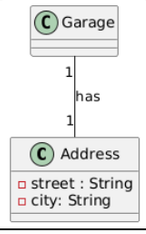
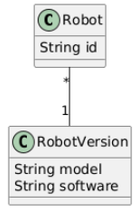
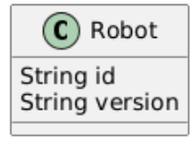
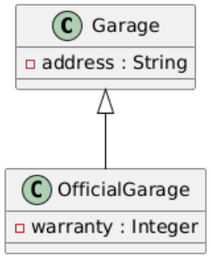
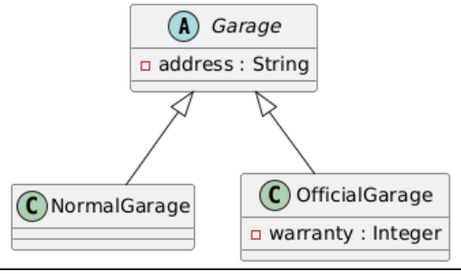
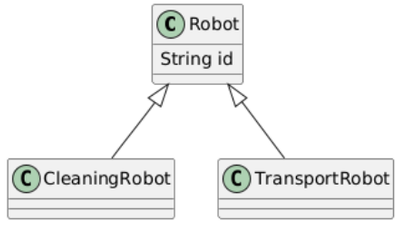
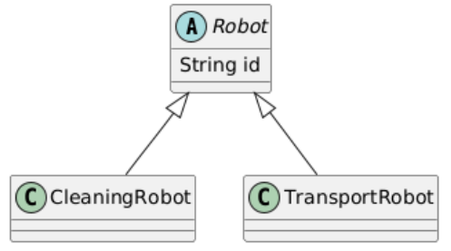
Supplementary Information: Towards Human-in-the-Loop LLM-Enabled Domain Modeling

Modeling Patterns

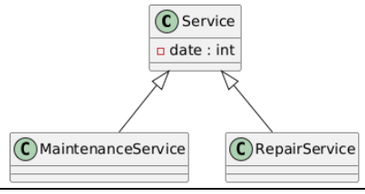
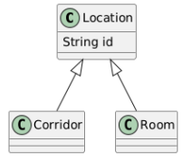
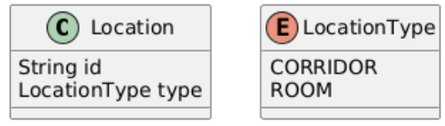
Class vs Attribute		
Decision	Use a class or attribute to represent a concept.	
Patterns	1. For all attributes: Alternative_1: attribute, Alternative_2: class 2. Class with only one Association: Alternative_1: class, Alternative_2: attribute	
Template	*We know that each {class_name} has an {attribute_name}. When we think about {attribute_name} is it just a simple piece of information, like a line of text? Or is {attribute_name} made up of different parts that might be important?. For example:* *1. For each {class_name} {attribute_name}, the {sub_attributes} information is saved separately, so users can filter {class_name} by {sub_attributes}.* *2. The {attribute_name} is a single piece of text. This works well when we just need to show the {attribute_name}.* *Which option is better for you:* ***<Answers>***	
Answers	*Choice 1: we need the {attribute_name} details separately.*	*Choice 2: a single line of text for {attribute_name} is enough.*
Actions	*1. Add class {attribute_name}.* *2. Set confidence for class high (Threshold x 1.1).* *3. Add Association from class {class_name} to class {attribute_name}* *4. Update the cardinality to "1" in the { class_name } side and "1" in the { attribute_name} side* *5. Set confidence for association low (Threshold / 2)* *6. Set confidence for cardinalities low (Threshold / 2)* *7. Remove {attribute_name} from garage.*	*1. Add attribute {attribute_name} to {class_name}.* *2. Set confidence for attribute high (Threshold x 1.1)* *3. Set attribute type to* {sub_attributes} *type* *4. Remove the association* *5. Remove class {attribute_name}*

Domain: Car Service		**Domain: Robot tasks**	
			
Variables:		**Variables:**	
{class_name}: Garage {attribute_name}: Address {sub_attributes}: Street, City		{class_name}: Robot {attribute_name}: Version {sub_attributes}: Model, Software	
Question		**Question**	
We know that each garage has an address. When we think about address, is it just a simple piece of information, like a line of text? Or is address made up of different parts that might be important?. For example: 1. For each garage address, the street and city information is saved separately, so users can filter garages by street or city. 2. The garage address is a single piece of text. This works well when we just need to show the address.		We know that each robot has a version. When we think about version, is it just a simple piece of information, like a line of text? Or is version made up of different parts that might be important?. For example: 1. For each robot version, the model and software information is saved separately, so users can filter robots by model or software. 2. The robot version is a single piece of text. This works well when we just need to show the version.	
Answer Options		**Answer Options**	
Choice 1: we need the address details separately.	Choice 2: a single line of text for address is enough.	Choice 1: we need the version details separately.	Choice 2: a single line of text for version is enough.

