IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/46AI/2025/4630925/
Vol. 2025 (34), Article ID 4630925

Concept for Automatic UI Generation for Hierarchical XML Structures in PLM Systems Based on a Hybrid Data Model*

Mariusz CHOLEWA [0000-0002-7263-4454]

Wroclaw University of Science and Technology, Wroclaw, Poland

Correspondence should be addressed to: Mariusz CHOLEWA, mariusz.cholewa@pwr.edu.pl

* Presented at the 46th IBIMA International Conference, 26-27 November 2025, Ronda, Spain

Abstract

This paper presents a initial concept for automatic user interface (UI) generation for multi-level hierarchical data structures (e.g., BOMs, variants, revisions) in PLM systems. It builds on a hybrid data model combining RDBMS and XML and on an MVC → XML → XSLT → HTML rendering pipeline previously demonstrated in the PROEDIMS environment. We propose: (1) a metamodel of tree-based views (NodeView, ChildrenRegion, RelationView, AggregateView), (2) declarative XML→component mapping rules with context inheritance and support for lazy-loading/pagination, and (3) a sketch of an XML→UI dependency graph enabling local refresh of affected interface fragments. The approach preserves a single source of truth (domain templates) for both data and UI, reducing time-to-screen as the model evolves (e.g., sustainability extensions: carbon/water footprint, recycling, Scope 1–3) while limiting engine-level changes. This is a vision paper: it formulates assumptions, architecture, and usage scenarios and identifies benefits and risks (e.g., deep-XML processing costs, XPath rule complexity, variability of XML support in RDBMS), leaving quantitative validation of a prototype for future work.

Keywords: PLM, XML, XSLT, MVC, BOM, automatic UI generation, hierarchical data.

Introduction

The digital evolution of enterprises (Industry 4.0, circular economy, sustainability) requires PLM to adapt data models and interfaces rapidly to changing requirements. In practice, the data layer adapts first, followed by business logic and UI, which can be costly and time-consuming.

Prior work established and implemented in the PROEDIMS system a hybrid persistence and presentation approach: core, frequently queried business attributes were materialised in relational tables for indexing and filtering, while the complete, versioned product structures were persisted as XML. A classic web MVC stack (Java/JDBC, Tomcat, Servlets/JavaBeans, MirageFramework) assembled request-scoped XML documents that were transformed via XSLT (XALAN/XERCES) into HTML views. Crucially, UI forms, lists and property panels were generated from the same domain templates that governed the data, so adding fields or sections in templates led to immediate, automatic changes in the UI without modifying the application engine. The approach was illustrated on sustainability-related attributes (e.g., carbon and water footprint, recycling indicators, Scope 1–3), demonstrating short time-to-screen and controlled customisation in an industry-oriented PLM context (Cholewa, 2026).

Cite this Article as: Mariusz CHOLEWA, Vol. 2025 (34) "Concept for Automatic UI Generation for Hierarchical XML Structures in PLM Systems Based on a Hybrid Data Model " Communications of International Proceedings, Vol. 2025 (34), Article ID 4630925, https://doi.org/10.5171/2025.4630925

The gap: the mechanism focused on single-level views, whereas PLM operates on nested structures (BOMs, component–material–operation relations). This paper outlines a conceptual approach to automatic generation of multi-level UIs, compatible with the existing architecture.

Fig. 1 Conceptual pipeline from template XML and mapping declarations to HTML via View AST and XSLT

Beyond our prior hybrid RDBMS+XML and MVC/XSLT pipeline, this work connects with recent model-based and intelligent user interface research. Contemporary frameworks revisit adaptation and model-driven transformations for multi-target UIs (Abrahão et al., 2021) and consolidate MBUI advances in systematic reviews (Erazo-Garzón et al., 2023). In parallel, practical systems demonstrate on-the-fly GUI generation from XML-based descriptions—underscoring the feasibility of declarative, data-driven UI construction (Bavay et al., 2022).

Within the PLM domain, recent studies emphasise PLM as a cross-lifecycle digital backbone and examine sectoral and sustainability-driven evolutions. System-level SLM work underlines the need to bridge Application and Product Lifecycle Management (Wyrwich et al., 2024), while systematic reviews on sustainable PLM (Seegrün et al., 2024) and implementations in industry (Gunjal & Belokar, 2023) highlight BOM-centric data as a recurring challenge for usability and performance.

For change propagation in hierarchical structures, recent progress on tree edit distance (TED) and related problems offers more scalable foundations for delta computation and impact analysis. For example, recent algorithmic advances accelerate weighted and unweighted TED (Nogler et al., 2025). Although our paper remains conceptual, such results motivate the proposed dependency-graph and local-refresh strategy for XML-driven UIs.

