Politecnico di Torino

Politecnico di Torino

MSc Computer Engineering

A.a. 2025/2026

Graduation Session July 2026

Design and Implementation of a VRED Plugin for Automated Tolerance Chain Analysis VR Visualization in Automotive Perceived Quality Assessment

Supervisors:

Sandro Moos

Francesca Nonis

Sergio Carena

Candidate:

Diana A. Husanu

Abstract

This work is part of the core challenge which wants to ensure measures between the vehicle body panels meet quality standards before physical prototypes are built. To address this, a systematic approach is adopted. This is called *Dimensional Management* and we are able to achieve the goal trough the *Tolerance Chain Analysis (TCA)* process. At Italdesign, assembly simulations with accurate measures analysis are performed using 3DCS (3-Dimensional Control Systems) software and the results are then visualized in a VR environment (Autodesk VRED) through a dedicated commercial plugin: nViz Perceived Quality Plugin. In this process there is a considerable complication: the 3DCS output data formats are incompatible with the ones required by nViz plugin. Because of this constraint an extremely time-consuming and error-prone manual process of data extraction and refactoring is required, which is even more difficult because there is not any visual guidance in the process of understanding the data.

The focus of this thesis is to present the design and the implementation of **VR Tolerance Chain Analysis Plugin**, a Python plugin for Autodesk VRED developed within the XR-solutions team at Italdesign. An automation of the upstream data pipeline is implemented: the two heterogeneous 3DCS exports (an Excel model workbook and a Monte Carlo simulation CSV) are ingested and merged via a three-way join into an internal representation. Thanks to the results of this operation interactive geometric markers (spheres) are instantiated at each measurement point location in the VRED 3D scene. By clicking them, the user is enabled to select and perform operations on the desired measures, and the plugin propagates context-aware information about all related measures in the same spatial group. To address the identity mismatch between 3DCS component identifiers and VRED scene node names a semi-automated name resolution workflow based on Levenshtein string similarity is implemented. Finally, the plugin generates a correctly formatted Excel file that the nVIZ Perceived Quality Plugin can consume directly, enabling immersive visualization and comparison of different configurations in Virtual Reality, without any manual data editing.

For the implementation the following technologies have been used: Python 3, PySide6, pandas, openpyxl, VRED Python API. A graphical interface is provided, and it's addressed mainly to users without a programming background. In summary, the work replaces the previously manual multi-step workflow with an automated pipeline closing the integration gap between TCA simulation tools and VR-perceived quality evaluation.

Acknowledgements

Che dire follettini e follettine?

Termina così questo lunghissimo e tortuosissimo percorso anche per me. Sarà un giro di "grazie" molto largo.

Comincio con il ringraziare chi questa tesi mi ha permesso di farla: i miei relatori e Italdesign. Devo dire che nel contesto aziendale mi sono sentita davvero seguita e apprezzata in ogni momento. Dal punto di vista tecnico il lavoro che ho svolto è stata una figata: ho finalmente visto "cose vere" e messo le mani in pasta su qualcosa di concreto, ho imparato tantissimo e mi sono divertita un sacco nel farlo. Non me l'aspettavo. Men che meno mi aspettavo di trovarmi così bene dal punto di vista umano. Voglio esplicitamente ringraziare il mio tutor Sergio, che trovo una persona splendida. Grazie in particolare per avermi lasciato libertà di esplorare, di proporre qualche mia idea e di avermi guidata con professionalità e disponibilità. Quello che mi hai lasciato è prezioso perché, forse senza neanche rendertene conto, mi hai aiutato un sacco a credere un po' di più in me stessa in un momento in cui ne avevo davvero tanto bisogno. Poi in realtà non poteva davvero andarmi meglio perché, e qua forse non mi sono mai sbilanciata da farlo veramente capire, ma io sono una grandissima fan delle "battute da papà" e devo ammetterti che qualcuna te l'ho rubata e l'ho riproposta in giro. Sarà la parte che mi mancherà più di tutte.

Passo adesso a una persona che, in un modo forse un po' curioso, è sempre stata presente. Saranno passati 4 anni da quando ci siamo visti le prime volte perché eri curioso di saperne di più di programmazione e informatica e adesso siamo finiti che non so chi abbia insegnato di più a chi. Sono state diverse le volte in cui ero un po' demotivata e stanca di tutto, ma dopo un paio d'ore con te a parlare di dependency injection con Spring, di quale sia il miglior algoritmo per il calcolo del tragitto più breve oppure della differenza tra una rete neurale fully connected e una convoluzionale, ne uscivo più rilassata e con più entusiasmo. Sono dei momenti che mi stanno molto a cuore e te ne sono davvero tanto grata.

E i miei amichetti dalla 'c' aspirata? Con alcuni di voi ho condiviso i banchi di scuola e ho un sacco di ricordi adolescenziali sia belli che divertenti. Realizzare che non si siano fermati lì mi rende molto contenta. Durante questi miei anni torinesi non sono mancati momenti per stare insieme. Magari un po' radi, però sono stati tanto cari.

Un grazie gigante poi alle mie super girls di via Ferrere (Jack incluso of course). Che magone. Che magone grande. Piango se ci penso che non ci saranno più le cene a ridere insieme. Non ci saranno più i turni delle pulizie, le feste organizzate in salotto, le risate strappate nei momenti di esaurimento, la quotidianità confort che ci siamo create. È la parte a cui sono meno pronta di tutte.

Grazie poi a chi c'è da un sacco di tempo. A chi c'è da talmente tanto tempo che quando mi hanno conosciuta ero un'altra persona. Eravamo tutte e tre delle altre persone. In qualche modo ancora, con un po' di fatica, riusciamo a incastrare qualche momento per vederci ogni tanto, ma il bene che ci vogliamo c'è sempre e questo non cambierà mai.

Un grazie super grande alla mia supporter numero uno. La miss tutta fiocchettini e rosa confetto. Ma chi me lo doveva dire che una così sarebbe diventata la mia migliore amica e che mi avrebbe influenzato tanto nell'essere un po' più colorata e nell'avere una prospettiva più colorata della vita? La tua spontaneità e la tua generosità farebbero sciogliere anche i sassi. Mi sento davvero fortunata ad averti nella mia vita. Grazie grazie!

Un grazie anche a un rametto della famiglia che mi è stata tanto vicina, che si è sorbita diverse volte "ho preso un biglietto da Fiumicino, posso stare da voi e poi mi accompagnate a un orario discutibile in aeroporto?". Mai mi avete fatta sentire in difetto. Al contrario mi avete sempre fatto sentire accolta, non solo con le porte sempre aperte per me, ma anche le braccia. L'affetto e il bene che provo per voi io non ve lo so neanche spiegare.

E per ultimo, ma di certo non per importanza, grazie alla mia famiglia. Chissà che sarei io senza di voi. La mia mamma, il mio babbo e il mio fratellino patatino che odia ogni sorta di manifestazione di affetto ma che, tutti insieme, mi fanno sentire che c'è qualcosa di incondizionato, un punto di riferimento sempre presente e che qualsiasi cosa accada loro per me ci sono. I sacrifici che avete fatto per farmi arrivare fin qua li sapete solo voi. Grazie di cuore.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Company context and project motivation

Italdesign is an international company founded in Italy in 1968 and headquartered in Moncalieri (Turin), Italy. At the moment it is part of Volkswagen Group and operates as a full-service partner for automotive OEMs worldwide. The activities performed span through the full vehicle development life-cycle such as styling & concept design, engineering, prototyping and production ramp-up support. The work in this thesis is done in collaboration with XR-Solutions Team, which is a part of Strak & XR-Solutions Department. Within it Augmented Reality,Mixed Reality and Virtual Reality solutions for vehicle engineering & design review are developed. The primary platform used for this kind of activities is Autodesk VRED, the main automotive 3D visualization and virtual prototyping platform.

An important role in vehicle design is taken by the analysis of measures and tolerances: a properly done work in this phase can led into a considerable increase in the performance of the subsequent production and assembly phases. The dedicated approach is called Dimensional Management and inside it, with special focus on tolerances, there is the Tolerance Chain Analysis (TCA). The goal is to ensure that measures like gaps and flushnesses meet quality targets between body panels and, of course, it is performed before physical prototypes are built. At Italdesign, the main tool used for this purpose is 3DCS (Dimensional Control Systems), which defines for the model all the constraints and the standards to be followed and runs a Monte Carlo simulation of the assembly process with 10,000 iterations. The output we can export from this software consists in two files: an Excel workbook with full dimensional model definition and a Monte Carlo CSV with the statistical simulation output. The results of 3DCS can be fed into a commercial plugin that Italdesign have the license for: nViz Perceived Quality

Plugin for VRED. It is able to generate tthe geometric morphing to 3D vehicle panels and this allows us to visualize and compare gap/flush scenarios in immersive VR.

The problem in this context is that there is a structural data incompatibility between 3DCS output format and nViz input format. This required a fully manual bridging process, with four key steps to follow:

- Inspect 3DCS output files.
- Identify measurements of interest.
- Extract coordinates, part names, direction vectors and simulation results from separate tables.
- Assemble into precise format expected by nViz plugin.

This manual workflow is critical and have several pain points to be highlighted: firstly it requires a deep familiarity with 3DCS data model, then it is extremely susceptible to human error, it is poorly scaled to large numbers of measurements and there is no spacial reference to vehicle geometry during the entire process. An additional problem that adds up to this is the identity mismatch between the 3DCS model and the VRED model: 3DCS component ids are different from VRED scene node names and it is mandatory to correct this as a constraint of nViz plugin, which requires to have a perfect match between part names in the input file and the scene. The manual reconciliation is an additional pain point to the ones previously mentioned.

The described operational bottleneck has been solved with the work presented in this thesis: the design and implementation of a custom VRED plugin that automates the complete upstream data pipeline: from the ingestion of raw 3DCS exports, through interactive spatial visualization of measurement points directly in the VRED scene, to the generation of a correctly formatted, immediately consumable nVIZ input file.

1.2 Thesis objectives

The primary objective of the thesis is to describe the **VR TCA Plugin** (Python plugin for VRED) and its phases of design, implementation and validation. In this context, five concrete objectives have been defined and then satisfied:

- **Data pipeline automation**: The plugin loads the heterogeneous 3DCS exports and merge them into a single internal representation. It is able to

produce valid nViz-compatible Excel, where no manual extraction or editing is needed.

- **Spatial measurement visualization**: Geometric markers (spheres) are instantiated at each measurement point in VRED scene. Spacial reference to vehicle geometry is provided to the users.
- **Context-aware interactive selection**: The selection of a point in scene determines the auto-propagation to all points which are part of the same spacial group of measures. Through different colors, the user easily understands the group and the measure the point is involved into and he/she can safely perform operations on them.
- **Part name resolution**: A semi-automated resolution based on string similarity is provided and the user is guided into smoothly addressing the mismatch between 3DCS ids and VRED node names before exporting data for nViz.
- **Usability and integration**: The plugin is operable by mechanical/quality engineers without programming background thanks to the GUI integrated in the VRED workspace and fit's Italdesign's visualization pipeline.

1.3 Thesis structure

Here a brief overview of the content of each chapter in this thesis:

- **Chapter 2 - State of the art**: it introduces the concepts of *Dimensional Quality Management* as an engineering discipline ensuring measure tolerances before prototypes and of *Perceived Quality in automotive industry* as an analysis method of the aesthetic impact of dimensional variation on panel fit & finish. Then theoretical foundations of *Tolerance Chain Analysis* are analyzed, diving deep into the main four methodologies (*Worst-case analysis*, *Monte Carlo simulation*, *Vector Loop model* and *Unified Jacobian-Torsor*). Between these, the most suitable for industry is Monte Carlo method and there are many commercial tools for this purpose; the principal one is 3DCS. Moving forward in the chapter the focus switches on XR and VR analysis in automotive vehicle styling and engineering decision processes. Having a digital twin of the product both for prototyping and improving existing products it's an excellent strategy in the improvement of dimensional quality management across the entire product life-cycle. Then finally, the integration gap is identified: no automated bridge between TCA simulation and VR perceived quality tools is present in today's publications.

- **Chapter 3 - Analytical and contextual framework**: here both functional and non-functional requirements for the plugin development are listed and detailed. The six functional ones give constraints regarding: data loading & merging, geometric visualization, interactive selection, part name resolution, nViz output generation and session persistence. On the other hand, the four non-functional ones give a guidance on usability, performance, integration and maintainability.
- **Chapter 4 - Design and implementation**: in this chapter it is explained the overall architecture, how the plugin package is structured, the used technology stack and how each single requirement has been met during the design and the implementation.
- **Chapter 5 - Use cases and workflow**: here the end-to-end operational workflow is illustrated through a concrete use case covering all the implemented features in the plugin.
- **Chapter 6 - Results, discussion and conclusions**: in this last chapter an evaluation of the plugin is discussed, also identifying current limitations and outlining directions for future development.

Chapter 2

State of the Art

2.1 Dimensional Quality Management and Perceived Quality in automotive industry

During the development process of an industrial product, it is essential to consider that theoretical models and real-world conditions are not perfectly aligned, especially when dealing with millimeter-level precision. Also, if the final product derives from the assembly of multiple pieces, this can sum-up and the result can present multiple "imperfections".

The experience and expertise of automotive industry is that high that this kind of problem doesn't fit only in the field of technical and functional quality, but also in the so-called **Perceived Quality (PQ)**. PQ is a very important concept, but in simple words can be defined as the overall customer perception of product robustness, reliability, and the level of satisfaction or delight generated during interaction with the product. Behind its apparent subjective nature, this perception is rigorously assessed during automotive craftsmanship audits, which contain categories of demerit score for each area that can affect PQ.

Between Visual, Tactile, Acoustic, Olfactory and Ergonomic Quality, which are the PQ perceptual dimensions, the Visual is the one most impacted by dimensional variation. This is done by the appearance of joints, panel alignments and surface continuity across the vehicle body. There are 2 types of measurements involved, that are that important that are critical KPI and are needed to be studied in deep. We are referring to *gap* and *flush* measurements. Here is a clear definition of them:

- **Gap**: also referred to as *clearance*, it is the physical distance/space between two adjacent components.

- **Flush**: *Flushness* to be more precise, and it is related to the alignment of two panels; numerically it is the height difference between them. [1]

From the design phase, both of these measurements have a nominal target value and an associated tolerance. In visual quality audits the deviations from the target values of these measures are the main source of demerit scores. To achieve a high-visual-PQ is mandatory to have control over gap and flush accepted deviations.

In addition to the elementary per-point measures of Gap and Flush, in the industrial practice is common to take into account also two composite measures. These are obtained as a combination of the information of the gap and flush measurements and give an indication on the relative inclination between panels. We're talking about:

- **Gap-Parallelism**: it quantifies the tilt in the gap between two panels and it describes the *wedge-shaped condition* for which in one end the panels get close one to another and in the opposite end they do the exact opposite, so they get more distant.
- **Flush-Parallelism**: on the other hand, this measure quantifies the relative *twist* between the flushness in two panels: in one end one panel places itself in a higher position with respect to the one it should assume and on the contrary in the other end it places itself in a lower position. [2]

Figure 2.1 provides a visual schema of the four described measures to have better understanding of each of them.

Additionally from a PQ perspective, not all areas of the vehicle contribute in the same way. Zones that are highly visible or with which the user particularly interacts play a dominant role. However, the zones that the user interacts with less play of course a marginal role. Based on this, vehicle surfaces are typically classified into three categories, commonly referred to as *Zone A*, *Zone B* and *Zone C*, in order of criticality and correspondingly different tolerance requirements.

Dimensional Management (DM) is the preventive engineering methodology that ensures that functional and aesthetic dimensional requirements are met *before* physical prototypes are built. It is *preventive* because it aims to find potential problems that could arise in later, more critical phases, such as assembly, where they could result in rework, component rejection and increased both time and cost.