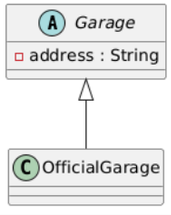
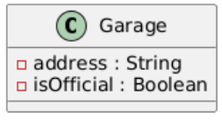
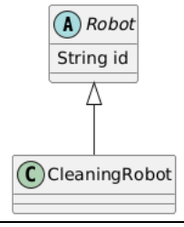
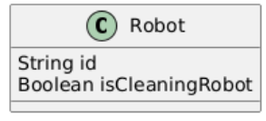
Concrete Class vs Abstract Class		
Decision	Represent concept as abstract or does it require a concrete implementation.	
Patterns	1. For all the concrete superclasses: Alternative_1: concrete, Alternative_2: abstract 2. For all the abstract superclasses: Alternative_1: abstract, Alternative_2: concrete	
Template	*When we talk about {class_name}, do we always have a clear idea of a specific example, or do we sometimes speak about it in a more general way? Should we think of {class_name} as a general concept that covers all possible forms, or is it always tied to a particular type or instance?.* *For example:* *1. A general {class_name} can exists that is not a {sub_classes}.* *2. A {class_name} is just a concept and cannot exist on its own. We only need specific types like {sub_classes}.* *Which option is better for you:* ***<Answers>***	
Answers	*Choice 1: we should think of general {class_name} independent of its types.*	*Choice 2: the general {class_name} is not needed, and we only need its types.*
Actions	*1. Update class {class_name} to concrete* *2. Set confidence for class being concrete high (Threshold x 1.1)*	*1. Update class {class_name} to abstract* *2. Set confidence for class being abstract high (Threshold x 1.1)*

Domain: Car Service		**Domain: Robot tasks**	
			
Variables:		**Variables:**	
{class_name}: Garage {sub_classes}: OfficialGarage		{class_name}: Robot {sub_classes}: CleaningRobot, TransportRobot	
Question		**Question**	
When we talk about garages, do we always have a clear idea of a specific example, or do we sometimes speak about it in a more general way? Should we think of garage as a general concept that covers all possible forms, or is it always tied to a particular type or instance?. For example: 1. A general garage can exists that is not an official garage. 2. A garage is just a concept and cannot exist on its own. We only need specific types like official garages.		When we talk about robots, do we always have a clear idea of a specific example, or do we sometimes speak about it in a more general way? Should we think of robot as a general concept that covers all possible forms, or is it always tied to a particular type or instance?. For example: 1. A general robot can exists that is not a cleaning robot, or transport robot. 2. A robot is just a concept and cannot exist on its own. We only need specific types like cleaning robot and transport robot.	
Answer Options		**Answer Options**	
Choice 1: we should think of general garage independent of its types.	Choice 2: the general garage is not needed, and we only need its types.	Choice 1: we should think of general robot independent of its types.	Choice 2: the general robot is not needed, and we only need its types.

