We have defined the following predicate rules for the elements and decorator used in an assurance case to ease understanding of an assurance case. The predicate rules for the elements and decorator of an assurance case begins with the delimiter *“@Predicate\_AC”* and ends with the delimiter *"@End\_Predicate\_AC”*

*@Predicate\_AC*

1. Goal(G): True if G is a goal within the assurance case. This predicate is represented as Goal (ID, Description) where ID is the unique identifier for the goal, and description is the textual information of the goal.
2. Strategy(S): True if S is a strategy within the assurance case. This predicate is represented as Strategy (ID, Description) where ID is the unique identifier for the strategy and description is the textual information of the Strategy.
3. Solution (Sn): True if Sn is evidence within the assurance case. This predicate is represented as Solution (ID, Description) where ID is the unique identifier for the evidence or solution and description is the textual information of the evidence.
4. Context(C): True if C is a context within the assurance case. This predicate is represented as Context (ID, Description) where ID is the unique identifier for the context and description is the textual information of the context.
5. Assumption (A): True if A is an assumption within the assurance case. This predicate is represented as Assumption (ID, Description) where ID is the unique identifier for the assumption and description is the textual information of the assumption.
6. Justification (J): True if J is a justification within the assurance case. This predicate is represented as Justification (ID, Description) where ID is the unique identifier for the justification and description is the textual information of the justification.
7. Undeveloped(X): True if X is either a Goal(G) or Strategy(S) marked as undeveloped. This predicate is represented as Undeveloped(X), where X can be either a goal or strategy.

*@End\_Predicate\_AC*

We have defined the following predicate rules for the additional decorators used to support assurance case patterns to ease understanding. The predicate rules for the additional decorators to support assurance case pattern begins with the delimiter *“@Predicate\_ACP”* and ends with the delimiter *"@End\_Predicate\_ACP”*

*@Predicate\_ACP*

1. Uninstantiated (X): True if element X (can be any GSN element) is marked as uninstantiated.
2. UndevelopStantiated (X): True if element X is either a Goal(G) or Strategy(S) and is marked both as uninstantiated and undeveloped.
3. HasPlaceholder (X): True if element ‘X’ (can be any GSN element) contains a placeholder ‘{}’ within its description that needs instantiation.
4. HasChoice (X, [Y], Label): True if an element ‘X’ (either a Goal(G) or Strategy(S)) can be supported by selecting among any number of elements in [Y] (where Y can be any GSN element) according to the cardinality specified by an optional Label. The label specifies the cardinality of the relationship between ‘X’ and ‘Y’. A label is of the general form “m of n” (e.g. a label given as “1 of 3” implies an element ‘X’ can be supported by any one of three possible supporting elements in [Y])
5. HasMultiplicity (X, [Y], Label): True if multiple instances of an element X (either a Goal(G) or Strategy(S)) relate to multiple instances of another element [Y] (where Y can be any GSN element) according to the cardinality specified by an optional Label. The label specifies the cardinality of the relationship between X and Y. (i.e., how many instances of an element in X relates with how many instances of an element in [Y]. e.g. m of n implies m instances of an element in X must be supported by n instances of an element in Y)
6. IsOptional (X, [Y], Label): True if an element X (either a Goal(G) or Strategy(S)) can be optionally supported by another element [Y] (where Y can be any GSN element) according to the cardinality specified by an optional Label. The label specifies the cardinality of the relationship between X and Y. (i.e. an instance of an element in X may be supported by another instance of an element in [Y], but it is not required)

*@End\_Predicate\_ACP*

To represent an assurance case or assurance case pattern in GSN is equivalent to depicting in a hierarchical tree structure. To achieve this hierarchical tree structure, the below predicates have been defined to ease understanding of this structure. The predicate rules to support the structure of an assurance case or assurance case pattern begins with the delimiter *“@Predicate\_Structure”* and ends with the delimiter *“@End\_Predicate\_Structure”*

*@Predicate\_Structure*

1. IncontextOf (X, [N], D): True if element X at depth D has a neighbour [N] to the left or right at depth D, where ‘[N]’ can be an Assumption (A), Justification (J), or Context (C), ‘X’ can be a Goal (G), or Strategy (S) and ‘D’ represents the height or depth of the goal or strategy element and its neighbours in the GSN hierarchical structure.
2. SupportedBy (X, [C], D): True if element X at depth D has children [C] directly below it, where [C] can include Goal (G), Strategy (S), or Solution (Sn) and ‘X’ can be a Goal (G), or Strategy (S).