The central tool of DM is **Dimensional Variation Analysis (DVA)**, an advanced simulation-based methodology, which through different assembly virtual

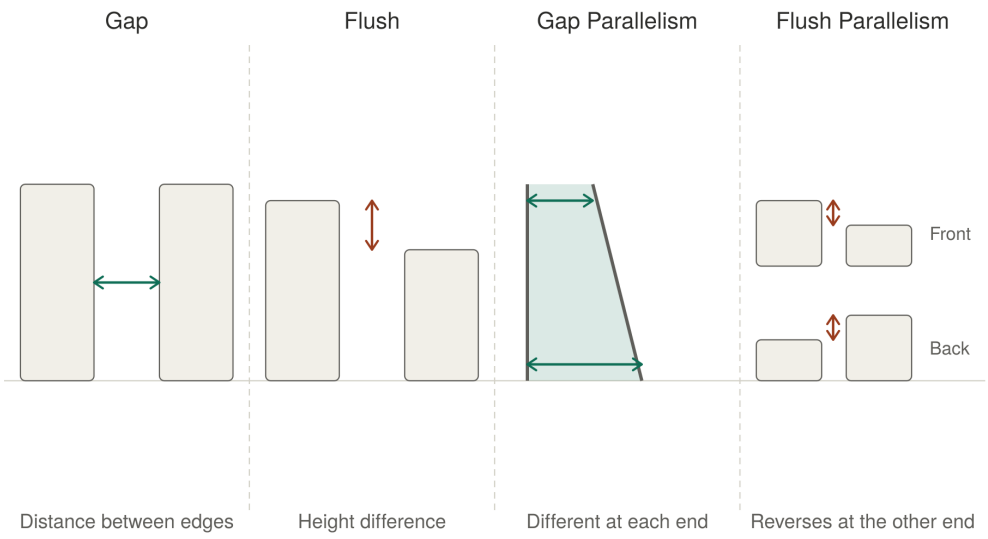


Figure 2.1: Visual representation of each type of measurement in industrial analysis for perceived quality assesment.

simulations predicts how dimensional deviations propagate and how they affect the result. In this research field the work of Sing et al. is crucial for optimal systematic understanding of the context and the method. In the two papers, *Dimensional Variation Analysis in Vehicle Design and Development: A World Class Quality Assurance Method* and *Dimensional Management in Automotive Vehicle Design & Bus Development for Perceived Quality (PQ) Improvement in buses* [3] the DVA framework is broken down and made clear through empirical relevant case studies. In more detail, the effectiveness of the method is demonstrated on the one hand through the optimization of the exhaust system though the Fitment Failure Analysis (FFA) to identify and resolve variations in pipe bends and flange orientation. On the other hand, the methodology has been applied to the improvement of non-uniform gaps and flushness between the front and side panels in buses, upgrading the overall Perceived Quality of the vehicle.

In general terms, DVA is structured around the following operational stages. First, an assembly model is defined, comprising CAD geometry, component tolerances, locating schemes, and the assembly sequence. Second, variation simulations are executed by propagating dimensional deviations through the assembly according to the specified tolerances, typically using Monte Carlo simulation to estimate

the statistical distribution of the quality response, or sensitivity-based methods to identify the parameters with the greatest influence on dimensional variation. Third, the results are analyzed to identify critical contributors and define design or process improvements. Finally, the model is iteratively refined to verify the effectiveness of the proposed modifications.

2.2 Tolerance Chain Analysis: Methods, Models and Software Tools

As discussed in the previous section, DVA relies on the ability to model and propagate dimensional deviations through an assembly. The mathematical and computational backbone of this process is known as **Tolerance Chain Analysis (TCA)**. This section provides an overview of the theoretical foundations of TCA, the spectrum of available analysis methods, the mathematical models that underpin them, and the commercial software tools, in particular 3DCS, that implement them in industrial practice.

Structured analysis methods and dedicated computer tools become necessary to track and manage tolerance addressing. Indeed, design-phase tolerances guarantee successful assembly while maintaining acceptable manufacturing costs, but additional analysis may be necessary due to the increasing in complexity caused by mechanical assemblies, in which case it is particularly error-prone and unreliable to manually manage tolerances.

Tolerances must be defined and formally expressed following established standards, which differ from one geographical region to another. In Europe, for example, the ISO GPS is the main framework, with ISO 1101 as the core standard for geometrical tolerancing. The standards guide the definition of the admissible deviation zone for each geometric feature like points, axes and surfaces and constitute the foundational input specification for any TCA model.

A key structural concept in TCA is the distinction between *open-loop* and *closed-loop* assemblies. The open-loop describes a dimension stack terminated with a gap or a critical assembly feature, a configuration typical of automotive body panel fitting, where gap and flush are the primary monitored outputs. The closed-loop, by contrast, defines a closure constraint within the assembly, implying the presence of adjustable elements. This distinction directly conditions the selection of the appropriate analysis method and the structure of the underlying mathematical model.

Different TCA methods, increasing in complexity and analytical capability, can be found in literature. Kosec et al. [4] presented the following systematic comparison, where 4 principal methods can be distinguished:

1. The **Tolerance Chart Method**, also known as *worst-case analysis*, is the simplest and most commonly used approach in industry. It creates a one-dimensional chain of tolerances and calculates the maximum deviation of a functional requirement by adding the worst contributions from each element in the chain. Its main advantage is simplicity: no computational tools are required since the analysis can be performed manually. However, this method is strictly *mono-dimensional*. Geometrical tolerances must be projected in a mono-dimensional space and orthogonal contributions are ignored. This can easily lead into excessively conservative or not accurate results. Moreover, it assumes that all components reach their worst-case scenario on their tolerance, all in the same moment. This is statistically very unlikely in real case industry scenarios.

2. **Monte Carlo Simulation (MCS)** on the other hand, instead of assuming non-realistic worst-case combinations, uses a statistical approach. From specified distributions, usually normal, which reflect real manufacturing variations, MCS randomly samples tolerance values. These values are then propagated through the assembly model over a high number of iterations. The result is a statistical distribution of the functional requirement for gap, flush or any other key characteristic. Its main strength is flexibility: MCS can handle also non-normal distributions, non-linear assembly relationships and complex GD&T specifications. It's not a coincidence that it is the standard method for industrial tolerance analysis in complex assemblies and serves as the simulation engine for commercial tools like 3DCS and VisVSA.

3. Moving towards more formally structured mathematical models, the **Vector Loop Method** extends stack-up analysis to two- and three-dimensional assemblies. The core idea is to represent each dimension of the assembly as a vector, then to connect them into chains or loops that reflects the physical assembly of the components. Contact points between components are modeled as kinematic joins and three variation classes are captured: *dimensional* (lengths and angles), *kinematic* (small adjustments in coupling joints) and *geometric/feature variation* (position, roundness, angularity). For each component, a reference system of local datum (*Datum Reference Frame, DRF*) and the associated tolerance is expressed as a small kinematic perturbation of the corresponding vector. The resulting system of equations, solved analytically

for open-loop problems or through linearisation for closed-loop ones, yields the deviation of the functional requirement. The vector loop approach constitutes the conceptual foundation adopted by 3DCS and related commercial CAT tools.

4. The most mathematically advanced, among the classical methods, is the **Unified Jacobian-Torsor (JT) Model**. This fully three-dimensional method uses the *Jacobian matrix* to connect functional requirements to virtual joint displacements. Meanwhile, the torsor model, based on small displacement screws, defines tolerance zones as spaces within which surface deviations occur. Each real surface is represented by a substitution surface defined by six screw components, three translational and three rotational, relative to the nominal geometry. This formulation does not rely on point-based approximations, which provides richer and more comprehensive information about the functional condition. Its main limitation is the significant computation and modeling complexity involved, which mostly limits its use to academic research.

Table 2.1 brings together the dimensionality, computational demands, and principal trade-offs of the four methods just discussed, before turning to a linearised alternative proposed as a complement to Monte Carlo simulation.

Dr. Cai proposed a methodological complement for tolerance modeling and analysis: the **two-step linearization process** for automotive body assembly [5]. In *step-one* for each component a linear relationship between deviations at locating points and deviations at user defined KPC points is established. In the *step-two* we have a second linearization, which propagates these component-level deviations to final assembly outputs, gap and flush. This method introduces a relevant advantage compared to MCS: **computational efficiency**. In the case study proposed by dr. Cai is revealed to be 30 times faster. Although a slight discrepancy in the statistical results is noted (with a correlation of 0.91) the computational efficiency of Cai's method is such as to make it preferable for advanced manufacturing process synthesis applications, such as tolerance allocation and optimization or assembly sequence optimization (which would otherwise require a prohibitive number of iterations with the traditional Monte Carlo method).

The method has been validated against 3DCS on an automotive body side assembly, demonstrating good trend agreement. Its applicability is, however, explicitly bounded: the approach is not suited to highly nonlinear systems, over-constrained locating schemes, or cases where non-normal input distributions introduce non-negligible higher-order effects.

Table 2.1: Comparison of the four principal Tolerance Chain Analysis methods.

Method	Dimensionality	Computational cost	Main strength	Main limitation
Tolerance Chart Method (worst-case analysis)	1D	None / manual (no computational tool needed)	Simplicity: the analysis can be done manually.	Strictly one-dimensional; ignores orthogonal contributions and assumes all components reach worst-case values simultaneously, which is statistically unlikely.
Monte Carlo Simulation (MCS)	Not specified	High (propagation over a high number of iterations)	Flexibility: handles non-normal distributions, non-linear assembly relationships and complex GD&T specifications.	Relies on propagating sampled tolerance values through the assembly model over a high number of iterations.
Vector Loop Method	2D / 3D	Moderate (analytical for open-loop; linearisation for closed-loop)	Captures dimensional, kinematic and geometric feature variations by representing each dimension as a vector.	Closed-loop configurations require linearisation of the equation system rather than an exact analytical solution.
Unified Jacobian-Torsor (JT) Model	Fully 3D	Very high (significant computation and modeling complexity)	Does not rely on point-based approximations, providing richer, more comprehensive functional information.	Significant computation and modeling complexity, mostly limiting its use to academic research.

A comparative evaluation of these methods, as conducted by Kosec et al. on an open-loop assembly problem, confirms that the four approaches yield significantly different numerical outputs even when applied to an identical problem, underscoring the critical importance of informed method selection. In industrial automotive practice, MCS remains the preferred approach due to its generality, while linearised methods find application in contexts where iterative optimisation is required.

The theoretical methods described above are implemented in *CAT (Computer Aided Tolerancing)* tools, which integrate directly in CAD systems, enabling practical, model-based TCA. Among the several existing platforms, **3DCS (Dimensional Control Systems)** is the predominant option [6], vastly used in automotive body engineering.

Natively, 3DCS operates within *CATIA V5*, or alternatively through a Multi-CAD interface, where tolerance information can be directly assigned to 3D component geometry. With 3DCS the vector loop model is implemented and Monte Carlo simulations are exectued to propagate deviations and generate statistical distributions of output measurements. Diving deeper the process is composed of the following 3 phases:

1. Model construction, comprising geometry import, locating scheme definition and GD&T annotation

2. Simulation execution (including Monte Carlo runs and sensitivity analysis)

3. Result interpretation, through contribution charts, output distributions and sigma levels for each monitored KPC.

A concrete demonstration of 3DCS capabilities in automotive *BIW* (body-in-white) assembly is provided by Meng et al. [7], who apply the software to the virtual assembly of a rear cover panel and tail light onto a white body. Using Monte Carlo simulation, the authors analyze the gap at a critical assembly point and iteratively optimise tolerance allocation through a dichotomy-based strategy: tolerances are tightened on high-contribution components and relaxed on low-contribution ones, achieving the required assembly precision while simultaneously reducing manufacturing costs. This case study confirms both the analytical power of 3DCS and the practical necessity of simulation-driven optimisation in real production contexts. The simulation output, a statistical characterization of gap and flush distributions at KPC points across the vehicle body, constitutes precisely the input data for the VR visualisation tool developed in this thesis.

2.3 Extended Reality Technologies in Automotive Product Engineering

The previous sections have established the theoretical and computational foundations of Tolerance Chain Analysis and its role in Dimensional Management. This section addresses the second technological pillar of the present work: the use of Extended Reality (XR) technologies, encompassing Virtual Reality (VR), Augmented Reality (AR), and Mixed Reality (MR), within the automotive product engineering domain. The objective is to demonstrate that XR is already an established presence across the automotive development lifecycle, and to position the specific contribution of this thesis within that landscape.

VR adoption in automotive industry has been guided by a persistent industrial demand: reduce the time-to-market and constantly improve the quality of the product in an increasingly competitive field. Physical prototypes and clay models, of course not-replaceable for some tactile evaluations, are expensive, slow to produce and difficult to modify. VR faces these limits substituting or integrating physical properties with virtual interactive counterparts, allowing early considerations in the development process, before physical prototyping.

Based on structured interviews to eleven engineers of *Jaguar Land Rover (JLR)*, Lawson, Salanitri and Waterfield documented the breadth of VR applications in one of the leading *OEM* (Original Equipment Manufacturer) of the world. These VR applications span through different use cases: from design review and virtual prototyping to production feasibility, virtual assembly, analysis of ergonomics, training of the operators. The authors note that virtual properties are typically employed earlier in the development process than their physical counterparts, precisely because they allow issues to be identified and resolved before physical commitment. At the same time, the study identifies persistent limitations of existing VR deployments, including depth perception issues at short distances, insufficient haptic and force feedback, and logistical constraints related to the fixed location of high-end systems such as CAVEs. These findings motivate a set of recommendations for future development, including richer multi-sensory simulation, improved body tracking, and networked VR for distributed teams.

The industrial relevance of VR in automotive design has grown further with the progressive shift from physical clay modeling towards fully digital styling workflows. De Clerk et al. [8], working within a BMW Group research context, address a specific and practical challenge that arises from this transition: how should designers, engineers, and management executives interact with large-screen

VR representations of vehicle exteriors in order to assess and compare design variants effectively? Their user-centered study evaluates six interaction techniques on two main tasks *Visual Inspection* and *Model Comparison* by using different input modalities like speech, gesture and touch. The results, obtained from two iterative studies with BMW design professionals, demonstrates that body tracking based on gesture (*First Person*) and direct touch on tablet (*Direct Touch*) offer the highest perceived usability, user experience and intuitiveness, minimizing at the same time the cognitive load. The study confirms that appropriate interaction design is a critical enabler for the effective adoption of VR in industrial design review workflows, and that the acceptance of VR in automotive design can be meaningfully improved through careful human-centered engineering of the interface layer.

A contribution of particular relevance to the present thesis is the work of Stoll and Paetzold, who address the problem of communicating the aesthetic impact of dimensional tolerances to designers in the early phases of product development. Their central observation is that the standard output of tolerance analysis, which are statistical distributions of predefined measurements points, are easily interpretable from tolerance specialists, but are mainly inaccessible to industrial designers, who are in charge of evaluate the aesthetic quality rather than numerical parameters. To bridge this gap, they propose a method to generate non-nominal geometry visualization: starting from CAD data, a triangular mesh is generated and locally deformed in gap and flush points defined by the user, with deviation values specified by the designer.

The resulting non-ideal prototype can be rendered and inspected in a VR environment, allowing the designer to evaluate whether the resulting gap and flush configuration is aesthetically acceptable. The deformation method fulfills a set of geometric constraints, exact satisfaction of specified gap and flush values at control points, smooth interpolation to surrounding geometry, and locality of the deformation, and was validated on automotive body data from Daimler. Stoll and Partzold demonstrate that this kind of tool allow designers to participate in a significant way to the process of tolerance allocation, reducing expensive iterations in the final phases. The work has been conducted into the *DFG Collaborative Research Center* and represents one of the first systematic attempt of close the gap between TCA output and visual design in a virtual environment.

NViz Quality Perceived Plugin represents a commercial evolution of this approach. It is a tool integrated in the VRED software, developed by *nVIZ GmbH* [9], specifically for automotive industry. The plugin imports gap and flush data measures from an Excel, generated from 3DCS and it applies a geometrical morphing to the 3D panels in the VRED scene, generating interactive variant sets that

allow to visualize and confront different gap and flush configurations including nominal, minimum and maximum and their combinations. nVIZ Perceived Quality Plugin represents the current state of the art in commercial tooling for gap and flush visualization in virtual environments, and constitutes a direct reference point for the present work.

However, both the approach of Stoll and Paetzold and the nVIZ plugin share a fundamental limitation: neither establishes a direct, automated connection to the output of a TCA simulation. The method of Stoll and Paetzold requires the designer to manually specify gap and flush deviation values at individual measurement points. NViz plugin is for sure more mature and industrially powerful, but still presents a limitation because it requires as input a strict Excel format, where measure points, part names, direction vector and statistical specific limits must be extracted and manually assembled from 3DCS heterogeneous output files. This data preparation phase is not automated and represents a practical limit for the integration of tca results in the VR visualization flow. The present work addresses precisely this gap: the plugin developed in this thesis automates the extraction, merging, validation, and formatting of 3DCS simulation data, and delivers a result table that nVIZ can consume directly, thereby closing the link between tolerance simulation and VR-based perceived quality evaluation.

The work most directly adjacent to the present contribution from the assembly side is that of Dalle Mura and Dini, who develop an **AR prototype system to support operators during panel fitting operations in car body assembly**. The practical challenge the system faces is that the alignment of the panels within the tolerance limits of gaps and flushes is a manual process depending on the ability of the operator, where he/she does not receive any guidance on how to correct detected misalignment. The propose solution integrates the robotic measurement through gap and flush sensors in control points with an AR interface provided as a wearable device on the head. An algorithm converts the measured deviations in step-by-step instructions overlapping the field of view of the operator. A *correlation matrix* is built empirically from the measured in the monitoring and it maps the regulations needed on the gaps and flushes on the control points. This allows the system to identify the optimal corrective action. The system is validated on a physical assembly of hood, fender, bumper and headlight and it reduces the assembly time of a factor of approximately four in comparison to the non-guided manual regulation. This contribution demonstrates the feasibility and practical value of integrating dimensional quality information into an XR-based interface, but operates downstream, on the assembly line, using measured data to guide corrective action in real time. The plugin developed in this thesis operates upstream, in the design phase, using simulated TCA output to support dimensional quality

evaluation before any physical part exists.

2.4 Digital Twin and Digital Integration in the Metrological Process

The previous sections have established the theoretical foundations of TCA and the role of XR technologies in automotive product engineering [10]. This section positions the present work within a broader and rapidly evolving trend: the digital integration of simulation, measurement, and quality assurance processes through the concept of the **Digital Twin (DT)**. Although this section is more concise than the preceding ones, it provides an important conceptual frame for understanding the nature of the contribution developed in this thesis.