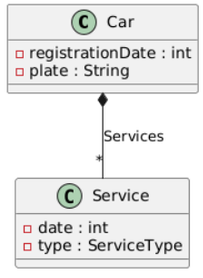
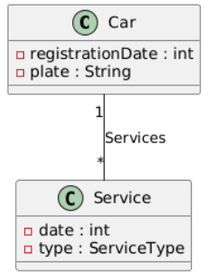
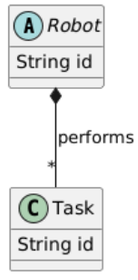
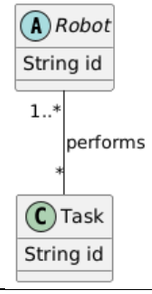
Enumeration vs Inheritance		
Decision	Represent a concepts as literals in enumeration or sub classes.	
Patterns	1. Subclasses has no attributes and no further associations besides inheritance: Alternative_1: Inheritance, Alternative_2: Enumeration 2. All enumerations: Alternative_1: Enumeration, Alternative_2: Inheritance	
Template	*There are different {attribute} of {class}: {enumeration_literals}. Do these {attribute} have distinct characteristics or requirements? . For example:* *1. {enumeration_literals} are considered different {attribute}, allowing for distinct details for each {attribute}.* *2. Every {class }*{attribute} *is classified as either {enumeration_literals}. Both {attribute} have the same details, and there's no need for extra information.* *Which option is better for you:* ***<Answers>***	
Answers	*Choice 1: {enumeration_literals} do have distinct characteristics.*	*Choice 2: {enumeration_literals} doesn't have distinct characteristics.*
Actions	*1. Create subclasses {{enumeration_literals} + {class}}* *2. Set confidence for subclasses high (Threshold x 1.1)* *3. Create inheritance from created classes to {class}* *4. Set confidence for inheritance relationships high (Threshold x 1.1)* *5. Remove enumeration {enumeration}* *6. Remove attribute {attribute}*	*1. Create enumeration {attribute}* *2. Add literals {enumeration_literals}.* *3. Set confidence for enumeration and literals high (Threshold x 1.1)* *4. Add attribute {attribute} in class {class}* *5. Set confidence for attribute high (Threshold x 1.1)* *6. Update class {class_name} to concrete* *7. Remove inheritance from {subclasses} to {class}* *8. Remove subclasses {subclasses}*

Domain: Car Service		**Domain: Robot tasks**	
			
Variables:		**Variables:**	
{class}: Service {attribute}: Type {enumeration_literals}: Maintenance, Repair {enumeration}: ServiceType {subclasses}: MaintenanceService, RepairService		{class}: Location {attribute }: Type {enumeration_literals }: Corridor, Room {enumeration}: LocationType {subclasses}: Corridor, Room	
Question		**Question**	
There are different types of services: maintenance and repair. Do these types have distinct characteristics or requirements?. For example: 1. Maintenance and repair are considered different service types, allowing for distinct details for each type. 2. Every service type is classified as either maintenance or repair. Both types have the same details, and there's no need for extra information.		There are different types for locations: corridors and rooms. Do these locations have distinct characteristics or requirements? . For example: 1. Corridors and rooms are considered different locations, allowing for distinct details for each location. 2. Every location type is classified as either corridor or room. Both locations have the same details, and there's no need for extra information.	
Answer Options		**Answer Options**	
Choice 1: Maintenance and repair do have distinct characteristics.	Choice 2: Maintenance and repair doesn't have distinct characteristics.	Choice 1: Corridors and rooms do have distinct characteristics.	Choice 2: Corridors and rooms doesn't have distinct characteristics.

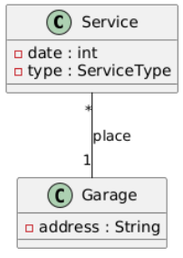
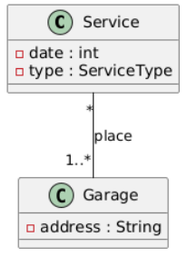
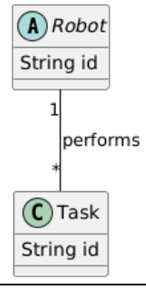
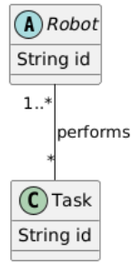
Attribute vs Inheritance		
Decision	A concept is an attribute or a sub class.	
Patterns	1. Parent class has only one subclass with no attributes and no further associations besides inheritance: Alternative_1: Inheritance, Alternative_2: Attribute 2. All boolean attributes: Alternative_1: Attribute, Alternative_2: Inheritance	
Template	*When thinking about {class_name}, what makes a {subclass_name} different from a regular one? Is there a specific role or responsibility that comes with being a {subclass_name}? Are there additional factors we need to consider when dealing with {subclass_name} compared to regular {class_name}?. For example:* *1. A {subclass_name} is treated as a different kind of {class_name}.* *2. {class_name} include a status that identifies if it as {subclass_name} or not.* *Which option is better for you:* ***<Answers>***	
Answers	*Choice 1: {subclass_name} are a special kind of garage.*	*Choice 2: {subclass_name} are similar to other robots.*
Actions	*1. Create subclass {subclass_name}* *2. Set confidence for subclass high (Threshold x 1.1)* *3. Create inheritance from created class to {class_name}* *4. Set confidence for inheritance relationships high (Threshold x 1.1)* *5. Remove attribute {attribute}*	*1. Create attribute {"is"+{subclass_name})* *2. Set confidence for attribute high (Threshold x 1.1)* *3. Update class {class_name} to concrete* *4. Remove inheritance from {subclass_name} to {class_name}* *5. Remove subclass {subclass_name}*

