INVENTION SPECIFICATION

INVENTION TITLE

[0001] Data driven computing system development

[0002] The present application claims priority to the earlier filed provisional application having the U.S. Application Number: 63/318,765, Filing Date: Mar 10, 2022, Name of Inventor: Jin Ming Wen, Title of Invention: Data driven computing system development, and hereby incorporates subject matter of the provisional application in its entirety.

COPYRIGHT NOTICE

BACKGROUND OF THE INVENTION

Field of the Invention

[0004] The invention discloses a definite approach to construct computing system software based on the data: Input data, Middle data and Output data, and the calculations or relationships between them and provides the safety mechanism development approach to make sure that the software will work correctly by focusing on only two unique systematic attributes: Data Value and Data Timing. In this way, the software constructions, the relevant software safety mechanisms developments and the measurement criteria for both software constructions and safety developments are defined once the data and the calculations between them are defined

in the system. Every computing system software has two, and only two attributes: Data Value and Data timing, which fully represent the system functionalities from the system external behavior point of view. And the system development goal is to realize those attributes for each required output data, in which, the data values are derived by the calculations for the output data, and the data timing is derived by the system latency including both data transmission durations and data calculations durations, and the safety mechanisms' goal is to detect any deviations of those two attributes in said data and prevent the system or software from impacting other systems by outputting the deviated data, so that the system development activities including the software constructions and the safety mechanisms' implementations can be optimized significantly.

Description of the Related Art

[0005] Computing system development

[0006] Currently there is no other fixed way to construct computer software. Even for same products in same organizations, different developers have different ways to develop the required software, which is prone to mistakes and inefficient. For example, in the Software Engineering Handbook from NASA, there are 18 different suggested ways to construct software, one of which is the "4+1" views which applies total of 5 view modules, and each of the view modules has multiple ways to construct the required software based on developers' experiences. The inconsistency of such developments significantly impacts the software products' quality and developments' efficiency. The commonly used computing system development is based on the developers' experience and the development specifications are specified either using the text tools, such as IBM DOORS or PTC Integrity, or the notation tools, such as the SysML that includes 9 types of diagrams, the issues of which are that there is neither the clearly defined

explicit and complete approach to design the system and specifications, nor is there the clearly defined explicit and complete method to fully cover all the system functionalities, which will cause issues in the software development.

[0007] For the text specified specifications, the issues will include that the text specifications are prone to ambiguous and incomplete, and it is difficult to figure out the logic relationships in the specifications, whose consequence is that the specifications may be inconsistent, incomplete and inaccurate, then it will cause the issues in the following development steps. For example, the system requirements that describe the users' needs for the products under development are commonly documented using IBM DOORS in text format combining with some diagrams, which will be easily interpreted into different meanings by different person, and it will be very difficult to fully and accurately describe the users' needs.

[0008] For the notation specified specifications, the issues include that it is difficult to fully specify the system functionalities, and it is difficult to use the notations in the entire development team, and it is difficult to figure out the relationships in all the diagrams used in the development.

[0009] Safety development in a computing system

[00010] There is the same issue in software safety development in a computing system as the one above: currently there is not fixed way to development the safety mechanisms for a specific computer software, even for same software products in same organizations, different developers have different ways to develop the safety mechanisms, there are not unique criteria to measure software safety. Safety development in a computing system is guided by the IEC 61508, and the safety development in an automotive electronic control unit (ECU) system is guided by the ISO 26262. In both standards, although there are some quantitative hardware criteria about

the safety, such as the Single Point Fault Metric (SPFM), Latent Fault Metric (LFM) and the Probabilistic Metrics for Hardware Failures (PMHF) listed in the Table 3.4 14 Hardware Fault Metrics in the Part 5 of ISO 26262, however, the criteria to measure if a software component or the whole system is safe are not specified, and the safety development measures mentioned in both of them are very vague because only very high-level activities are specified, for example, in the ISO 26262 Part 4: Product development at the system level and the ISO 2626 Pat 6: Product development at the software level, but those activities are highly dependable to interpretation and implementation, which is very difficult to make accurate adjudgment.

[00011] The commonly used safety development approaches in computing systems such as an automotive electronic control unit (ECU) system are based on the developers' experience, and based on the requirements and design specifications that are specified either using the text tools, such as IBM DOORS or MKS Integrity, or the notation tools, such as the SysML that includes 9 types of diagrams, the issues of which are that there is neither the clearly defined explicit and complete approach to design the safety mechanisms, nor is there the clearly defined explicit and complete method to fully cover all the safety aspects in the system.

[00012] For the text specified specifications, the issues will include that the text specifications are prone to ambiguous and incomplete, and it is difficult to figure out the logic relationships in the specifications, whose consequence is that the safety mechanisms may be inconsistent, incomplete and inaccurate, and the safety development is inefficient.

[00013] For the notation specified specifications, the issues include that it is difficult to fully specify the system safety, and it is difficult to use the notations in the entire development team, and it is inefficient to develop the safety mechanisms based on all the diagrams used in the development.