The Digital Twin is a concept that has evolved significantly since its early formulations, and is now understood as an integrated, multiphysics, probabilistic simulation of a system or object that uses available models, sensor data, and input information to mirror and predict the behavior of its corresponding physical counterpart. A fundamental characteristic of a mature DT is its dynamic nature: the virtual model is not static, but continuously updated with real-time data from the physical world, enabling monitoring, prediction, and decision support across the product life-cycle.

Within the field of metrology and dimensional quality assurance, the application of DT technology is a relatively recent but rapidly expanding research direction. Measurements and measurement systems play a central role in DT design and operation: they constitute the primary data source that links the physical and virtual worlds, and their quality directly conditions the reliability of any DT-based analysis. As reviewed by Poniatowska et al. [11], current DT applications in metrology span a broad range, from *virtual coordinate measuring machines (VCMMs)* used to optimize measurement planning and estimate measurement uncertainty, to DT-based systems for geometry assurance that use 3D scan data to enable real-time optimization of assembly processes. A particularly relevant thread in this literature is the concept of geometry assurance through DT, as developed by Söderberg et al., the idea that geometric deviations of individual parts, the propagation of variation through assembly, and the effects of joining processes can all be captured, simulated, and managed within a unified digital framework across the product lifecycle.

Critically, Poniatowska et al. [11] identify interoperability and data integration as one of the primary open challenges in DT-based metrology. The effective exploitation of DT technology requires seamless bidirectional data flow across

the product development chain, from design specification through simulation, manufacturing, and inspection, a vision closely related to the concept of the Digital Thread. This challenge is particularly acute in contexts where dimensional quality data must be communicated across disciplinary and tool boundaries: from tolerance analysis software to visualization environments, from engineering teams to design stakeholders. It is precisely within this broader trajectory that the present work is situated. The plugin developed in this thesis can be understood as a specific implementation of the DT principle applied to the dimensional quality domain: it creates a bridge between the output of a 3DCS tolerance simulation, a statistical, data-driven prediction of gap and flush behavior, and its visual digital twin in Autodesk VRED. Rather than leaving the TCA output confined within the numerical and graphical conventions of the simulation tool, the plugin makes it navigable, explorable, and interpretable in an immersive virtual environment, by engineers and designers who may not possess the specialist background required to read simulation reports directly. In doing so, it contributes to the broader objective of integrating dimensional quality information into the digital product development workflow, reducing the communication barriers that separate simulation specialists from design decision-makers.

2.5 Literature Gap and Motivation for the Contribution

The preceding sections have surveyed the two principal domains that constitute the foundation of the present work. On one side, a consolidated body of literature addresses the theory and practice of Tolerance Chain Analysis: from worst-case methods to Monte Carlo simulation, from vector loop models to the Unified Jacobian-Torsor formulation, and from academic comparison studies to industrial implementations in tools such as 3DCS. On the other side, a growing body of research documents the adoption of Extended Reality technologies in automotive product engineering, encompassing VR-based design review, human-centered interaction design for virtual environments, gap and flush visualization on non-nominal prototypes, AR-assisted assembly guidance, and the emerging integration of DT concepts with metrological processes. Examined together, however, this literature reveals a significant and unaddressed gap. The TCA methods and tools described in §2.2 are highly effective at predicting dimensional variation and its statistical distribution across assembly KPC points. Their output, typically presented as histograms, sigma levels, and contribution charts within dedicated software environments, is technically rigorous but inherently inaccessible to the broader design team. Designers, programme managers, and quality engineers without a background in tolerance analysis are rarely in a position to translate a Monte Carlo

output distribution into an intuitive understanding of what the vehicle surface will look like when dimensional variation is present. This communication barrier is widely acknowledged in the literature: Stoll and Paetzold identify it explicitly as the central motivation for their visualisation work, noting that distribution curves from tolerance analysis tools are readily interpretable by specialists but opaque to designers who need to assess aesthetic quality. The XR contributions reviewed in §2.3 demonstrate that virtual environments are a powerful medium for communicating spatial and dimensional information in the automotive context. Lawson et al. [12] confirm the strategic value of VR across the development lifecycle; de Clerk et al. [8] show that appropriate interaction design can make VR-based design review genuinely usable for automotive professionals; Dalle Mura and Dini [8] demonstrate that AR can convey gap and flush correction instructions effectively on the assembly floor; and Stoll and Paetzold [13] show that non-nominal geometry can be visualized in virtual environments in a way that is meaningful for designers.

Yet none of these contributions establishes a direct, systematic integration between the statistical output of a TCA simulation and an interactive VR environment designed for design-phase decision support. Stoll and Paetzold's approach [13], the closest precursor in intent, requires manual input of gap and flush deviation values by the designer and operates independently of any simulation data. Dalle Mura and Dini's [10] system uses real measured data from the assembly line rather than predicted simulation data, and targets assembly operators rather than design engineers. The XR contributions of Lawson et al. [12] and de Clerk et al. [8] address design review and model assessment, but do not incorporate dimensional quality data from tolerance simulations.

At this point the integration gap should appear both clear and relevant. From one side, TCA tools produce rich and precious statistically forecasts that are limited to the experts in the fields. On the other side, VR environments are becoming a consolidated tool for automotive design review, but they are still not connected to TCA outputs in a systematic way. The presented work proposes a customized plugin for Autodesk VRED that imports 3DCS data and elaborates it in the format required by nViz Perceived Quality Plugin. After this step, nViz supplies the engin for the geometrical morphing for the visualization in scene of the statistical results, allowing engineers and designer to perform a visual perceived quality assessment.

Chapter 3

Prerequisites and Requirements

This chapter establishes the analytical and contextual framework underlying the design and implementation of the plugin. The first part identifies the functional and non-functional constraints that the plugin must satisfy and derives them from the problem context they are intended to address. The second part formally specifies the structure and content of the data exchanged at each boundary of the system: the heterogeneous 3DCS simulation exports that serve as input, and the nVIZ-ready Excel file that the plugin generates as output.

3.1 Requirements Analysis

3.1.1 Problem Definition

As established in Chapter 2, the nVIZ Perceived Quality Plugin provides a powerful engine for gap and flush visualization through geometric morphing in a VRED scene. Given a suitably formatted input file, the plugin is capable of generating interactive variant sets that allow designers and engineers to visualize and compare different gap and flush configurations (*nominal*, *minimum*, *maximum*) in an immersive virtual environment.

However, the prerequisite for this capability is a precisely formatted Excel file whose content (measure point coordinates, part names, direction vectors, and statistical specification limits) must be extracted and assembled from the heterogeneous output of a 3DCS simulation. In practice, the 3DCS software exports its model and simulation data in two distinct files: an Excel workbook containing the full model definition across multiple structured tables, and a CSV file containing the statistical

results of the Monte Carlo simulation. Neither file is directly consumable by nVIZ, and the information required to construct the nVIZ input table is distributed across both.

Before the plugin development described in this thesis, this data preparation step was performed manually. This tedious process involved first examining the simulation file and identifying the measures considered. Then, from the 3DCS model, the relevant rows corresponding to those measures and the related point geometries (coordinates and vector data) had to be extracted. Finally, all the relevant information had to be assembled in the nViz table format. Considering that this process requires both a detailed understanding of the 3DCS data model and meticulous manual effort, it is extremely time-consuming and error-prone, making it also very difficult to analyze a high number of measures simultaneously. Also, the process is made more complicated and difficult to do because the selection was done only from the inspection of raw data tables (no reference to the actual vehicle geometry in the virtual environment).

The present work addresses this problem by automating the full upstream pipeline: from the ingestion of raw 3DCS output, through spatial visualization of measurement points in the VRED scene, to the generation of a correctly formatted nVIZ input file. The following sections define the requirements that this automated pipeline must satisfy.

3.1.2 Functional Requirements

The plugin must satisfy the following functional requirements:

- **FR1 - Data Ingestion.** The plugin must parse and extract the relevant information from the two input files provided by the simulation department: an Excel workbook (.xlsx) containing the 3DCS model data, and a semicolon-delimited CSV file containing the statistical results of the 10,000-iteration Monte Carlo simulation. An internal unified representation must be created from the validated and merged parsed data in order to use it for all the downstream operations.
- **FR2 - Measure Point Visualization.** The plugin must instantiate a set of interactive geometric objects in the VRED scene, one for each unique measurement point identified in the merged data. Each object must be positioned at the three-dimensional coordinates extracted from the input data, and its visual state must reflect its current status within the workflow, in particular whether a nVIZ variant set association has already been established for it.

- **FR3 - Interactive Measure Selection.** The plugin must allow the user to identify and select measurement points of interest by directly interacting with the VRED scene, rather than requiring navigation of raw data tables. The selection mechanism must be context-aware: selecting a single point must propagate information about all other points belonging to the same logical group and to the same part pair, enabling the engineer to understand spatial relationships within and between measurement groups without leaving the virtual environment.

- **FR4 - Part Name Resolution.** Because the internal component identifiers used in the 3DCS model may differ from the node names used for the corresponding geometry in the VRED scene, the plugin must provide a workflow for the user to establish and confirm the correspondence between these two naming conventions. The resolved part names must replace the original 3DCS identifiers in the output data so that nVIZ can correctly associate each measurement with its corresponding geometry in the scene.

- **FR5 - Output Generation.** When the user selects a set of measurement points, the plugin must be able to generate an Excel file in the precise format required by nViz Perceived Quality Plugin. This file must contain seventeen columns indicating each selected measure with the coordinates of the two associated measurement features, the resolved part names, the direction vector and the estimated statistical specification limits derived from the Monte Carlo simulation. The format is important and rigid because this file is going to be consumed by nViz without requiring any further manual editing.

- **FR6 - Session Persistence.** The plugin must keep track of the connections between measurement subgroups and any nVIZ variant set names across VRED sessions. This ongoing record allows the visual state of the scene to show previously established connections when the plugin is reopened, preventing users from having to redo the selection process from the start.

3.1.3 Non-Functional Requirements

In addition to the functional requirements above, the plugin must satisfy a set of non-functional constraints that govern its integration into the VRED environment and its usability by the intended audience.

- **NFR1 - Scene Non-Invasiveness.** The interactive measurement point objects introduced by the plugin into the VRED scene must not visually interfere with the vehicle geometry or obstruct the spatial assessment that the session is intended to support. Their size must be calibrated to be readily

identifiable without dominating the scene, and their presence must not affect the performance or rendering quality of the existing VRED environment.

- **NFR2 - Performance.** The plugin must remain responsive under the data volumes typical of an automotive interior assembly that may involve several hundred measurement points distributed across dozens of logical groups. Data loading, sphere creation, and interactive operations must complete within a extended waits between user actions. extended periods between user actions.

- **NFR3 - Usability.** The plugin must be operable by mechanical and quality engineers with domain knowledge in tolerance analysis but without necessarily have a programming background. All interactions, from project selection through measure export, must be achievable through the graphical user interface without access to source code or manual file editing.

- **NFR4 - Data Integrity.** The plugin must detect and alert the user to any inconsistencies between current input data and the state of the VRED scene, such as measurement points whose coordinates in the loaded data differ from those of the corresponding objects currently present in the scene. The user must be gidiscarded to prevent inadvertent data loss.

Table 3.1 maps each requirement introduced above to the section of Chapter 4 where its implementation is described.

3.2 Input and Output Data Specifications

The plugin works between three different data environments: the 3DCS dimensional variation analysis software, the VRED scene, and the nVIZ Perceived Quality Plugin. This section specifies the structure and content of the data exchanged at each boundary.

3.2.1 Input 1 - 3DCS Model Export File (.xlsx)

The primary input is an Excel workbook exported from 3DCS. This file contains the full structured definition of the dimensional analysis model and contains multiple sheets covering geometry, GD&T specifications, locating schemes, move definitions, tolerance assignments, and measurement configurations. The plugin extracts data exclusively from two of these sheets: *Measures* and *Points*.

The **Measures** sheet contains one row per measurement defined in the 3DCS model. Of the many columns present in this sheet, the plugin uses the following:

Table 3.1: Traceability matrix between requirements (Chapter 3) and their implementation (Chapter 4).

Requirement	Description	Implemented in
Functional Requirements		
FR1	Data Ingestion: parse and merge the two 3DCS input files (model export and Monte Carlo results) into a unified internal representation.	§4.2
FR2	Measure Point Visualization: instantiate interactive geometric objects in the VRED scene at each measurement point, reflecting their current workflow status.	§4.3
FR3	Interactive Measure Selection: allow context-aware selection of measurement points directly in the VRED scene, propagating group and part-pair information.	§4.4
FR4	Part Name Resolution: provide a workflow to resolve the mismatch between 3DCS part identifiers and VRED scene node names.	§4.5
FR5	Output Generation: generate the nVIZ-compatible Excel file containing the selected measurements in the required format.	§4.6
FR6	Session Persistence: keep track of the connections between measurement subgroups and nVIZ Variant Set names across VRED sessions.	§4.7
Non-Functional Requirements		
NFR1	Scene Non-Invasiveness: measurement point objects must not visually interfere with the vehicle geometry or the scene rendering.	§4.8
NFR2	Performance: the plugin must remain responsive under data volumes typical of an automotive interior assembly.	§4.8
NFR3	Usability: the plugin must be operable, from project selection through measure export, by engineers without a programming background.	§4.8
NFR4	Data Integrity: the plugin must detect and alert the user to inconsistencies between current input data and the state of the VRED scene.	§4.3

- **Measure**: the unique alphanumeric identifier for the measurement (e.g., `C190a_F`), which encodes the logical group, the specific point, and the measurement type according to the naming convention described in Section 3.2.2.
- **Description**: a human-readable label describing the measurement.
- **Name(Feature)**: the name of the first geometric feature involved in the measurement, corresponding to the reference point on the first part.
- **Part Name**: the identifier of the part containing the first feature.
- **Name(Feature).1**: the name of the second geometric feature, corresponding to the reference point on the second part.
- **Part Name.1**: the identifier of the part containing the second feature.
- **I, J, K**: the three components of the unit direction vector along which the measurement is evaluated.

The **Points** sheet contains one row per geometric point defined in the 3DCS model. The plugin uses the following columns:

- **Point**: the name of the point, used as a join key to match against the `Name(Feature)` and `Name(Feature).1` columns of the Measures sheet.
- **X, Y, Z**: the three-dimensional coordinates of the point in the vehicle coordinate system, expressed in millimetres.

Two successive left joins between the Measure and the Points tables are performed in order to attach the coordinates to the corresponding feature names, producing an unified internal results table.

3.2.2 Measure Naming Convention

Each gap and flush measurement identifier follows a structured naming convention:

`[PREFIX][GROUP_ID][POINT_ID]_[SUFFIX]`

Where:

- **PREFIX**: a single alphanumeric character identifying the sub-assembly context (for example, `C` for cockpit, `M` for miko, et cetera).
- **GROUP_ID**: a sequence of digits identifying a logical group of measurement points located in a spatially concentrated area of the assembly.

- **POINT_ID**: a single alphabetical character distinguishing individual measurement points within the same logical group.
- **SUFFIX**: either `_F` or `_B`, indicating the measurement type. In the context of this plugin, `_F` identifies a *gap* measurement and `_B` identifies a *flush* measurement, following the convention adopted in the 3DCS simulation model.

It is worth noting that while POINT_ID as defined above refers exclusively to the single alphabetical character (e.g., `a`), the plugin internally identifies each measurement point by concatenating GROUP_ID and POINT_ID into a single string (e.g., `190a`). This concatenated identifier is used to name the corresponding sphere node in the VRED scene, ensuring global uniqueness across all groups.

For example, the identifier C190a_F shows a gap measurement at point a in logical group 190 of the cockpit assembly context. While C190a_B indicates the corresponding flush measurement in that same location. Within a logical group, there can be multiple subgroups, each defined by a different pair of parts. A subgroup includes all measurements that share the same part pair, whether they are gap or flush type. For instance, group 190 may include measurements between Part A and Part B, Part A and Part C, and Part B and Part C. This creates three distinct subgroups in the same area. This structure forms the basis for the plugin's interactive selection and output generation.

Furthermore to "simple" gap and flush measurements, it is possible to get insights from Monte Carlo simulations also about more sophisticated types of measurements: *gap-parallelism* and *flush-parallelism*. This are an addition to measures of gap and flush, which means there must exist at least the corresponding points of gap or flush with the mentioned above convention. The additional information about the *inclination* (or *wedge* or *parallelism*) between the panels is taken from lines with this convention:

`[PREFIX][GROUP_ID][POINT_ID_A][POINT_ID_B]__SUFFIX`

with the difference that this time we have both the point ids and a double underscore before the suffix. The suffix can be **FP** for gap parallelism and *BP* for flush parallelism, but this last case has not been taken into consideration.

To make it even more clear, let's consider an example. We have a couple of measures of gap: C190a_F and C190b_F. This means we have some information about the minimum and the maximum deviation in those 2 points and from this information we can calculate the minimum and maximum gap. If in the point

190a the deviation is minimum and in 190b the deviation is maximum, we cannot describe this situation with C190a_F and C190b_F. There will be an additional measure C190ab___FP where the key information about the wedge is encoded.