Domain: Car Service		**Domain: Robot tasks**	
			
Variables:		**Variables:**	
{class_name }: Garage {subclass_name}: OfficialGarage {attribute}: isOfficial		{class_name }: Robot {subclass_name}: CleaningRobot {attribute}: isCleaningRobot	
Question		**Question**	
When thinking about garages, what makes an official garage different from a regular one? Is there a specific role or responsibility that comes with being an official garage? Are there any additional factors we need to consider when dealing with official garages compared to regular garages?. For example: 1. An official garage is treated as a different kind of garage. 2. Garages include a status that identify if it as official garage or not.		When thinking about robots, what makes a cleaning robot different from a regular robot? Is there a specific role or responsibility that comes with being a cleaning robot? Are there any additional factors we need to consider when dealing with cleaning robots compared to a regular robot?. For example: 1. A cleaning robot is treated as a different type of robot. 2. Robots include a status that identify if it as cleaning robot or not.	
Answer Options		**Answer Options**	
Choice 1: Official garages are a special kind of garage.	Choice 2: Official garages are similar to other garages.	Choice 1: cleaning robots are a special kind of robot.	Choice 2: cleaning robot are similar to other robots.

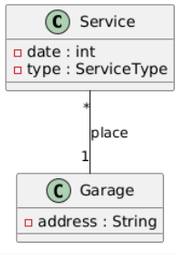
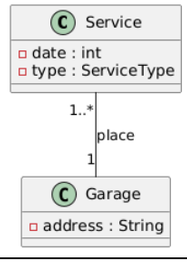
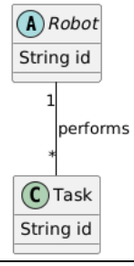
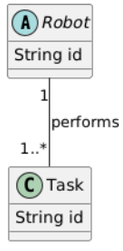
Composition vs Association		
Decision	A relationship is strong and affects the lifecycle (composition) or is between classes with independent lifecycle (association).	
Patterns	1. Association name includes 'has' or 'part': Alternative_1: Association, Alternative_2: Composition	
Template	*Does a {target_class} always require a {source_class} to exist, or could it exist independently and possibly be linked to multiple {source_class} instances? If a {source_class} is removed, would all the associated {target_class} also need to be removed, or could a {target_class} still make sense on its own?. For example:* *1. A {target_class} belongs to a {source_class} and cannot exist without a specific {source_class}.* *2. A {target_class} is linked to a {source_class} but can exist independently of it.* *Which option is better for you:* ***<Answers>***	
Answers	*Choice 1: a {target_class} is always for a specific {source_class}.*	*Choice 2: a {target_class} is independent and can still exist even if the {source_class} is removed.*
Actions	*1. Update relationship to Composition* *2. Set confidence for the Composition to high (Threshold x 1.1)* *3. Update the cardinality on the {source_class} side to 1* *4. Set confidence for the cardinality to high (Threshold x 1.1)*	*1. Update relationship to Association* *2. Set confidence for the Association to high (Threshold x 1.1)* *3. Update the cardinality on the {source_class} side to 1* *4. Set confidence for the cardinality to low (Threshold / 2)*

Domain: Car Service		**Domain: Robot tasks**	
			
Variables:		**Variables:**	
{source_class}: Car {target_class}: Service		{source_class}: Robot {target_class}: Task	
Question		**Question**	
Does a service always require a car to exist, or could it exist independently and possibly be linked to multiple car instances? If a car is removed, would all the associated services also need to be removed, or could a service still make sense on its own?. For example: 1. A service belongs to a car and cannot exists without a specific car. 2. A service is linked to a car but can exists independently of it.		Does a task always require a robot to exist, or could it exist independently and possibly be linked to multiple robot instances? If a robot is removed, would all the associated tasks also need to be removed, or could a task still make sense on its own? . For example: 1. A task belongs to a robot and cannot exists without a specific robot. 2. A task is linked to a robot but can exists independently of it.	
Answer Options		**Answer Options**	
Choice 1: a service is always for a specific car.	Choice 2: a service is independent and can still exist even if the car is removed.	Choice 1: a task is always for a specific robot.	Choice 2: a task is independent and can still exist even if the robot is removed.