PLM Architecture and Role

PLM is the digital backbone in a layered architecture (presentation, business logic, data), typically with a relational metadata repository and a file vault. Adapting PLM to new needs is expensive; excessive customization hinders maintainability and interoperability.

Recent reviews reinforce PLM's role as a cross-lifecycle digital backbone and highlight the need for tight integration between product and system lifecycle viewpoints (Wyrwich et al., 2024). In parallel, sustainability-oriented studies emphasise that PLM user interfaces must expose multi-level product structures and impact indicators to support decision-making across engineering and operations (Seegrün et al., 2024).

Industry-focused syntheses report uneven PLM adoption and recurring customisation, underlining the value of declarative, data-driven UI layers that can evolve with product data models (Gunjal & Belokar, 2023).

Hybrid Data Model (Recap)

Key XML attributes are mapped to relational columns (for fast filtering/indexing), while the full object structure and versioning are preserved as an XML tree. This speeds up searching while keeping deep structure in XML.
From a UI-engineering perspective, recent model-based and intelligent UI approaches demonstrate that declarative transformations can remain practical and maintainable when backed by explicit models (Abrahão et al., 2021). Outside PLM, automatic GUIs generated from semantic/XML descriptions have proven viable in production-grade tools (Bavay et al., 2022), which supports the feasibility of template-driven hierarchic views in our context.

Challenges for XML and UI

XML is expressive but memory- and CPU-intensive for large trees. XML support in RDBMSs comes with limitations and may require careful indexing or alternative stores. This motivates local UI updates instead of full re-rendering.

Multi-view and multi-level BOM management continues to evolve, with recent proposals addressing mapping and consistency across views for complex products (Chen et al., 2023). For change localisation in hierarchical structures, ongoing advances in tree edit distance improve the computational foundation for delta computation and impact analysis (Nogler et al., 2025). These strands jointly motivate a dependency-aware UI generation approach capable of local refresh.

Prior PROEDIMS Implementation (from earlier research)

Data layer. The implementation followed a hybrid persistence model: core business attributes mapped to relational columns for filtering/indexing, while full object structures and version histories were persisted as XML trees (e.g., xml_system_object_template and xml_system_object_version). This enabled schema evolution without intrusive database refactoring, and supported domain templates as the contract between data and UI. Presentation pipeline. A classic MVC web stack (Java/JDBC, Tomcat, Servlets/JavaBeans, MirageFramework) constructed request-scoped XML documents which were transformed by XSLT (XALAN/XERCES) into HTML views. By reading the same object templates, the view layer could automatically surface newly introduced fields and sections without changes to the application engine. Automatic UI adaptation. UI forms, lists and property panels were generated from template annotations, including basic validation and formatting rules. Selected attributes were duplicated (materialized) into relational columns for efficient search, while deep, multi-level structures remained in XML. This separation kept interactive filtering responsive yet preserved the expressiveness of hierarchical data. Sustainability extensions. The approach was validated conceptually on environmental attributes (e.g., carbon and water footprint, recycling indicators, Scope 1–3). Adding fields to the templates resulted in immediate UI availability across relevant product/part/material views—illustrating short time-to-screen for evolving requirements. Versioning and change handling. Templates and instances were versioned at the XML level, allowing side-by-side evolution of definitions and data. While performance optimisation and local refresh policies are goals of the present concept paper, the earlier implementation already demonstrated practical feasibility of template-driven adaptation in the PROEDIMS system (Cholewa, 2026).

Conceptual Framework for Automatic Tree-Based UI Generation

Metamodel of Hierarchical Views

We propose a lightweight metamodel comprising:

- NodeView — a panel for the current node (e.g., 'part', 'material' with attributes derived from the XML template).
- ChildrenRegion — a child list region (table with lazy expand/pagination).
- RelationView — a relation section (e.g., assigned materials/operations).
- AggregateView — an aggregation section (e.g., carbon footprint sums over a branch).

The metamodel is derived from the existing object templates (the same definitions that drive the data model), ensuring data–UI consistency.

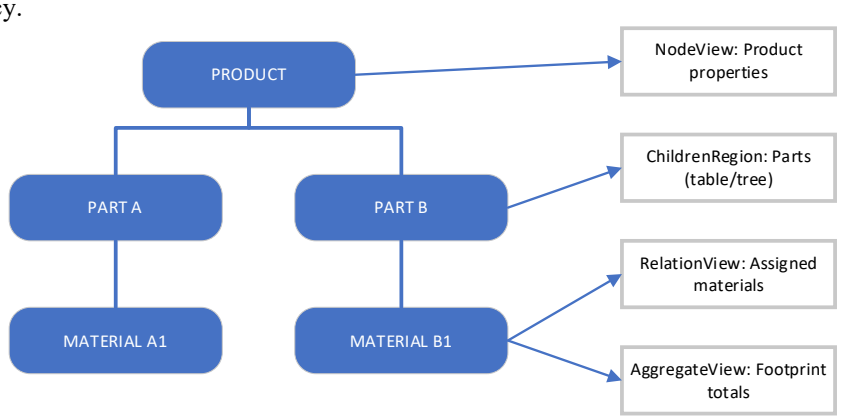

Fig. 2 Example mapping of a BOM fragment to NodeView, ChildrenRegion, and AggregateView

Note: Diagram the left-side XML tree and right-side UI composition; arrows indicate mapping rules.