3.2.3 Input 2 - Monte Carlo Results File (.csv)

The second input is a semicolon-delimited CSV file containing the statistical output of the 10,000-iteration Monte Carlo simulation performed by 3DCS. Each row corresponds to one of the measurements defined in the Measures sheet and characterizes the statistical distribution of the gap or flush value at that point across the simulation sample.

Of the columns present in the file - which include the nominal value, mean, minimum, maximum, out-of-tolerance rates, and several estimated distribution parameters - the plugin extracts only the following three:

- **Name**: the measurement identifier, used as the join key to match each simulation result row against the corresponding row in the internal results table.
- **Est.Low**: the estimated lower bound of the measurement distribution, expressed in millimetres. This value is derived from the tails of the simulated distribution and represents the statistical minimum of the gap or flush across the 10,000 assembly iterations.
- **Est.High**: the estimated upper bound of the measurement distribution, in millimetres. This value represents the statistical maximum of the gap or flush across the simulation sample.

These two values are transferred to the `LSpecL` and `USpecL` fields of the output table respectively, providing the dimensional data required by nVIZ to drive the geometric morphing at the extreme positions of the statistical distribution.

3.2.4 Output - nVIZ Input File (.xlsx)

The output generated by the plugin is an Excel file formatted for nVIZ Perceived Quality Plugin. It contains one row per selected measurement, with the following columns:

Column	Content
`Measure`	the measurement identifier.
`Description`	text with human-readable information about the measure.
`Name(Feature)_1, Part Name_1, X_1, Y_1, Z_1`	the feature name, scene part name, and three-dimensional coordinates of the first measurement point.
`Name(Feature)_2, Part Name_2, X_2, Y_2, Z_2`	same, but for the second measurement point.
`I, J, K`	the components of the measurement direction vector.
`USpecL, LSpecL`	the upper and lower specification limits, populated from the `Est.High` and `Est.Low` values of the Monte Carlo simulation results.

A critical distinction between the internal results table and the output file concerns the part names. In the internal representation, `Part Name_1` and `Part Name_2` contain the original 3DCS component identifiers, which follow the naming conventions of the CAD model and are typically long, structured strings encoding organizational metadata. These identifiers do not correspond to the node names used for the same geometry within the VRED scene, which are typically shorter and follow a different convention. Since nVIZ identifies the geometry to be morphed by matching part names against VRED scene node names, the output file must contain the VRED-compatible names rather than the 3DCS identifiers. The part name resolution workflow (FR4) bridges this gap: before export, the user maps each original 3DCS part name to its corresponding VRED scene node name, and the plugin replaces the identifiers in the output data accordingly. The resulting file is immediately consumable by nVIZ without further modification.

Chapter 4

Plugin Implementation

This chapter describes the design and implementation of the *VR Tolerance Chain Analysis Plugin*, a VRED-integrated tool that automates the data pipeline from raw 3DCS output to an interactive VR environment and a nVIZ-ready export file. The chapter is organized around the six functional requirements defined in Chapter 3: each section maps to one FR and explains the architectural choices, data structures, and algorithms that satisfy it. An opening section establishes the overall system architecture and the technology stack on which all components rely.

4.1 System Architecture

The plugin is implemented as a Python package that is loaded inside the VRED runtime. It exposes a Qt-based panel docked within the VRED workspace and interacts with the host application through the VRED Python API. Figure 4.1 illustrates the high-level data flow through the system.

Module decomposition. The package is partitioned into three sub-packages:

- **core/** - stateful components responsible for data management and scene interaction: `TablesManager` and `SpheresManager`.
- **ui/** - Qt widgets that compose the graphical interface: `VisualizationTab` (the main orchestrator), `ProjectSelector`, `MeasureInspector`, and `PartNameMatchingDialog`.
- **utils/** - stateless helpers: the Levenshtein string matcher, a Qt/pandas table model adapter, and a structured logger.

Technology stack. The plugin is written in Python 3 and relies on four external dependencies, all available inside the VRED runtime:

- *PySide6* - Qt bindings used for all GUI components, the signal/slot event model, and 3D vector arithmetic (`QVector3D`).

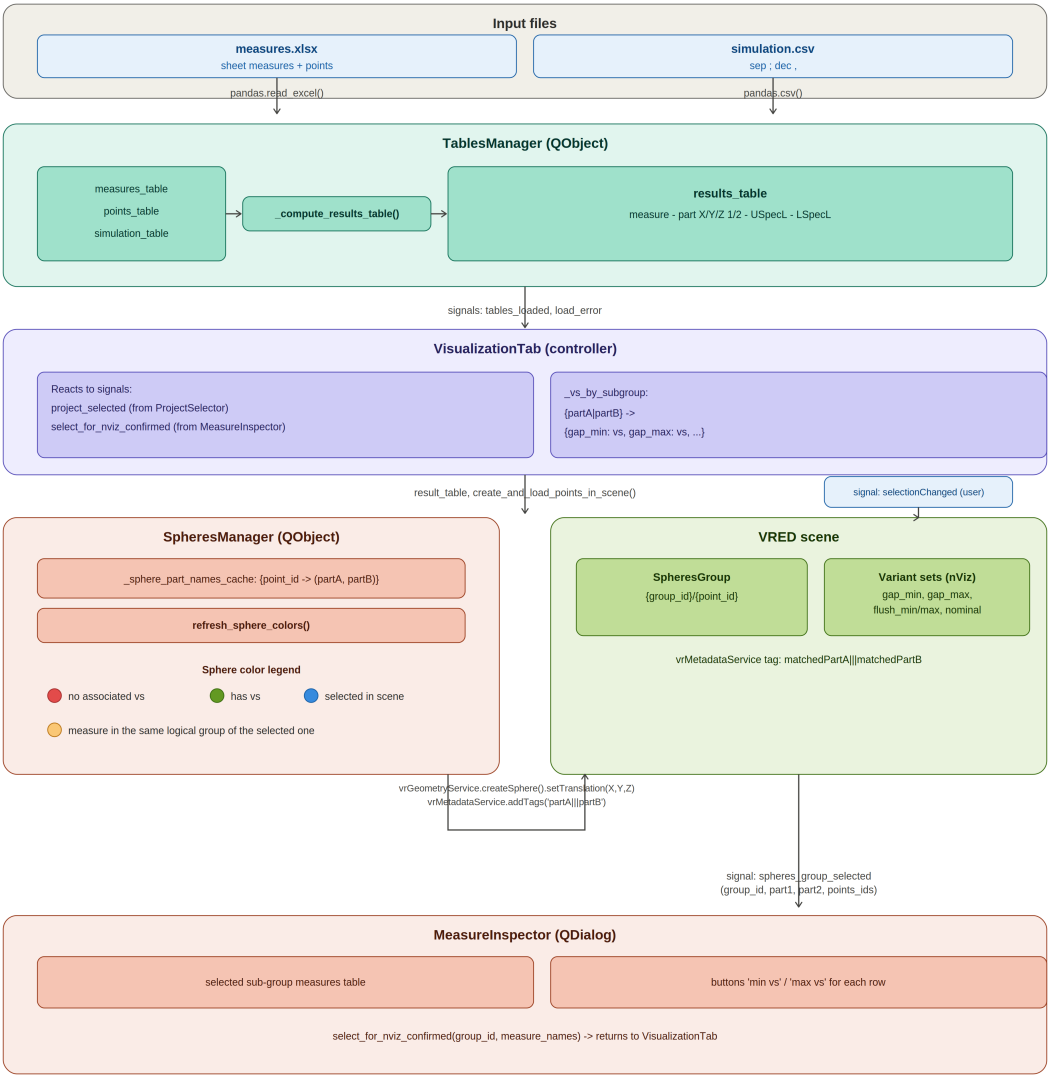


Figure 4.1: High-level architecture of the VR Tolerance Chain Analysis Plugin. The plugin ingests 3DCS output tables, merges and processes the data, visualises it in the VRED scene, and exports a nVIZ-compatible Excel file.

- *pandas* - tabular data manipulation for loading, merging, and filtering the measurement tables.
- *openpyxl* - Excel file reading and writing, used by pandas as its `.xlsx` engine.
- The *VRED Python API* - a collection of service modules exposed by the host application (`vrScenegraph`, `vrGeometryService`, `vrMetadataService`,

vrVariantSets, vrNodeService) that give the plugin full programmatic access to the 3D scene.

Communication model. Components are decoupled through Qt's signal/slot mechanism. TablesManager emits a tables_loaded signal after a successful load; VisualizationTab listens to it and updates the UI state. VRED's own vrNodeService.selectionChanged signal is wired to SpheresManager.on_sphere_selection_ so every interactive click in the 3D scene automatically propagates through the plugin without polling.

4.2 Data Loading and Merging - FR1

FR1 requires the plugin to ingest two heterogeneous files exported by 3DCS and merge them into a single internal table that acts as the single source of truth for all downstream operations.

4.2.1 Input files

The first input is a multi-sheet Excel workbook. Two of its sheets are relevant to the plugin:

- the *Measures* sheet, whose rows each describe one measurement in the tolerance chain, carrying the measure identifier, a textual description, the names of the two geometric features being measured (Name(Feature)_1, Name(Feature)_2), the associated part names (Part Name_1, Part Name_2), and the direction cosines of the measurement axis (I, J, K);
- the *Points* sheet, which maps each feature name to its three-dimensional coordinates (X, Y, Z) in the 3DCS reference frame.

The second input is a semicolon-delimited CSV file containing the results of the 10 000-iteration Monte Carlo simulation: for each measure it provides the estimated lower specification limit (Est.Low) and upper specification limit (Est.High).

4.2.2 Merge pipeline

Loading and merging is the responsibility of the TablesManager class. Its entry point, load_and_compute_project_data(), performs three sequential operations.

First, the three source tables are loaded into pandas DataFrames using pd.read_excel() (with header=1 to skip the 3DCS title row) and pd.read_csv() with the semicolon separator.

Second, the private method `_compute_results_table()` assembles the final table through two successive left joins on feature names: the Measures sheet is joined with the Points sheet once for the first feature and once for the second, attaching the (X_1, Y_1, Z_1) and (X_2, Y_2, Z_2) coordinate pairs. A third merge attaches the Monte Carlo limits from the CSV, joined on the measure identifier. The result is renamed and reordered into a canonical 16-column DataFrame with the following schema:

Column group	**Columns**	**Origin**
Identifier	`Measure`, `Description`	Measures sheet
First feature	`Name(Feature)_1`, `Part Name_1`, `X_1`, `Y_1`, `Z_1`	Measures + Points
Second feature	`Name(Feature)_2`, `Part Name_2`, `X_2`, `Y_2`, `Z_2`	Measures + Points
Direction	`I`, `J`, `K`	Measures sheet
Limits	`USpecL`, `LSpecL`	Simulation CSV

Third, the method emits the `tables_loaded` Qt signal carrying the full table dictionary so that the UI can update without being directly coupled to the loading logic.

Figure 4.2 visualizes the three-way merge.

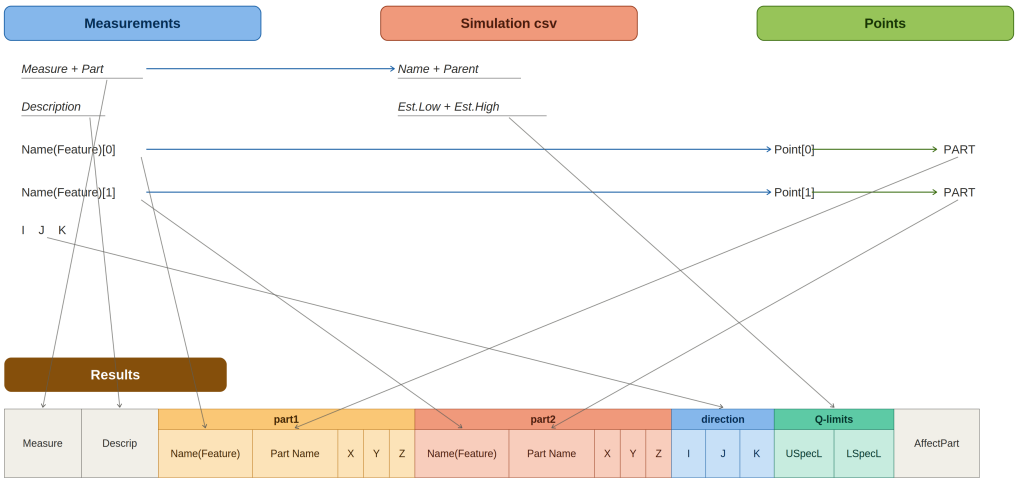


Figure 4.2: Three-way merge pipeline from the 3DCS source tables to the unified results table.

4.3 Geometric Visualisation in VRED - FR2

FR2 requires the plugin to place interactive geometric objects in the VRED scene, one per unique measurement point, positioned at the coordinates extracted from the merged table.

4.3.1 Measure name parsing

Each row in `results_table` identifies a measurement by a name that encodes three pieces of structural information: the group to which the measurement belongs, the specific point within that group, and whether the measurement is of type gap or flush. The parsing uses the following regular expression:

```
^.(\d+)([a-zA-Z])_([BF])$
```

Applied to a name such as `C190a_F`, the three capture groups yield:

- **group_id**: `190` : the integer group identifier, shared by all measurements that belong to the same spatial cluster.
- **point_id**: `190a` : concatenation of the group number and the single-character point suffix; this is the unique key used to name the sphere node in the VRED scene.
- **suffix** : distinguishes *Gap* (`_F`, from the german word *Fuge*) measurements from *Flush* (`_B`, from the german word *Bündigkeit*) ones.

Figure 4.3 illustrates the decomposition.

4.3.2 Scene graph construction

The *create_and_load_points_in_scene()* method of *SpheresManager* trans- lates *points_groups* VRED geometries. It checks the VRED scene graph and looks if a node for points group exists; if not, it creates one via *vrScenegraph.createNode()*. Under this container, a child group node is created for each group_id, and within each group a sphere primitive is created for each point using *vrGeometryService.createSphere()*, positioned at the extracted coordinates and named with the point_id string.

When `SpheresGroup` already exists, for example after a VRED session is re-opened, the method re-maps the existing sphere nodes to the internal dictionaries rather than re-creating them. This avoids geometry duplication and preserves any visual state or metadata that may have been attached to the spheres during a previous session.

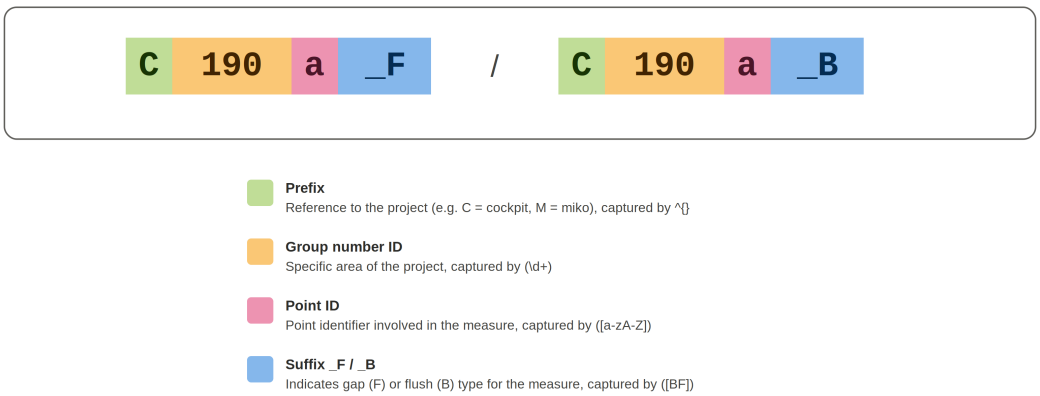


Figure 4.3: Decomposition of a measure identifier according to the naming convention defined in Section 3.2.2.

Visual state. Each sphere carries a default red color (*RGB 255, 80, 80*) indicating that no nViz Variant Set has yet been associated with its subgroup. Once an association is established, the sphere is recolored green (*RGB 0, 200, 0*). This binary color code gives the user an immediate spatial overview of which areas of the vehicle have been fully configured and which require further configuration.

Coordinate conflict detection. If a project is reloaded while its spheres are already present in the scene, the method compares the expected coordinates from the new table with the positions stored in the existing sphere nodes. Any discrepancy exceeding a threshold of 0.001 mm, chosen to match the resolution of 3DCS coordinate output, is treated as a conflict: the offending sphere is visually flagged by doubling its scale, and the user is prompted to choose between keeping the original position and replacing it with the new one.

4.4 Interactive Measure Selection - FR3

FR3 requires the plugin to let the user select measurement points of interest by clicking directly in the VR scene rather than navigating a flat data table.

4.4.1 Scene-driven selection callback

VRED has a *selectionChanged* signal on *vrNodeService* that activates whenever the user changes their scene selection. The plugin connects this signal at startup to *SpheresManager.on_sphere_selection_changed()*, which makes sphere clicks a primary event without needing polling.

