clear buckling behavior, the classification indicates that for some parts a similar behavior is also present in the other runs, but a different time. Though for the involved parts a categorization could not be found based up on the definitions described in chapter 4.
This also led us to the conclusion that additional type definitions could be helpful to classify behavior. An obvious classification in this context could be contact – no contact in the future.

6 Summary

While in the past PCA based analysis allowed to identify root causes for variation in simulation results, the new modules and workflow described here allow for an automated analysis of simulation results. This automated process detects new behavior, called events, and determines correlated parts throughout the car to identify the cause, which allows for a faster reaction to counteract in case of unwanted behavior.

7 Literature

[1] C. A. Thole, D. Borsotto, L. Jansen: “Use of Data Reduction Methods for Robust Optimization”, 14th int. LS-DYNA User Conference, June 12-14, 2016

[2] Thole C.A., Nikitina Lialia, Nikitin Igor, Clees T.: “Advanced Mode Analysis for Crash Simulation Results”, 9. LS-DYNA Forum, Bamberg, 2010

[3] Lee M., Verleysen J.: „Nonlinear Dimension Reduction“, Information Science and Statistics, Springer Science+Business Media,2007

[4] Pearson K.: “On lines and planes of closest fit to systems of points in space”, philosophical Magazine,1901,2:559-572

[5] Karhunen K: "Zur Spektraltheorie stochastischer Prozesse", Ann. Acad. Sci. Fennicae, 1946, 34

[6] Ackermann S., Gaul L., Hanss M., Hambrecht T.: „Principal component analysis for detection of globally important input parameters in nonlinear finite element analysis“, In Optimisation and Stochastic Days 5.0 dynardo-dynamic software & engineering, Weimar, 2008

[7] NHTSA, Chevrolet Silverado,https://www.nhtsa.gov/staticfiles/rulemaking/pdf/cafe/Mass-Reduction-Feasibility-2014-Silverado.zip (accessed 01/25/2023)

[8] C. A. Thole, D. Borsotto, R. Strickstrock: “Robustness analysis – Significant reduction of scatter occurrence”, NAFEMS Seminar: Optimization and Robust Design, March 23-24, 2015

[9] SIDACT GmbH, DIFFCRASH, https://www.sidact.com/diffcrash, accessed Jan. 25, 2023