Declarative XML→UI Mapping Rules

Rules (annotations in templates or XSLT attributes) define component selection (form, tree, table), visibility and validations, context inheritance (XPath/domain keys), and pagination/lazy-load of subtrees. These rules extend the existing XML→XSLT→HTML pipeline in PROEDIMS, without changing the application engine.

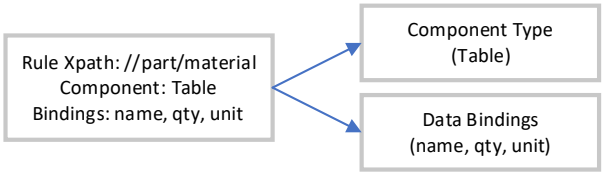

Fig. 3 . Illustrative snippet of a declarative mapping rule

Sketch of an XML→UI Dependency Graph

Conceptually, we assume a dependency graph linking XML document fragments (nodes/attributes) with UI components. Changes in the tree (insert/update/delete/move) locally invalidate only the components that depend on affected paths. Dependencies can be captured by instrumenting XSLT templates (collecting used XPath expressions at render time). We present the idea, not an implementation.

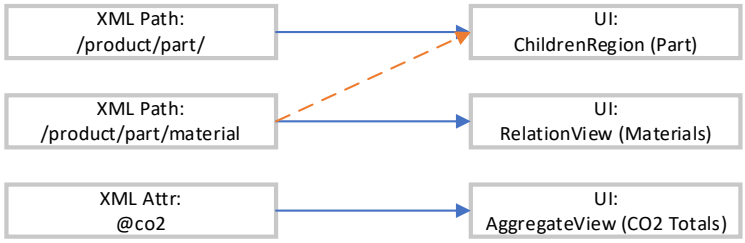

Fig. 4 XML→UI dependency graph concept

Note: Nodes represent XML paths and UI components; edges represent dependencies; highlight local invalidation path after an XML change.

High-Level Architecture

We retain the existing MVC pattern: Model: Java + JDBC/DBO (PostgreSQL/MySQL/MS SQL), XML object templates, versioning in xml_system_object_version. Controller: Tomcat, Java Beans/Servlets/MirageFramework; building XML trees per request. View: browser; XSLT (XALAN/XERCES) transforms XML plus mapping rules into HTML 4.0. The novelty is a 'Hierarchical View Definition' layer (metamodel + rules), processed alongside data.

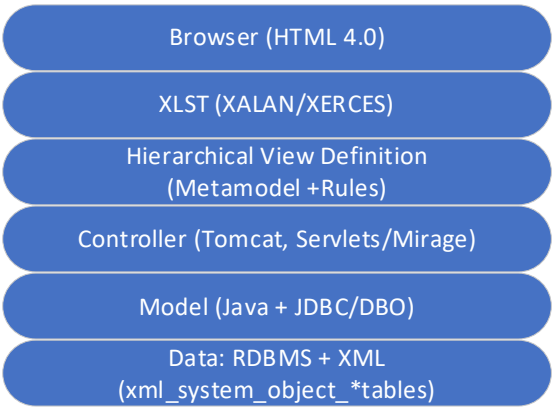

Fig. 5 Layered architecture with the new Hierarchical View Definition layer

Expected Benefits and Risks

Benefits (from previous observations and concept extension):

- Faster adaptation of the data model and automatic UI refresh based on templates;
- No engine changes when the model evolves;
- Polyglot openness (Java + multiple RDBMSs);
- Efficient filtering by mapping selected attributes to SQL while preserving rich XML structure.

Risks/limitations (conceptual):

- Performance of deep XML processing and memory costs;
- Limitations of XML support in RDBMSs; potential need for alternative stores;
- XPath complexity with many rules.

These are marked for future research; we do not claim results here.

Discussion, Limitations, and Conclusions

This work advances a vision for automatic generation of hierarchical UIs in PLM by extending a proven hybrid (RDBMS+XML) and MVC/XSLT foundation. Rather than proposing a full prototype and empirical evaluation, we consolidate architectural choices (templates as a single source of truth; declarative XML→UI mapping; dependency-aware local refresh) and articulate how they address multi-level product structures (BOMs, variants, revisions). By keeping templates central, the concept preserves alignment between evolving data definitions and the UI surface, a recurring pain point in PLM customisations.