When the *on_selection_changed()* fires, the currently selected nodes are captured by the API *vrNodeService.getSelectedNodes()* and a lookup is performed in the internal dictionary *_name_to_sphere*. This is done in response to a problem encountered during the development: VRED returns different Python wrappers for the same underlying C++ scene nodes if accessed through the two APIs: *getChild()* and *getSelectedNodes()*, which makes the comparison unreliable and leads to undesired bugs. With *_name_to_sphere* the correct match is guaranteed.

4.4.2 Context-aware propagation

Once the selected sphere is identified, the plugin determines which *subgroup* it belongs to. A subgroup is defined as the set of all measurement points that share the same pair of 3DCS part names (`Part Name_1`, `Part Name_2`). This partitioning captures the physical reality that gap and flush measurements between a given pair of components form a coherent unit: selecting one point in the subgroup should surface all related measurements simultaneously.

When each sphere is created, an entry in the *_sphere_part_names_cache* dictionary is inserted keyed by *point_id*, as a tuple of two alphabetically sorted strings. This sorting ensures that the pair (A, B) and (B, A) always map to the same key, without making a difference between a point that measures, for example, the distance between A and B and a point that measures the distance between B and A.

When a sphere is clicked, all other spheres in the same `group_id` are recolored: spheres that belong to the same subgroup as the clicked one (i.e. same part pair) are highlighted in blue, while spheres belonging to a different subgroup within the group are highlighted in yellow. This visual distinction helps the engineer immediately understand the spatial distribution of measurement types within a group. You can see a visual representation of this logic in figure 4.4.

Figure 4.5 summarizes the complete interactive selection workflow described above, from the initial sphere click to the population of the MeasureInspector table.

4.4.3 MeasureInspector

After resolving the subgroup, `SpheresManager` calls `MeasureInspector.update_subgroup()`, which repopulates the inspector panel with all measures belonging to the selected subgroup. The inspector organises the measures into two rows, one for gap measurements (`_F` suffix) and one for flush measurements (`_B` suffix), and provides three button slots for each row: `Min`, `Max` and . Each slot is associated with a nVIZ

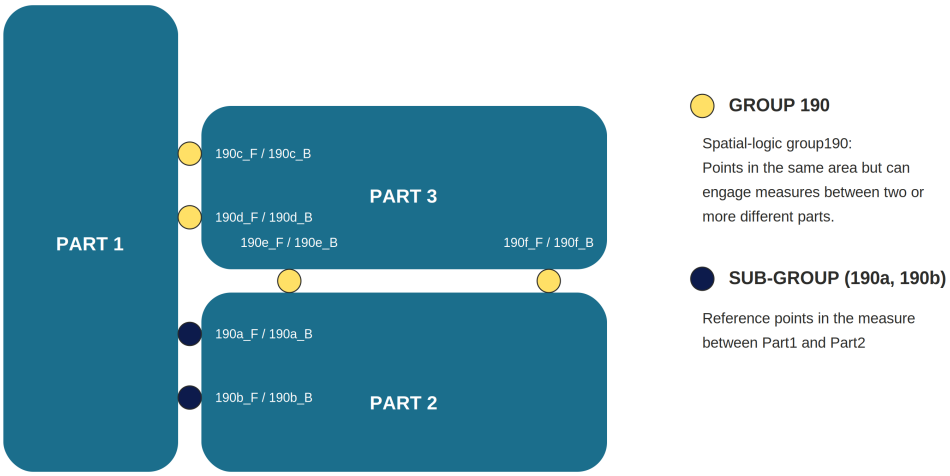


Figure 4.4: Color highlighting when a sphere is clicked.

Variant Set name once the VS association step has been completed; clicking a min or max button calls `vrVariants.selectVariantSet()` to activate the corresponding geometric morphing in the VRED scene. Clicking a second time the nominal vs will be set.

Activating variants with min and max buttons can produce an unexpected ambiguous effect. It can happen that they appear to be inverted. This is not a bug in the code, but a consequence of the measurement's direction vector (`I`, `J`, `K`) with witch nViz calculates the deviation. Being something we can not have control on, the third button serves the job to visually correct this. Clicking it, `_on_swap_min_max()` is activated and the variants are simply inverted. The effect is just a swap in the `_vs_by_subgroup` data structure. Of course the nominal value of the measure is re-set and from that moment clicking min or max buttons will activate the coherent vs.

Moreover, when the selection in scene of a subgroup happens, `MeasureInspector` applies this logic to handle the gap-parallelism measure: if for that specific sub-group exactly two measures of gap are present, through `_find_parallelism_measure()` method it will search in the `result_table` an explicit measure whose name corresponds to the denomination pattern ___*FP* as defined in Section 3.2.2.

Of course, if no matching line is found, only ordinary measures of gap and flush are displayed, otherwise an additional line will be provided for selection, labeled *Parallelism(Gap)*, and it is avaiable to be selected for export.

When this option is exported, the output table for nViz is not populated with

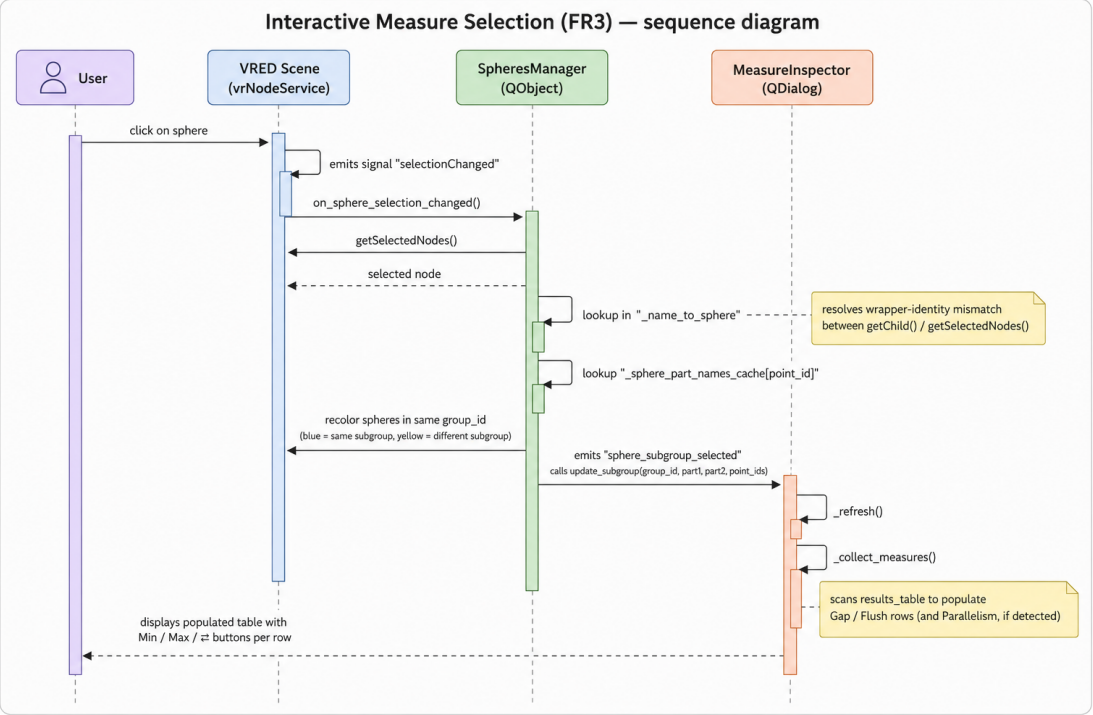


Figure 4.5: Sequence diagram of the interactive measure selection workflow (FR3), from sphere click to MeasureInspector population. Note: this assumes the MeasureInspector dialog is already open; if not, the user must first open it via the "Show Inspector" button.

a third measurement. Instead, the two original Gap measurements are exported with their upper and lower specification limits cross-swapped. This reproduces a wedge-shaped deviation effect while remaining compatible with nViz, which is able to perform only gap and flush morphings.

Since for nViz it is seen as a regular gap measurement, the possibility to export both gap and gap-parallelism is blocked. Consequently, the two analyses must be performed in separate export sessions.

4.5 Part Name Resolution - FR4

FR4 addresses the identity mismatch between the component names used inside the 3DCS model and the node names assigned to the corresponding geometry in the VRED scene. This mismatch is something that frequently happens in industrial context: the siumlation engineer who builds the 3DCS model and performs the Monte Carlo simulations and the VRED scene operator can name differently the

part names in scene since they work almost indipendently.

4.5.1 String similarity matching

The resolution mechanism uses the *Levenshtein distance* [14], the minimum number of single-character insertions, deletions, or substitutions required to transform one string into another, as a quantitative measure of name similarity. Given a 3DCS part name s_1 and a VRED node name s_2, the similarity percentage is computed as:

$$\text{similarity}(s_1, s_2) = \left(1 - \frac{d_L(s_1, s_2)}{\max(|s_1|, |s_2|)}\right) \times 100$$

where d_L denotes the Levenshtein distance and $|\cdot|$ the string length. This normalization maps the raw edit distance onto a $[0, 100]$ scale independent of the absolute length of the names, making scores comparable across name pairs of different lengths.

The distance is computed by the `levenshtein_distance()` function in `utils/string_matche` via a standard dynamic programming recurrence. For strings of lengths m and n, the algorithm fills an $(m+1) \times (n+1)$ table in $\mathcal{O}(mn)$ time. Given that VRED scene node names are typically short (tens of characters) and the set of candidates is bounded by the number of nodes in the scene, this complexity is entirely acceptable in interactive use.

4.5.2 PartNameMatchingDialog

The dialog displays two parallel lists. The left list shows the 3DCS part names taken from the selected subgroup. The right list shows all geometry node names in the VRED scene, sorted in order of similarity to the currently focused 3DCS name so that the best match appears at the top. The comparison is not case-sensitive to minimize the impact of capitalization differences common in CAD datasets.

The user can accept the top choice, select a different entry from the ranked list, or use VRED's wireframe-selection mode to choose a node directly in the 3D scene. Confirmed matches are saved in the name_matching_results dictionary, which has keys as the original 3DCS part-name pairs and corresponding values as the resolved VRED node names for both parts.

4.5.3 Applying the resolution

Once a subgroup has been matched, `VisualizationTab.apply_name_matching_to_dataframe(` overwrites the `Part Name_1` and `Part Name_2` columns in the working copy of the selected measures DataFrame, replacing the 3DCS identifiers with the confirmed

VRED names. The export step described in Section 4.6 then writes these resolved names to the output file, ensuring that nVIZ can locate the correct geometry nodes.

4.6 nVIZ Output Generation - FR5

FR5 requires the plugin to generate an Excel file that conforms to the exact input format expected by the nVIZ Perceived Quality Plugin, with no further manual editing needed.

4.6.1 Output format

The output workbook contains a single sheet (*Foglio1*) in which each row represents one selected measurement. The columns required by nVIZ are a subset of those in `results_table`, with the part names replaced by the resolved VRED node names:

nVIZ column	**Source in plugin**
Part name 1, Part name 2	Resolved VRED node names (FR4 output)
Feature coordinates (X_1, Y_1, Z_1), (X_2, Y_2, Z_2)	`results_table`
Direction vector (I, J, K)	`results_table`
USpecL, LSpecL	Monte Carlo simulation limits

The file is written by pandas' `ExcelWriter` with the *openpyxl* engine. The data rows are written starting at row 2 (`startrow=1`) to match the nVIZ parser's expectation of a one-row header offset.

4.6.2 Sphere tagging

Immediately after the file is written, the export method calls `_tag_spheres()`, which attaches a VRED metadata tag to every sphere that belongs to a matched subgroup. The tag encodes the resolved VRED part-name pair as a pipe-separated, alphabetically ordered string (e.g. `DoorPanel|||Sill`), using `vrMetadataService.addTags()`. This tag serves as the persistence layer described in Section 4.7: it allows the plugin to reconstruct the name-matching state in future sessions without requiring the user to repeat the matching step.

4.7 Session Persistence - FR6

FR6 requires that associations between measurement subgroups and nVIZ Variant Set names survive the closure and reopening of VRED.

4.7.1 Persistence model

VRED provides a metadata tagging service that allows arbitrary string labels to be attached to scene nodes; these tags are saved with the VRED project file. The plugin uses this service as a lightweight key-value store: as described in Section 4.6, each sphere receives a tag encoding the resolved VRED part-name pair for its subgroup. No external file or database is required.

4.7.2 Rebuild on session initialization

When VRED starts, the plugin rebuilds its internal state in two stages:

1. **Sphere map reconstruction.** `SpheresManager.rebuild_sphere_maps_only()` traverses the `SpheresGroup` node in the scene graph, re-populates all internal dictionaries (`sphere_to_group`, `group_to_spheres`, `_name_to_sphere`, `_sphere_part_names_cache`). For each sphere, the method also reads the metadata tags via `vrMetadataService.getTags()`. These tags contain the complete part-name mapping, allowing the plugin to restore its state directly from the VRED project without reloading the original source tables.

2. **Variant Set auto-association.** `_auto_associate_vs_from_tags()` processes all Variant Sets returned by `vrVariantSets.getVariantSets()`. For each VS it determines the *slot* (gap/flush × min/nominal/max) from the VS name via a naming-convention parser, extracts the two part names embedded in the geometry nodes whose names follow the `nVIZpqSwitch_{PartName}` pattern, and constructs the subgroup key as the alphabetically sorted, pipe-joined pair of part names. If this key matches an entry in the sphere metadata cache, the VS is recorded in `_vs_by_subgroup` under the appropriate slot.

The central data structure `_vs_by_subgroup` has the form:

```
{ "partA|partB":  { "gap_min":  vs_name, "gap_nominal":  vs_name,
                    ..., "flush_max":  vs_name } }
```

`_vs_by_subgroup` is shared by reference between `VisualizationTab`, `SpheresManager`, and `MeasureInspector`. Rebuilding it during initialization ensures that all three components operate on the same data and remain synchronized.

4.8 User Interface Architecture

The plugin window is a Qt `QWidget` panel that VRED embeds in its workspace alongside the standard application panels. It is structured as a tab container

(`MainWidget`) currently hosting a single active tab, `VisualizationTab`, which orchestrates all six functional workflows.

Guided workflow structure. The tab presents the user with a top-down sequence of controls that mirrors the intended usage order. At the top, two buttons (*Clear Scene* and *Refresh VS Associations*) handle session restoration (elimination of all the spheres not associated to any vs in the scene). Below them, `ProjectSelector` provides a drop-down of available measurement projects discovered from the input tables directory; an import button allows new project folders to be registered. The *Load Spheres in Scene* button triggers FR2, and *Show Inspector* opens the floating `MeasureInspector` dialog. The name matching workflow panel and the export button are hidden until the user has selected measures from the scene, appearing progressively as prerequisites are met.

Compliance with non-functional requirements. The GUI-only interaction model satisfies NFR3 by making all operations accessible without programming knowledge. The progressive disclosure pattern, hiding panels until they are actionable, reduces cognitive load for inexperienced users. NFR1 (scene non-invasiveness) is addressed by keeping the sphere radius as a configurable parameter (default 5 mm) and grouping all plugin geometry under a single `SpheresGroup` node that can be hidden with one click in the VRED scene tree. NFR2 (performance) is addressed by building all sphere-to-group and name-to-sphere lookup structures as Python dictionaries at load time, so that interactive operations such as selection callbacks and color updates execute in $\mathcal{O}(1)$ rather than requiring scene traversal.

Chapter 5

Use Cases and Workflow

5.1 Introduction to the Chapter

Chapter 4 presented the plugin from a static and structural point of view: it illustrates its architecture, data structures and algorithms one functional requirement at a time. The aim of this chapter is to have a complementary and more *dynamic* view, showing an end-to-end case from the final user's perspective. We transition from an approach which dissects the plugin into how each component works and is build to another one that follows a single use case from the beginning to the end, showing how all the functionalities combine together in a coherent operational workflow and what the user effectively does, sees and obtains at each step.

The intended user for the plugin is a mechanical/quality engineer or designer with some background in dimensional management and tolerancing, but not necessarily in programming. Also 3DCS, the internal convention for the measures and output format for nViz is not a requirement for the usage of the plugin. This is the profile that was supposed to do the manual process described in Chapter 3 but due to the fact that the connection between 3DCS output and nViz input required a detailed knowledge of 3DCS data model, the task was addressed to a very few number of professionals able to perform it. The presented workflow is intended to remove this limitation.

Since the data and the geometry of the project used as a reference for the implementation are protected by a non-disclosure agreement, what is used in this chapter is something made up for demonstrative purposes to be anonymized, but it preserves the exact same structure of a real use case, the denomination conventions and the organization in groups and subgroups, omitting confidential identificators, coordinates and geometries.

Accordingly, the figures in this chapter use synthetic data and neutral placeholder geometry, and are not screenshots of the confidential project scene. This choice does not affect the generality of the workflow, which is identical for any project that conforms to the input specification of Section 3.2.

5.2 Use Case Scenario and Starting Point

The workflow begins at the boundary between the department taking care of the simulation and the one in charge of virtual review. The simulation engineer has already run built and run the 3DCS model, assigned the tolerances and run Monte Carlo simulation with 10.000 iterations.