* If X is Strategy (S), [C] can only be Goal (G).
* If X is Goal (G), [C] can be either Goal (G), Strategy(S), or Solution (Sn).

*@End\_Predicate\_Structure*

Now, I would provide you with domain information about ACAS Xu (Airborne Collision Avoidance System Xu) for which you would create a security case from a given security case pattern. The domain information begins with the delimiter *“@Domain\_Information”* and ends with the delimiter *"@End\_Domain\_Information”*

*@Domain\_Information*

ACAS Xu (Airborne Collision Avoidance System Xu) is a collision avoidance system designed for use in unmanned aerial vehicles (UAVs), commonly known as drones. The primary objective of ACAS Xu is to enhance the safety of drone operations by preventing collisions between drones or between a drone and other objects in its environment.

The scenario involves two drones. One called the “intruder” which is any other drone or object that poses a collision threat to the ownship. and the other called the “ownship.” which is the perspective we adopt. The ownship is equipped with ACAS Xu and has a functional space in which it must operate. This space is conceptually partitioned into two operational areas: collision avoidance threshold and collision volume with an elevated risk of collision for the ownship with intruders. When no risk of collision is detected, the ownship follows the current heading to the destination area. Otherwise, if another drone is detected in the collision volume, the ownship will turn right or left to avoid the collision and prevent the intruder from reaching the collision avoidance threshold.

The architecture of ACAS Xu contains the following components.

* Sensors: The ownship's sensors gather data on potential intruders, including their velocity, angle, and distance relative to the ownship.
* Processor: The collected data is processed to compute a suitable avoidance strategy (e.g., turn left, turn right, or do nothing).
* Planner: Based on the processor's decision, a trajectory is planned to navigate the ownship safely while avoiding collisions.
* Actuator: The planned trajectory is executed by the actuator, ensuring the ownship follows the new path.

ACAS Xu's security can be compromised if an attacker alters the messages sent to the processor, leading to incorrect decisions that may result in collisions. Therefore, ensuring the security of ACAS Xu involves:

security requirements decomposition that aims to identify security threats, and formalization of the system and the security threats to later verify the absence of threats when developing a secure system. If it can be shown that all the relevant threats have been identified and mitigated, then the system is acceptably secure.

The following security requirements (SRs) below are imposed to design a secure ACAS Xu.

* SR1: The GPS messages are genuine and have not been intentionally altered.
* SR2: The processor must receive data only from valid sensors.
* SR3: The system should employ mechanisms to mitigate unauthorized disclosure of the planning information.
* SR4: ACAS Xu development shall be done considering security risk assessment procedures.

The four SR are decomposed into requirements about the satisfaction of asset protection (SR1 –SR3)

and secure development process requirements (SR4). The former concerns requirements to protect resources that are worth protecting. The latter concerns the requirements about the development activities that must conform to a relevant secure development methodology and/or security standard.

In addition, ACAS Xu has low level elements that capture functional architecture in terms of components and connectors, and the behavioural aspects of the architectural elements. These elements include the following.

* Component: a modeling artifact which represents a piece of software architecture.
* MsgPassing: the representation of a message exchanged between two components (sender, receiver).
* Port: the interaction point through which a Component can communicate with its environment.
* ConnectorMPS: a link that enables communication between Ports.
* Payload: the useful data contained in a Message.

Based on the Microsoft STRIDE threat analysis technique, the following security threats (STs) against the components and the communication links are identified from the security requirements (SRs).

* ST1: Tampering – This threat is identified from SR1 and involves GPS sensors and processor.
* ST2: Spoofing - SR2 Sensors and processor
* ST3: Elevation of privileges - SR3 Planning system

Finally, to ensure that ACAS Xu is acceptably secure, during the creation of its security case, an instance of the goal (G0.X) is created for each security threat against which the system must be protected, where 𝑋 denotes the order of the threat.

*@End\_Domain\_Information*