Limitations and threats to validity:

- Concept-only status. We intentionally report no quantitative results; feasibility and performance remain to be validated on real workloads.
- XML processing costs. Deep and wide XML trees can be memory/CPU-intensive; effectiveness depends on indexing, streaming, and selective materialisation.
- XPath rule complexity. A large rule set may become hard to manage; naming conventions, modularisation and tooling will be necessary.
- RDBMS variability. XML feature support, indexing strategies and query planners differ across vendors; portability requires an abstraction layer.
- Coupling to templates. Strong reliance on template quality and governance may propagate modelling errors directly to the UI.

- Usability risks. Automatically generated hierarchical views may overwhelm users; pattern libraries and progressive disclosure are essential.
- Security and access control. Fine-grained, attribute-level visibility in generated UIs must remain consistent with RBAC/ABAC policies.

Mitigation strategies and research directions:

- Prototype and benchmarks. Implement an end-to-end proof-of-concept and measure refresh scope, rendering latency and memory overhead vs. baselines.
- Dependency graph engineering. Evaluate capture granularity (path-level, node-level, attribute-level) and batching strategies for local refresh.
- Rule management. Introduce a declarative registry, linting and tests for mapping rules; explore higher-level DSLs generating XPath/XSLT.
- Data architecture options. Compare hybrid RDBMS+XML with document or graph stores for specific workloads (e.g., deep BOM expansion).
- UX evaluations. Conduct formative studies of navigation/aggregation patterns (breadcrumbs, focus-mode, in-place editing) on representative tasks.
- Governance and security. Bind mapping rules to access-control metadata; verify policy-consistent UI generation in versioned templates.

Conclusions

We have outlined a coherent concept to elevate automatic UI generation from flat to hierarchical structures in PLM. The approach leverages template-driven modelling, declarative mapping and dependency-aware refresh to balance agility with maintainability. Anchored in prior PROEDIMS work and an already published reference (Cholewa, 2026), the concept is technologically plausible and practically motivated. The next step is a reproducible prototype and comparative evaluation against full re-rendering and naïve subtree invalidation, followed by usability studies in industry-grade scenarios.

References

- Abrahão, S., Insfran, E., Sluÿters, A., & Vanderdonckt, J. (2021). Model-based intelligent user interface adaptation: challenges and future directions. Software and Systems Modeling, 20(5), 1335–1349. https://doi.org/10.1007/S10270-021-00909-7/FIGURES/13
- Bavay, M., Reisecker, M., Egger, T., & Korhammer, D. (2022). Inishell 2.0: Semantically driven automatic GUI generation for scientific models. Geoscientific Model Development, 15(2), 365–378. https://doi.org/10.5194/GMD-15-365-2022
- Chen, J., Xiao, Y., Wang, G., & Guo, B. (2023). Research on the integrated management and mapping method of BOM multi-view for complex products. Mathematical Biosciences and Engineering, 20(7), 12682–12699. https://doi.org/10.3934/MBE.2023565
- Cholewa, M. (2026). A Concept for Automatically Adapting PLM System User Interfaces to Dynamic Business Requirements Using an Adaptive Data Model. 181–196. https://doi.org/10.1007/978-3-032-01517-4_14
- Erazo-Garzón, L., Suquisupa, S., Bermeo, A., & Cedillo, P. (2023). Model-Driven Engineering Applied to User Interfaces. A Systematic Literature Review. Communications in Computer and Information Science, 1755 CCIS, 575–591. https://doi.org/10.1007/978-3-031-24985-3_42
- Gunjal, H. V., & Belokar, R. M. (2023). Systematic Review: Implementation of Product Lifecycle Management in Industries. Lecture Notes in Mechanical Engineering, 263–279. https://doi.org/10.1007/978-981-19-6107-6_19
- Nogler, J., Polak, A., Saha, B., Vassilevska Williams, V., Xu, Y., & Ye, C. (2025). Faster Weighted and Unweighted Tree Edit Distance and APSP Equivalence. Proceedings of the Annual ACM Symposium on Theory of Computing, 2167–2178. https://doi.org/10.1145/3717823.3718116

- Seegrün, A., Hardinghaus, L., Riedelsheimer, T., & Lindow, K. (2024). Incorporating sustainability into product lifecycle management: a systematic literature review. Proceedings of the Design Society, 4, 1437–1446. https://doi.org/10.1017/PDS.2024.146
- Wyrwich, F., Kharatyan, A., & Dumitrescu, R. (2024). Interdisciplinary system lifecycle management – a systematic literature review. Proceedings of the Design Society, 4, 2765–2774. https://doi.org/10.1017/PDS.2024.280