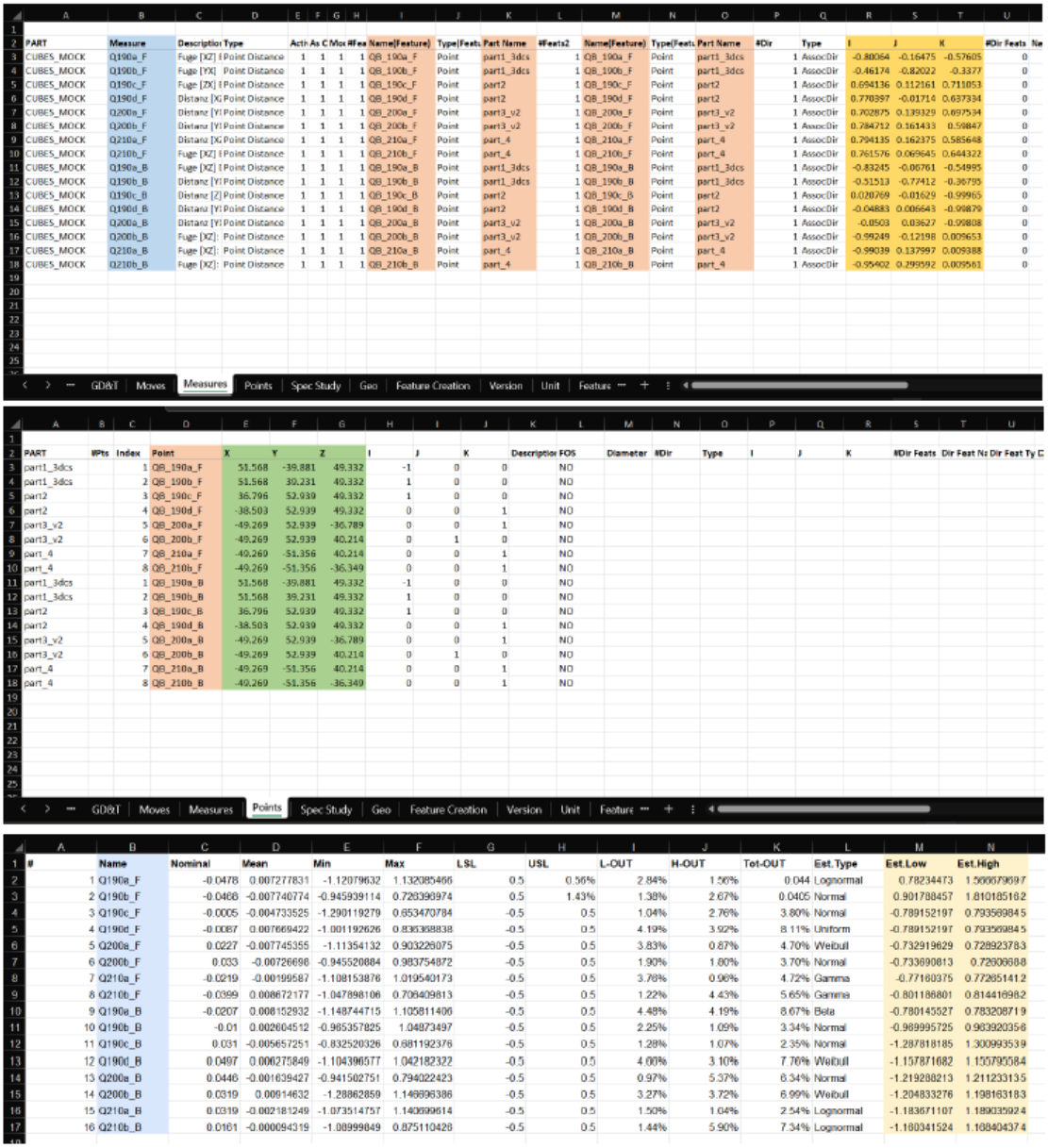
Figure 5.1 shows and highlights the most important columns of the dataset that the user receives.

5.3 Step 1 - Project Selection and Data Loading

The first thing the user does is simply to open Autodesk VRED, load the scene of the vehicle and open the ui of the plugin. In the ui panel the controls are in vertical order, with respect to the order of the operations required. At the top of it there is the `ProjectSelector`, which allows the selection of a project representing the correspondent input tables. It is possible to load a new project by importing the folder containing the 3DCS files without exiting the interface.

Once the project is selected the loading is immediately done. In a single action the plugin executes in a transparent way the data ingestion pipeline described in Section 4.2: `Measures` and `Points` sheets and Montecarlo CSV are red and then the coordinates and statistical limits are paired to each measure by consecutive joins. The user is not exposed to any part of this operations: the only knowledge is that the project has succesfully been loaded and the subsequent controls of the panel are now enabled. The previous heterogeneous and multi-file structure which required manual navigation is now reduced to a single representation behind the interface.

Figure 5.2 captures the VRED scene and the moment the user selects the project. As said in 3.2 the selected project is a folder containing the two 3DCS files, the ones illustrated in the previous Section 5.2.



PART	Measure	Description	Type	Name(Feature)	Type(Feats	Part Name	#Feats2	Name(Feature)	Type(Feats	Part Name	#Dir	Type
CUBES_MOCK	Q190a_F	Fuga [XZ]	Point Distance	QB_190a_F	Point	part1_3dcs	1	QB_190a_F	Point	part1_3dcs	1	AssocDir
CUBES_MOCK	Q190b_F	Fuga [YX]	Point Distance	QB_190b_F	Point	part1_3dcs	1	QB_190b_F	Point	part1_3dcs	1	AssocDir
CUBES_MOCK	Q190c_F	Fuga [ZX]	Point Distance	QB_190c_F	Point	part2	1	QB_190c_F	Point	part2	1	AssocDir
CUBES_MOCK	Q190d_F	Distanz [X	Point Distance	QB_190d_F	Point	part2	1	QB_190d_F	Point	part2	1	AssocDir
CUBES_MOCK	Q200a_F	Distanz [Y	Point Distance	QB_200a_F	Point	part3_v2	1	QB_200a_F	Point	part3_v2	1	AssocDir
CUBES_MOCK	Q200b_F	Distanz [Y	Point Distance	QB_200b_F	Point	part3_v2	1	QB_200b_F	Point	part3_v2	1	AssocDir
CUBES_MOCK	Q210a_F	Distanz [X	Point Distance	QB_210a_F	Point	part_4	1	QB_210a_F	Point	part_4	1	AssocDir
CUBES_MOCK	Q210b_F	Fuga [XZ]	Point Distance	QB_210b_F	Point	part_4	1	QB_210b_F	Point	part_4	1	AssocDir
CUBES_MOCK	Q190a_B	Fuga [XZ]	Point Distance	QB_190a_B	Point	part1_3dcs	1	QB_190a_B	Point	part1_3dcs	1	AssocDir
CUBES_MOCK	Q190b_B	Distanz [Y	Point Distance	QB_190b_B	Point	part1_3dcs	1	QB_190b_B	Point	part1_3dcs	1	AssocDir
CUBES_MOCK	Q190c_B	Distanz [Z	Point Distance	QB_190c_B	Point	part2	1	QB_190c_B	Point	part2	1	AssocDir
CUBES_MOCK	Q190d_B	Distanz [Y	Point Distance	QB_190d_B	Point	part2	1	QB_190d_B	Point	part2	1	AssocDir
CUBES_MOCK	Q200a_B	Distanz [Y	Point Distance	QB_200a_B	Point	part3_v2	1	QB_200a_B	Point	part3_v2	1	AssocDir
CUBES_MOCK	Q200b_B	Fuga [XZ]:	Point Distance	QB_200b_B	Point	part3_v2	1	QB_200b_B	Point	part3_v2	1	AssocDir
CUBES_MOCK	Q210a_B	Fuga [XZ]:	Point Distance	QB_210a_B	Point	part_4	1	QB_210a_B	Point	part_4	1	AssocDir
CUBES_MOCK	Q210b_B	Fuga [XZ]:	Point Distance	QB_210b_B	Point	part_4	1	QB_210b_B	Point	part_4	1	AssocDir

GD&T | Moves | Measures | Points | Spec Study | Geo | Feature Creation | Version | Unit | Feature

PART	#Pts	Index	Point	X	Y	Z	I	J	K	Description	FOS	Diameter	#Dir	Type
part1_3dcs		1	QB_190a_F	51.568	-39.881	49.332	-1	0	0		NO			
part1_3dcs		2	QB_190b_F	51.568	39.231	49.332	1	0	0		NO			
part2		3	QB_190c_F	36.796	52.939	40.332	1	0	0		NO			
part2		4	QB_190d_F	-38.503	52.939	49.332	0	0	1		NO			
part3_v2		5	QB_200a_F	-49.269	52.939	-36.789	0	0	1		NO			
part3_v2		6	QB_200b_F	-49.269	52.939	40.214	0	1	0		NO			
part_4		7	QB_210a_F	-49.269	-51.356	40.214	0	0	1		NO			
part_4		8	QB_210b_F	-49.269	-51.356	-36.349	0	0	1		NO			
part1_3dcs		1	QB_190a_B	51.568	-39.881	49.332	-1	0	0		NO			
part1_3dcs		2	QB_190b_B	51.568	39.231	49.332	1	0	0		NO			
part2		3	QB_190c_B	36.796	52.939	40.332	1	0	0		NO			
part2		4	QB_190d_B	-38.503	52.939	40.332	0	0	1		NO			
part3_v2		5	QB_200a_B	-49.269	52.939	-36.789	0	0	1		NO			
part3_v2		6	QB_200b_B	-49.269	52.939	40.214	0	1	0		NO			
part_4		7	QB_210a_B	-49.269	-51.356	40.214	0	0	1		NO			
part_4		8	QB_210b_B	-49.269	-51.356	-36.349	0	0	1		NO			

GD&T | Moves | Measures | Points | Spec Study | Geo | Feature Creation | Version | Unit | Feature

#	Name	Nominal	Mean	Min	Max	LSL	USL	L-OUT	H-OUT	Tot-OUT	Est.Type	Est.Low	Est.High
1	Q190a_F	-0.0478	0.007277831	-1.12079632	1.132085466	-0.5	0.5	0.56%	2.84%	1.56%	0.044	Lognormal	0.78234473
2	Q190b_F	-0.0468	-0.007740774	-0.945939114	0.728390974	-0.5	0.5	1.43%	1.38%	2.67%	0.0405	Normal	0.901788457
3	Q190c_F	-0.0005	-0.004733525	-1.290119279	0.853470784	-0.5	0.5	1.04%	2.76%	3.80%	Normal	-0.789152197	0.793569845
4	Q190d_F	-0.0087	0.007669422	-1.001192626	0.836368838	-0.5	0.5	4.19%	3.92%	8.11%	Uniform	-0.789152197	0.793569845
5	Q200a_F	0.0227	-0.007745355	-1.11354132	0.903228075	-0.5	0.5	3.83%	0.87%	4.70%	Weibull	-0.732919629	0.728923783
6	Q200b_F	0.033	-0.00726698	-0.945520884	0.983754872	-0.5	0.5	1.90%	1.80%	3.70%	Normal	-0.733690813	0.72809888
7	Q210a_F	-0.0219	-0.00199587	-1.108153876	1.019540173	-0.5	0.5	3.78%	0.96%	4.72%	Gamma	-0.77180375	0.772851412
8	Q210b_F	-0.0399	0.008672177	-1.047898106	0.706409813	-0.5	0.5	1.22%	4.43%	5.65%	Gamma	-0.801186801	0.814416982
9	Q190a_B	-0.0207	0.008152932	-1.148744715	1.105811406	-0.5	0.5	4.48%	4.19%	8.67%	Beta	-0.780145527	0.783208719
10	Q190b_B	-0.01	0.002604512	-0.985357825	1.04873497	-0.5	0.5	2.25%	1.09%	3.34%	Normal	-0.989995725	0.963920356
11	Q190c_B	0.031	-0.005657251	-0.832520326	0.681192376	-0.5	0.5	1.28%	1.07%	2.35%	Normal	-1.287818185	1.300993539
12	Q190d_B	0.0497	0.006275849	-1.104396577	1.042182322	-0.5	0.5	4.60%	3.10%	7.76%	Weibull	-1.157871682	1.155795584
13	Q200a_B	0.0446	-0.001639427	-0.941502751	0.794022423	-0.5	0.5	0.97%	5.37%	6.34%	Normal	-1.219288213	1.211233135
14	Q200b_B	0.0319	0.00914632	-1.28862859	1.146696386	-0.5	0.5	3.27%	3.72%	6.99%	Weibull	-1.204833276	1.198163183
15	Q210a_B	0.0319	-0.002181249	-1.073514757	1.140699614	-0.5	0.5	1.50%	1.04%	2.54%	Lognormal	-1.183671107	1.189035924
16	Q210b_B	0.0161	-0.000094319	-1.08999849	0.875110428	-0.5	0.5	1.44%	5.90%	7.34%	Lognormal	-1.180341524	1.188404374

Figure 5.1: Starting point of the workflow: the two heterogeneous 3DCS exports delivered by the simulation department. In order here are displayed: Measures, Points and Simulation tables.

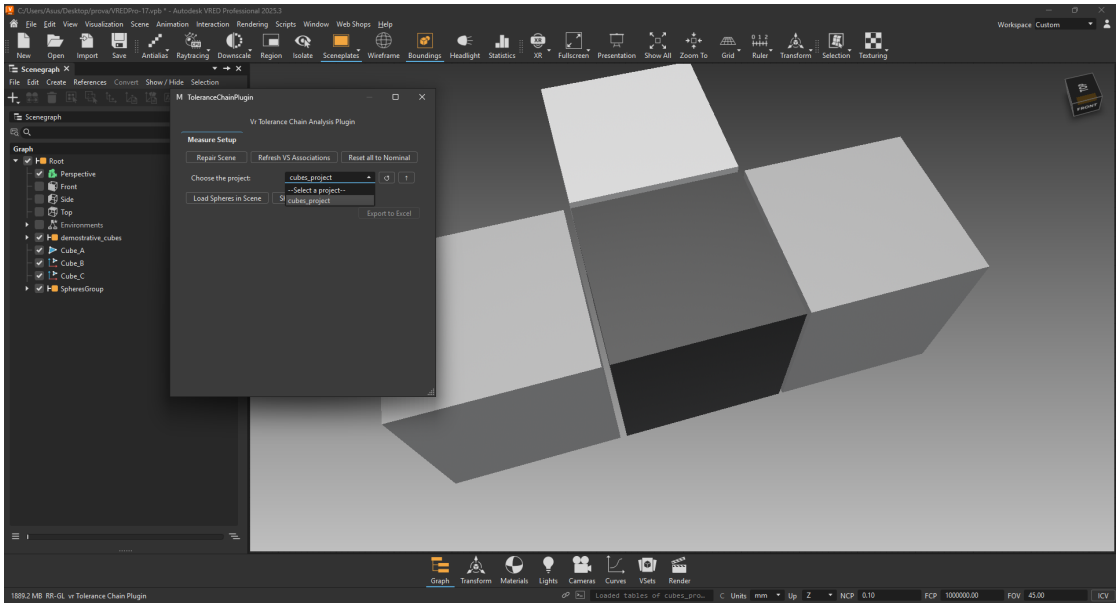


Figure 5.2: Project selection and data loading through the plugin panel (FR1).

5.4 Step 2 - Generating the Measure-Point Spheres in the Scene

With all the data loaded, the user can now click on *Load Spheres in Scene*. The plugin instantiates an interactive sphere for each unique measure point, centered in the coordinates extracted during the previous step. All the spheres are organized in the scenegraph in a container node that has a child for each logical group, as described in Section 4.3. Initially all the spheres are rendered with red color, signaling that there are still no variant sets associated to its subgroup.

This step really adds a qualitative difference, more immediate with respect to the previuos workflow. In this last one, the user was forced to reason on measure points as rows in abstract tables, outside of the vehicle geometry, while with the spheres the user acquires a direct spacial reference: he/she now is able to see where is is placed on the real geometry and can immediately understand how the measures are grouped. Moreover, the chosen shade of red allows to understand which areas still requires configuration at a glace.

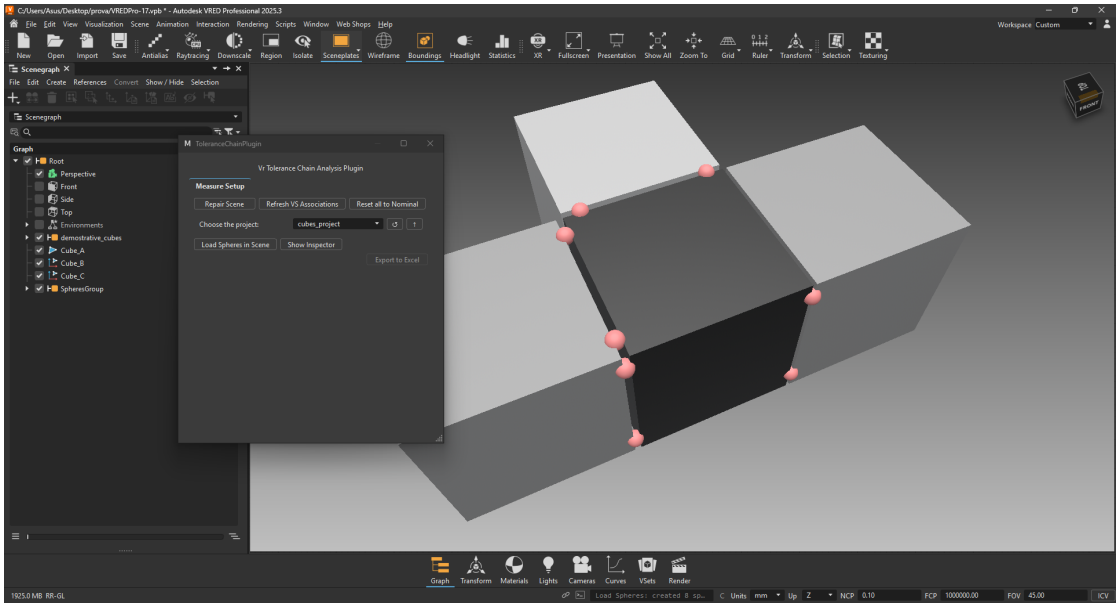


Figure 5.3: Measure-point spheres instantiated in the VRED scene after loading (FR2).

