	P1	P2	P3	P4	P5	Author's analysis
Does the architecture allow the developer to add or modify generators?	Try	Try	4 – It supports generation modules well, but requires manual effort to integrate new filters or contexts.	Yes. Among the architectural requirements is explicitly stated "Add or modify generator modules".	Yes. The architecture was designed to be modular, allowing for the addition, replacement, or modification of generators.	
Does architecture allow the designer to have complete control over the content generation processes?	I don't know	Try	4 – It offers broad parameter control, but does not allow you to change core algorithms without modifying the code.	No, the designer can change generation parameters and insert manual content via the LevelEditor, but final control over orchestration, validation, and parallel execution remains the responsibility of PCGManager and its internal components.	Yes. Through LevelEditor, the designer interacts directly with PCG Manager, being able to change parameters and insert content manually.	Key criticisms point to the designer's lack of control over the orchestrator and validator.
Does architecture allow the designer to understand how their decisions change the generation process?	Try	Try	3 – It provides basic feedback (validator score), but lacks visual dashboards for the designer to track impacts in real time.	Yes. One of the requirements is "Show generation information," providing feedback to the designer about the impact of their changes to the parameters and the generated result.	Yes. There is a feedback mechanism in which the system displays generation information, allowing the designer to understand the impacts of their decisions.	Lack of feedback, in addition to the validator's score.
Does the architecture allow the designer to manually change and add content?	Try	Try	4 – Allows manual insertion of content, but this operation may require SysML diagrams and is not very intuitive.	Yes. There is a specific instruction (AddManualPieceInstruction) that allows the designer to manually insert or modify parts of the generated content (for example, specific buildings on the map).	Yes. The designer can add manual content through specific instructions.	
Does the architecture allow for adapting content for different players?	Try	Try	4 – Adapts content via PlayerInfo/GameInfo, but depends on prior data collection and external questionnaires.	Yes. The context module includes PlayerInfo, which collects player data (profile, survey responses) to guide adaptive content generation.	Yes. Player (PlayerInfo) and game (GameInfo) data are collected to personalize content according to the player's profile and progress.	

Does the architecture allow for the effective orchestration of content?	Try	Try	3 – Basic orchestration combines artifacts, but lacks advanced blending or conflict resolution strategies.	Yes. There is an Orchestrator component dedicated to combining individual contents into a single artifact before validation.	Yes. The Orchestrator component is responsible for combining multiple pieces of content into a single artifact in a modular way.	There is a lack of merging and conflict resolution techniques.
Does the architecture allow for the effective validation of the generated content?	Try	Try	3 – Validation ensures minimum feasibility, but does not cover variability or usability metrics without customization.	Yes. The Validator component assesses the feasibility of the generated artifact and may request a new generation if the minimum threshold is not met.	Yes. The Validator component can apply feasibility algorithms and criteria before the content is delivered to the player.	
Does the architecture allow for the easy use of various algorithms to validate the content?	Try	Try	3 – The generic interface accepts various validators, but each new algorithm requires the implementation of adapters.	Yes. The Validator was designed to support different algorithms or intelligent agents, configurable according to the desired feasibility threshold.	Yes. The Validator module is generic and can be implemented using different techniques, which tends to facilitate the exchange and testing of algorithms.	
Does the architecture allow for satisfactory interaction with the game manager (GameManager)?	Try	Try	4 – GameManager interacts via ContentManagerInterface, but the data flow is restricted to synchronous calls.	Yes. The ContentManagerInterface defines generic methods that allow the GameManager to request generation, provide context, and receive generated content.	Yes. The GameManager sends contextual data and content requests to the PCGManager, receiving the generated content in response.	
The architecture allows interaction with the level editor (LevelEditor) in a satisfactory manner?	Try	Try	4 – LevelEditorIt provides direct control, but the final usability depends on the interface implementation.	Yes. TheLevelEditorIt offers direct interaction to the designer, allowing for richer manual parameter and generation instruction changes than GameManager.	Yes. LevelEditor is a tool geared towards the designer, allowing for more direct control over the generation process.	
Does the architecture allow for the content creation process to be performed in a high-performance and scalable way?	Try	Try	4 – Parallel generation and caching improve performance, but orchestration and sequential validation can become	Yes. The use of parallel workers, content caching, and in-game generation goals ensure performance and scalability.	Yes. The architecture provides for real-time generation, support for parallel generation via Workers, and the use of caching to improve	Bottleneck in orchestration and validation.

			bottlenecks.		performance.	
Does the architecture allow for the easy use of various procedural generation algorithms?	Try	Try	4 – It supports multiple algorithms, but the framework does not offer ready-made adapters for all popular standards.	Yes. The extensibility requirement allows for the modular coupling of new generators, without needing to alter the core architecture.	Yes. The modularity and extensibility of the architecture were designed for that purpose.	
Is the architecture easily ported to different game engines?	I don't know	Try	4 – The generic interface makes porting to other engines easier, but it is necessary to refactor SysML reports and module dependencies.	Yes. Portability is an explicitly defined quality metric, and the ContentManagerInterface provides an abstraction layer to facilitate integration across various platforms.	I believe so. The architecture proposes a generic interface intended to be used in different projects.	
Is architecture easily ported to different types of games?	I don't know	Try	3 – Conceptually applicable to various games, but lacks practical examples beyond maps and missions.	Yes. Although focused on maps and quests, the architecture defines Content in a generic way, allowing for expansion to include music, enemies, etc., catering to different game genres.	Yes. The architecture proposes a generic interface so that it can be used in different games.	
Can the architecture be easily installed?	Try	Try	4 – Instantiable via dynamic Workers, but requires manual thread and cache management.	Yes. Each generator, worker, and cache can be installed dynamically, and PCGManager manages these instances in a standardized way.	Yes. Components such as Workers, Generators, and PCGManager are described as reusable and instantiable in a parallel or modular fashion.	Thread management can complicate implementation.
Are the requirements translated into architectural elements?	I don't know	Try	3 – Most requirements are mapped, but metrics such as variability and comprehensiveness do not have a dedicated architectural element.	Yes. There is a direct mapping between requirements (e.g., adaptability, controllability, portability) and components (Context, Validator, Generic Interface) in the architectural requirements section.	Yes. The document clearly maps the functional and quality requirements for elements of the architecture.	Variability and Comprehensiveness do not have a dedicated architectural element.
Was any attribute of procedural generation left out?	I don't know	No	3 – Some attributes, such as UI comprehensibility and controllability, were	No. All extracted quality metrics (modularity, variability, feasibility, etc.) were addressed by	It's difficult to say. However, architecture encompasses a wide variety of attributes.	UI comprehensibility and controllability lack dedicated architectural elements.

			underdeveloped within the scope of the architecture.	the listed architectural components or requirements.		
Have the quality requirements and attributes been satisfactorily integrated into the architecture?	Try	Try	4 – Requirements and quality are integrated coherently, but there is a lack of defined standards for each attribute.	Yes. The PCG-RA architecture demonstrates coherence between functional requirements, quality metrics, and architectural views (conceptual, functional, structural, and procedural), providing a comprehensive and modular solution.	Yes. The integration between the requirements and the architecture modules is clear and well detailed.	