5.5 Step 3 - Interactive Selection and Measure Inspection

The user can now inspect the scene and he/she clicks on a sphere belonging to group 190. The context-aware propagation described in Section 4.4 has an immediate effect: all the spheres belonging to the same subgroup become blue, while the remaining spheres of the same group are now colored in yellow. With a single click the user understands the extention of the selected subgroup and the spacial relationship with the neighbor measures without consulting any table.

Even without keeping the main ui panel open it is possible to start the selection of the measures for nViz flow. By selecting a sphere the `MeasureInspector` is automatically opened and populated with the available measures for the selected subgroup. For each row there are the controls *Min*, *Max*, *Swap*. When there is an association with a variant set, clicking on one of these controls will activate the corresponding geometrical morphing in scene. This means that the user is able to see all the different configurations and clicking a second time on the button the nominal value will be restored.

When the direction vector of he measure makes the morphing of gap min and max appear inverted, the user can easily correct it visually with the *Swap* control,

which simply swaps the associations without any edit to the corresponding data.

Group 190 would lend itself to illustrating the management of the composite measurement of gap-parallelism. The configuration is idle because the subgroup, (`190c`, `190d`) for example, contains exactly two measures of gap. On this measure, the mechanism would work like this: at the moment of the selection, the inspector would look up into the `results_table` for an explicit row of gap-parallelism corresponding to the pattern `__FP`, so for a row with the measure `Q190cd__FP`. If this row is founded, in the inspector an additional `Parallelism(Gap)` row is appended. In this way the user would be able to select also this other measurement, with the only limitation that he/she cannot export gap and gap-parallelism at the same time for the reasons explained in 4.4: the gap-parallelism export is generated from a composite situation from the regular gap and nViz plugin would not be able to handle both cases at the same time.

The reason the gap-parallelism is not included explicitly in this walkthrough is that it has been implemented as a feature, but it still needs additional analysis and validation to be effectively entirely considered.

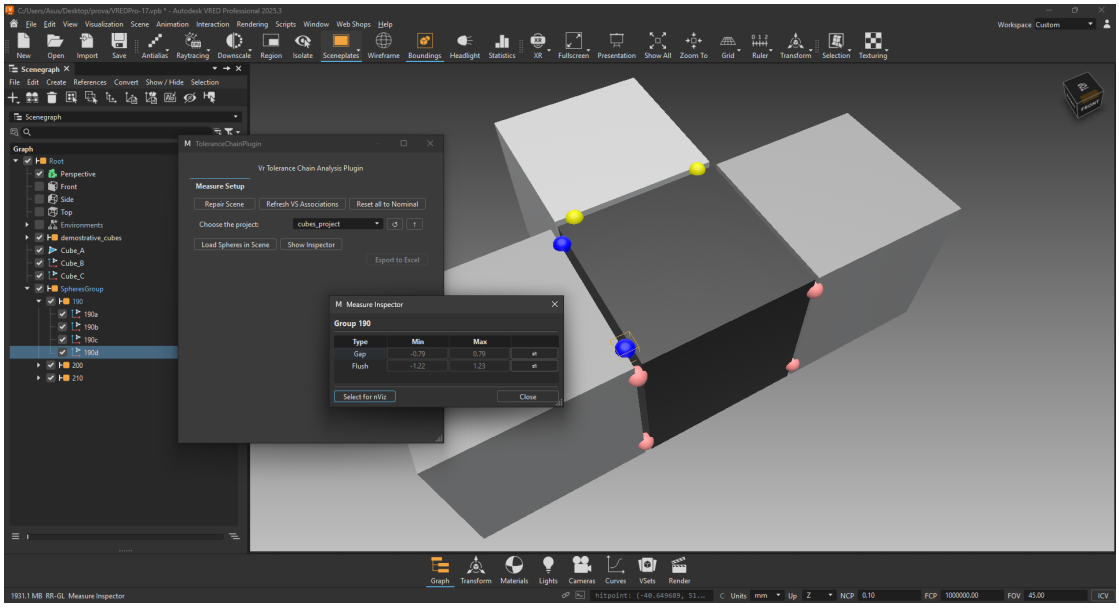


Figure 5.4: Context-aware highlighting triggered by a sphere selection (FR3) and MeasureInspector populated for the selected subgroup.

5.6 Step 4 - Part Name Resolution

Before the selected measures can be exported, the identity mismatch between scene and 3DCS identifiers must be solved, as required by FR4. After the click on the button *Select for nViz* in the main ui panel appears an additional component to guide the user in doing so. For each measure he/she needs to open the `PartNameMatchingDialog` and solve the matching. In this dialog, for each of the two scene parts involved, it shows the 3DCS name and below an interactive list with all the scene node names in decreasing order of similarity. Here, by clicking one element of the list, through `Wireframe` mode automatically activated in the scene, the corresponding scene node is highlighted. The user can also select in the scene a node and in the list with which he/she interacted last, that node will be selected. This is necessary because it can happen that the similarity score between the 3DCS name and the scene node name can be low due to a high numer of characters is the scene node name such as underscores or timestamps.

After confirming, the matchings are memorized and they will substitute the 3DCS identifiers in the exported table for nViz. This is mandatory for nViz, which would be unable to perform the morphing otherwise. Moreover, this semi-automatic approach allows to preserve the needed human control in a critical step for the correctness of the morphing and at the same time deleting almost completely the manual search needed before.

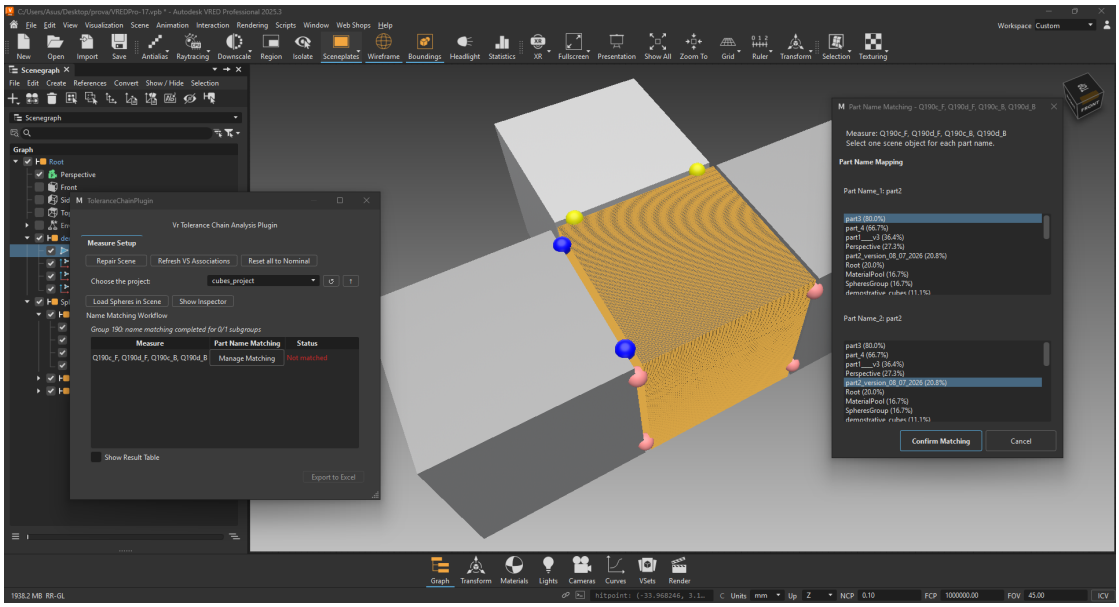


Figure 5.5: Semi-automatic resolution of 3DCS-to-VRED part names (FR4).

5.7 Step 5 - Exporting the nVIZ Input File

For the selected subgroup the user is now able to perform the export of the file for nVIz. The plugin automatically generates the input file for nViz in the exact format required, as described in Section 3.2.4 in a location chosen by the user. Any additional edit is not required and the user is ready to use it with nViz plugin.

When the successful export happens, the subgroup spheres are tagged with a string containing the information of both the names of the part names involved in the measure as chosen in the matching phase. This will be crucial for associating to the spheres the correct variant sets that will be generated by nViz.

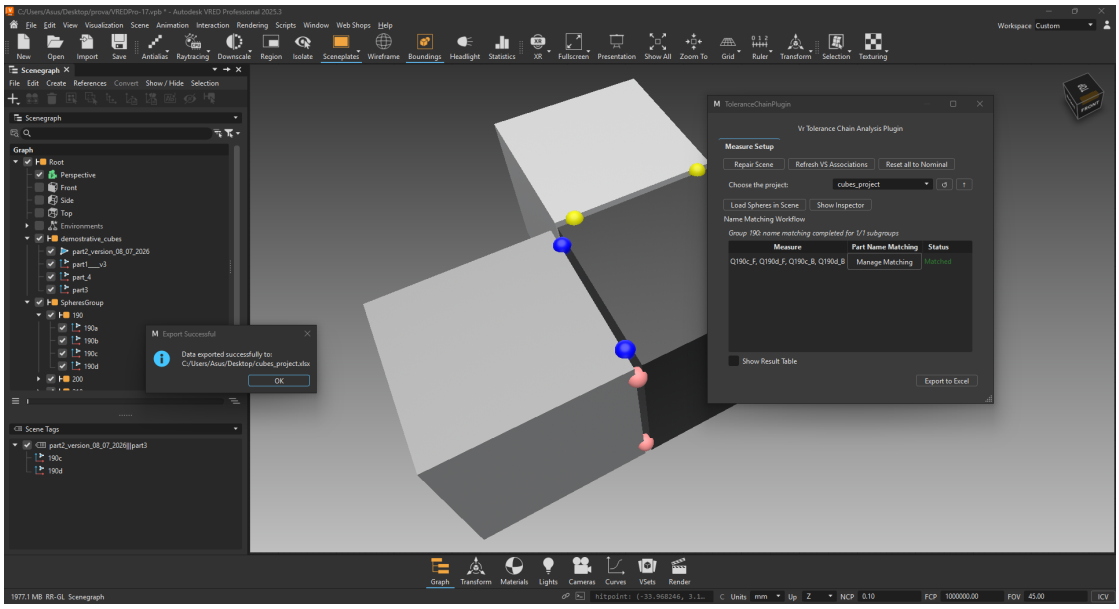


Figure 5.6: Successfull export of nVIZ-ready file and metadata tags added for the corresponding group.

5.8 Step 6 - Consuming the File in nVIZ and Reviewing in VR

The generated file is now handed to nViz Perceived Quality Plugin, which imports it, modifies the scene substituting for each node involved in a measure with a switch node. This switch node will contain a copy of the original node, which represents the nominal value and additional nodes with the configuration for min and max. It

also generates variant sets, which will select from the switch the corresponding node for the configuration described by it. For example a variant set can be associated with gap max between Part1 and Part2. By activating that vs in the scene will be selected only the nodes from the switch of Part1 and Part2 corresponding to the gap max between those two parts.

In this step the spheres with associated variant sets are colored in green and clicking on one of it in the `MeasureInspector` that pops up the buttons for the corresponding associations are now enabled. Each button is labeled with a number, which is the mean between all the measures of each point.

This phase is where the value of the entire pipeline becomes tangible: the output of the tolerancing simulation in 3DCS is now navigable as a visual experience both spacial and interactive. The user is able to see the visual aspect of the assembly and evaluate the quality of the gap and flush configuration to decide if it is acceptable or not.

In terms of gap integration described in Section 2.5, this is the point where the bond between tolerance simulation and perceived quality evaluation in VR is finally closed, without manual data preparation by the reviser.

While Figure 5.7 illustrates the mechanism on abstract figures and the walk-through is based on synthetic geometries it is significant to give an idea of the result of this phase on a real automotive surface. Figure 5.8 illustrates the real cases of morphing on an actual body panel joint in the configurations of: minimum gap and flush combined, nominal and maximum gap and flush combined.

5.9 Step 7 - Reopening the Session and Persistence

A dimensional review is rarely completed in a single session. When the user closes VRED and re-opens it in a later time, the plugin at the moment of the opening of the saved scene, will recreate its internal state from metadata tags, as described in Section 4.7: internal maps of the spheres are reconstructed, assocaition names are retrieved from tags and variant sets are re-associated to the corresponding subgroups.

Previously configured subgroups will be green and all non-configured spheres are simply deleted from scene, since without the internal result table representation

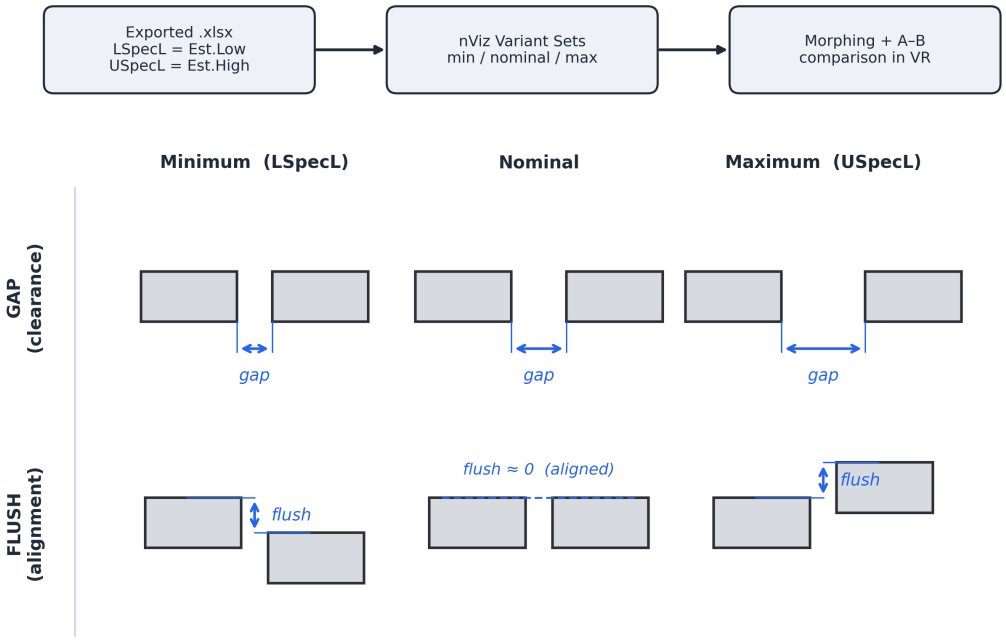


Figure 5.7: Closing the loop: nVIZ variant sets and gap/flush morphing driven by the exported file, reviewed in VR.

they would be useless. The practical consequence of this phase is that the review becomes an *incremental* process instead of a *monolithic* one. This is particularly useful for large-dimensional assemblies with hundreds of measure points, where now they can be configured in progressive and multiple sessions and the state of the work is always immediately visible, thanks to the green color of the already-configured spheres. This supports volumes and working methods described by the non-functional requirements in Chapter 3.

5.10 Workflow Summary

The previous sections followed an over-simplified subgroup of cubes, virtually representing three vehicle parts, through the entire workflow of the plugin, from the receiving of the two 3DCS files to the immersive VR review of the geometry that has been morphed. The summarizing Table 5.1 consolidates this experience associating each step to the action performed by the user and the response to the functional requirement that it satisfies. The Figure 5.10 represents the same

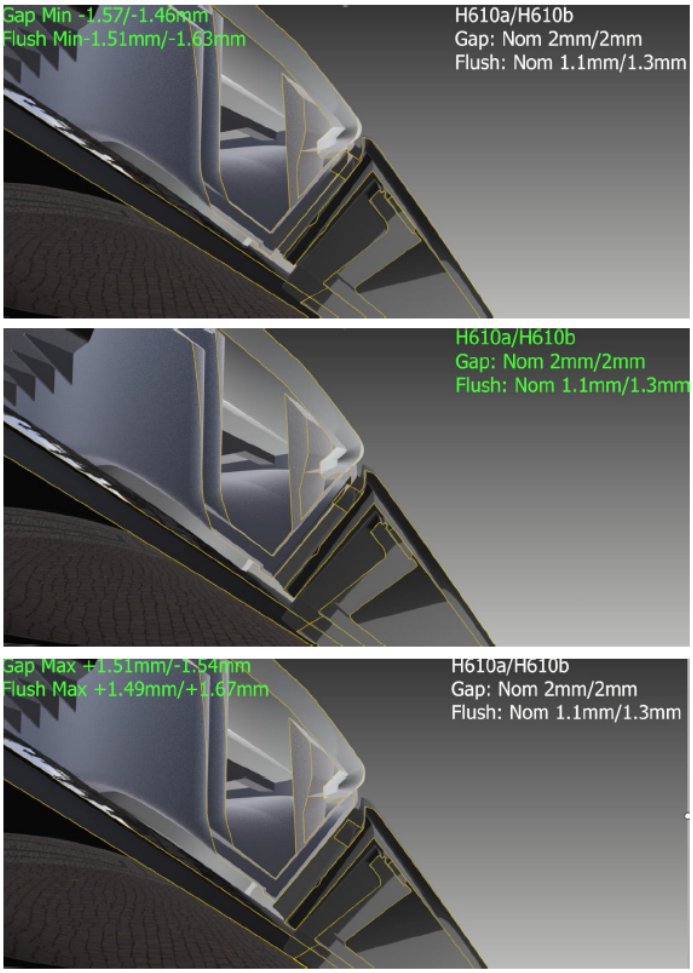


Figure 5.8: Real-world illustration of the nVIZ gap and flush morphing on an automotive body-panel joint. From top to bottom: minimum, nominal and maximum configurations (combined gap and flush).

sequence, but as an end-to-end diagram.

Overall, the workflow demonstrates the central thesis of this work in operational terms. What was a manual error-prone operation, accessible only to few skilled people, is now a guided sequence of interaction performed entirely through a graphical interface. Every step that required specialized knowledge, is now automated or reduced to a confirm-interaction in the virtual scene. To the user is left the freedom to focus on the tasks that really requires human judgment: decide which measure to focus on and to evaluate its dimensional behavior.

Table 5.1: End-to-end workflow: user action, plugin response, and satisfied requirement for each step.

Step	User action	Plugin response	FR
1	Select the project and load the data.	Ingests and merges the two 3DCS files into the unified results table.	FR1
2	Load the spheres in the scene.	Instantiates one red sphere per measure point, grouped by logical group.	FR2
3	Click a sphere and inspect the measures.	Highlights the subgroup (blue) and the other subgroups (yellow); populates the MeasureInspector, including the parallelism entry.	FR3
4	Resolve the part names.	Ranks scene nodes by similarity and stores the confirmed 3DCS-to-VRED matches.	FR4
5	Export the selected measures.	Writes the nVIZ-ready file, tags the spheres, and recolours them green.	FR5
6	Import the file in nVIZ and review in VR.	(External) nVIZ generates the variant sets and morphs the geometry for A-B comparison.	—
7	Reopen the project later.	Rebuilds the state from the metadata tags; configured subgroups reappear green.	FR6

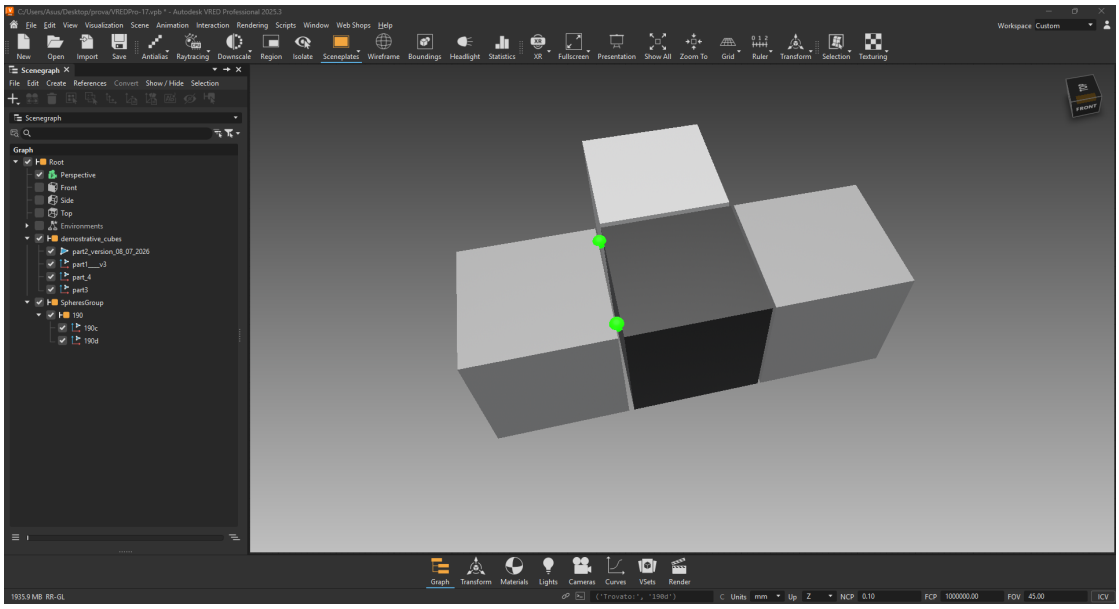


Figure 5.9: State restoration on session reopening (FR6).

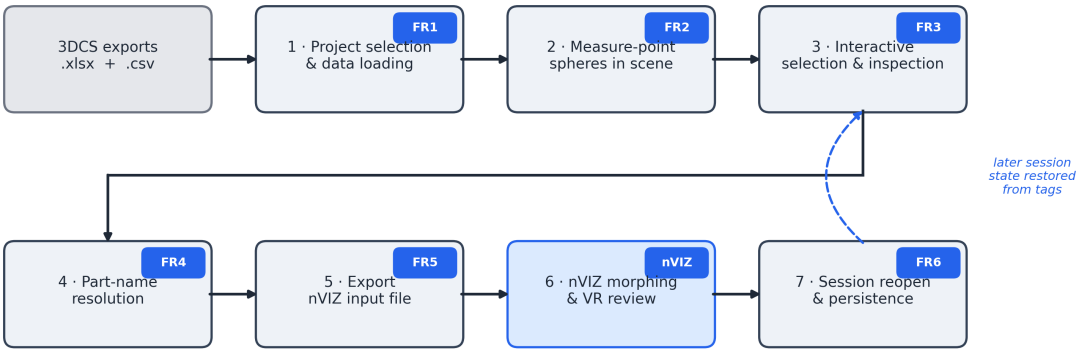


Figure 5.10: End-to-end diagram of the plugin workflow.

Chapter 6

Results, Discussion and Conclusions

This final chapter evaluates the work presented in the thesis. It first verifies that the plugin satisfies the objectives and the requirements from which it was derived, then discusses its practical significance and the nature of the improvement it introduces. The current limitations are stated openly, the directions along which the tool can evolve are outlined, and the chapter closes with a reflection on the contribution as a whole.

6.1 Fulfilment of Objectives and Requirements

Being a design- and implementation-oriented work rather than an experimental one, the primary result of this thesis is the artefact itself: a functioning VRED plugin that automates the pipeline from raw 3DCS output to a nVIZ-ready input file, together with the interactive, scene-based workflow that surrounds it. The correctness of this result is established by two facts. First, each of the five objectives defined in Chapter 1 has been realised. Second, the functional and non-functional requirements formalised in Chapter 3 have all been implemented, as traced in Table 3.1 to the corresponding sections of Chapter 4, and demonstrated operationally, end to end, in the walkthrough of Chapter 5.

Table 6.1 closes the loop on the objectives, associating each of them with the mechanism that satisfies it and the point at which it is shown in action.

Taken together, these results show that the operational bottleneck identified at the outset has been removed: the heterogeneous, manual bridging between 3DCS and nVIZ has become a single, guided sequence of interactions carried out entirely through a graphical interface, and the resulting file is consumed by nVIZ without

Table 6.1: Fulfilment of the thesis objectives defined in Chapter 1.

Objective	**Realised through**	**Demonstrated in**
Data pipeline automation	FR1 (ingestion and merge) and FR5 (nVIZ-ready export)	Ch. 5, Steps 1 and 5
Spatial measurement visualisation	FR2 (measure-point spheres)	Ch. 5, Step 2
Context-aware interactive selection	FR3 (scene-driven, group/-subgroup propagation)	Ch. 5, Step 3
Part name resolution	FR4 (Levenshtein string similarity)	Ch. 5, Step 4
Usability and integration	NFR3 and the GUI embedded in VRED	Ch. 5 (whole)

any further editing.

6.2 Discussion

A conventional quantitative before/after comparison of the data-preparation process could not be produced, and the reason for this is, in itself, the most eloquent result of the work. Before the plugin, the manual preparation could not be carried out beyond a very small number of measurement points, and even those few were assembled with considerable effort by a single person, who possessed the rare combination of skills required to interpret the 3DCS data model and to construct the nVIZ input by hand. The limiting factor was therefore not the time required for each measurement, but the ability to carry out the task at scale, as the process had not yet become a repeatable and transferable practice.

Seen from this perspective, the plugin represents more than a simple improvement in speed: it fundamentally changes the nature of the task. Rather than relying on specialised expertise, the workflow is reduced to a straightforward graphical interaction. Data are automatically imported and merged, measurements are identified, part names are reconciled, and the output file is assembled with minimal user intervention. In most cases, the operator's role is limited to confirming the results within the virtual environment. As a consequence, the task is no longer confined to a single specialist. An engineer without a programming background or a detailed knowledge of the 3DCS and nVIZ internal conventions can now cover the full set of measures of interest, in a reproducible way and without depending on a single key person. This outcome is, in practical terms, the empirical validation of the usability requirement NFR3: the qualitative evidence collected in the industrial context is that the workflow is genuinely operable by its intended audience.

Beyond the immediate industrial benefit, the work fills the specific gap identified in Section 2.5: the absence of an automated bridge between the statistical output of a tolerance-chain analysis and an interactive VR environment for perceived-quality evaluation. By delivering a result table that nVIZ can consume directly, the plugin closes the link between tolerance simulation and immersive design review, and does so within the broader trajectory of the Digital Twin discussed in Section 2.4: it makes a statistical, data-driven prediction of gap and flush behaviour navigable and interpretable as a visual, spatial experience, moving dimensional-quality information out of specialist reports and into the design decision process, before any physical prototype exists.

6.3 Limitations

The tool, in its current state, has a number of limitations that it is important to state explicitly.

- **Gap-parallelism still to be validated.** The handling of the composite gap-parallelism measurement has been implemented, but it requires further testing and validation before it can be considered fully reliable, and for this reason it was not included in the walkthrough of Chapter 5.
- **Flush-parallelism not yet available.** The corresponding flush-parallelism measurement is not implemented. As discussed in Section 6.4, however, it is a natural extension of the same mechanism once the gap-parallelism case has been consolidated.
- **Semi-automatic part-name resolution.** The matching between 3DCS identifiers and VRED node names is assisted by string similarity but still requires human confirmation. This is a deliberate safety choice, since the correctness of the morphing depends on it, but it means the step is not fully automatic.
- **The plugin is one link in the chain.** The plugin is one component of a broader workflow. It generates the input file in the required format, while geometric morphing and the creation of variant sets are handled by nVIZ. Although the overall pipeline is complete, the transfer of the generated file to nVIZ still requires a manual step.
- **Scope of the current implementation.** The plugin operates on one project at a time and extracts data exclusively from the *Measures* and *Points* sheets; its responsiveness on very large assemblies has not been formally benchmarked, and the resolution of coordinate conflicts on reload is left to the user.

6.4 Future Work

Several directions of development follow naturally from the present work.

- **Full automation through the nVIZ APIs.** Italdesign is already in contact with nVIZ GmbH and has requested access to the plugin's programming interfaces, with the aim of triggering the nVIZ import and the generation of the variant sets directly from the plugin's graphical interface. This would remove the last manual hand-off in the pipeline and bring the automation of the entire chain - from the raw 3DCS exports to the morphed VR scene

— to a nearly complete level, driven from a single interface. It is the most immediate and impactful evolution of the tool.

- **Validation of gap-parallelism and extension to flush-parallelism.** Once the gap-parallelism handling has been thoroughly tested and validated, the implementation of flush-parallelism becomes natural and immediate: it is an extension of the very same mechanism and would require only limited additional effort, completing the coverage of the composite measurements.

- **Broader scope and validation.** Natural extensions include batch and multi-project processing, a more systematic usability study with a larger group of engineers, and the application of the workflow to assemblies beyond the interior case used here.

6.5 Conclusions

This thesis started from a concrete operational problem: at Italdesign, the rich statistical output of a 3DCS tolerance-chain analysis could not easily reach the immersive VR environment in which perceived quality is assessed, because the data preparation required by the nVIZ Perceived Quality Plugin was a manual, error-prone task that only a very few specialists were able to perform. The work presented here addressed this problem with a custom VRED plugin that automates the complete upstream pipeline: it ingests and merges the heterogeneous 3DCS exports, visualises each measurement point as an interactive sphere in the scene, lets the engineer select and inspect measures directly in the virtual environment, resolves the mismatch between 3DCS and VRED naming, and produces a file that nVIZ can consume without further editing.

The result is more than a convenience. By turning a task that depended on scarce, non-transferable expertise into a guided interaction accessible to any engineer, the plugin does not merely make the process faster: it makes it possible at scale, and repeatable. In doing so it closes, in a practical and industrially usable way, the missing link between tolerance simulation and VR-based design review, and it brings dimensional-quality information into the design phase, where decisions can still be taken cheaply, before any physical part is produced.

On a more personal note, the value of this work also lies in what it demonstrates about the direction of digital product development: that the barrier between highly technical simulation output and human, visual judgement can be lowered, and that doing so can give more people a voice in decisions that were previously the domain of a few. With the automation of the nVIZ hand-off already on the horizon, the

tool developed in this thesis is a first, working step along that path, and one on which it is easy to imagine building further.

References

[1] MG Techworks. *Gap and Flush Analysis: Complete Guide for Automotive Engineers*. Technical company documentation on automotive gap and flush. June 2026. URL: https://mg-techworks.com/blogs/mastering-gap-and-flush-automotive-interiors (visited on 07/05/2026) (cit. on p. 6).

[2] GD&T Basics. *Parallelism*. Accessed for the geometric definition of parallelism. URL: https://www.gdandtbasics.com/parallelism/ (visited on 07/05/2026) (cit. on p. 6).

[3] Vinay Kumar Singh, Ved Prakash Dewangan, Rahul Kumar, and Amar Deep. «Dimensional Management in Automotive Vehicle Design & Development for Perceived Quality (PQ) Improvement in Buses». In: *Symposium on International Automotive Technology (2026)*. SAE Technical Paper. 2026 (cit. on p. 7).

[4] P Kosec, S Škec, and D Miler. *A comparison of the tolerance analysis methods in the open-loop assembly. Advances in Production Engineering & Management, 16 (1), 44–56*. 2020 (cit. on p. 9).

[5] Wayne Cai. «A new tolerance modeling and analysis methodology through a two-step linearization with applications in automotive body assembly». In: *Journal of Manufacturing Systems* 27.1 (2008), pp. 26–35 (cit. on p. 10).

[6] Dimensional Control Systems. *3DCS Software Overview*. Dimensional Control Systems. URL: https://community.3dcs.com/help_manual/3dcssoftwareoverview.htm (visited on 07/09/2026) (cit. on p. 12).

[7] Qiaofeng Meng, Linxuan Zhang, Linzheng Yin, Zhixia Hou, and Zou Fang. «Three dimension tolerance analysis and optimization based on 3DCS». In: *Journal of System Simulation* 30.5 (2019), pp. 1730–1738 (cit. on p. 12).

[8] Matthias De Clerk, Manfred Dangelmaier, Gernot Schmierer, and Dieter Spath. «User centered design of interaction techniques for VR-based automotive design reviews». In: *Frontiers in Robotics and AI* 6 (2019), p. 13 (cit. on pp. 13, 18).

[9] nVIZ GmbH. *nViz Dimensional Variation Analysis – Perceived Quality with VRED*. Software presentation. Autodesk Automotive Innovation Forum. 2024. URL: `https://boards.autodesk.com/automotive-innovation-forum/items/nviz-dimensional-variation-analysis-%E2%80%93-perceived-quality-with-vred` (visited on 07/09/2026) (cit. on p. 14).

[10] Michela Dalle Mura and Gino Dini. «An augmented reality approach for supporting panel alignment in car body assembly». In: *Journal of Manufacturing Systems* 59 (2021), pp. 251–260 (cit. on pp. 16, 18).

[11] Małgorzata Poniatowska, Dariusz Mazurkiewicz, and Piotr Sobecki. «Digital twin in metrology: opportunities, current implementations and research challenges». In: *Metrology & Hallmark* 1 (2024) (cit. on p. 16).

[12] Glyn Lawson, Davide Salanitri, and Brian Waterfield. «Future directions for the development of virtual reality within an automotive manufacturer». In: *Applied ergonomics* 53 (2016), pp. 323–330 (cit. on p. 18).

[13] T Stoll and K Paetzold. «Gap and flush visualization of virtual nonideal prototypes». In: *DS 48: Proceedings DESIGN 2008, the 10th International Design Conference, Dubrovnik, Croatia*. 2008 (cit. on p. 18).

[14] GeeksforGeeks. *Introduction to Levenshtein Distance*. GeeksforGeeks. URL: `https://www.geeksforgeeks.org/dsa/introduction-to-levenshtein-distance/` (visited on 07/09/2026) (cit. on p. 37).