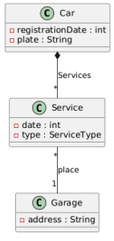
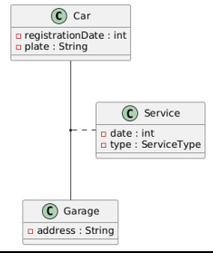
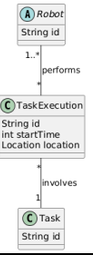
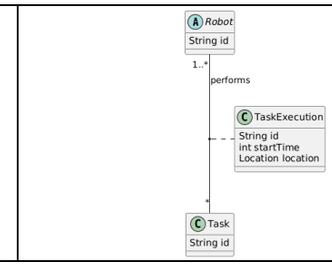
Upperbound Cardinality: 1 vs Many (*)		
Decision	The upperbound cardinality is to only one instance or multiple instances.	
Patterns	1. For all the upperbound cardinalities with 1: Alternative_1: Upperbound cardinality is 1, Alternative_2: Upperbound cardinality is * 2. For all the upperbound cardinalities with Many: Alternative_1: Upperbound cardinality is *, Alternative_2: Upperbound cardinality is 1	
Template	*In the context of a {source_class}, can there be more than one {target_class} involved, or is it always associated with just one?. For example:* *1. A {source_class} is linked to a single {target_class}.* *2. A {source_class} is linked to multiple {target_class}.* *OR* *In the context of a {target_class}, can there be more than one {source_class} involved, or is it always associated with just one?. For example:* *1. A {target_class} is linked to a single {source_class}.* *2. A {target_class} is linked to multiple {source_class}.* *Which option is better for you:* ***<Answers>***	
Answers	*Choice 1: Each {source_class} is always linked to a single {target_class}.* *OR* *Choice 1: Each {target _class} } is always linked to a single {source_class}.*	*Choice 2: A {source_class} can be linked to multiple {target_class}.* *OR* *Choice 2: A {target_class} can be linked to multiple {source_class}.*
Actions	*1. Update the cardinality on the {source_class} side from "1..*" to "1"* *OR* *1. Update the cardinality on the {target_class} side from "1..*" to "1"* *2. Set the confidence for the cardinality to high (Threshold x 1.1)*	*1. Update the cardinality on the {source_class} side from "1" to "1..*"* *OR* *1. Update the cardinality on the {target_class} side from "1" to "1..*"* *2. Set the confidence for the cardinality to high (0.9)*

Domain: Car Service		**Domain: Robot tasks**	
			
Variables:		**Variables:**	
{source_class}: Service {target_class}: Garage		{source_class}: Robot {target_class}: Task	
Question		**Question**	
In the context of a service, can there be more than one garage involved, or is it always associated with just one?. For example: 1. A service is linked to a single garage. 2. A service is linked to multiple garages.		In the context of a task, can there be more than one robot involved, or is it always associated with just one?. For example: 1. A task is linked to a single robot. 2. A task is linked to multiple robots.	
Answer Options		**Answer Options**	
Choice 1: Each service is always linked to a single garage.	Choice 2: A service can be linked to multiple garages.	Choice 1: Each task is always linked to a single robot.	Choice 2: A task can be linked to multiple robots.

Lowerbound Cardinality: 0 vs 1		
Decision	The lowerbound cardinality is to zero instance or one instance.	
Patterns	1. For all the lowerbound cardinalities with 0: Alternative_1: Lowerbound cardinality is 0, Alternative_2: Lowerbound cardinality is 1 2. For all the lowerbound cardinalities with 1: Alternative_1: Lowerbound cardinality is 1, Alternative_2: Lowerbound cardinality is 0	
Template	*In the context of a {target_class}, does it always involve at least one {source_class}, or can it exist without any?. For example:* *1. A {target_class} can exist without any {source_class}.* *2. A {target_class} must have at least one {source_class}.* *OR* *In the context of a {source_class}, does it always involve at least one {target_class}, or can it exist without any?. For example:* *1. A {source_class} can exist without any {target_class}.* *2. A {source_class} must have at least one {target_class}.* *Which option is better for you:* ***<Answers>***	
Answers	*Choice 1: A {target_class} can exist without any {source_class}.* *OR* *Choice 1: A {source_class} can exist without any {target_class}.*	*Choice 2: A {target_class} must have at least one {source_class}.* *OR* *Choice 2: A {source_class} must have at least one {target_class}.*
Actions	*1. Update the cardinality on the {source_class} side from "1..*" to "*"* *OR* *1. Update the cardinality on the {target_class} side from "1..*" to "*"* *2. Set the confidence for the cardinality to high (Threshold x 1.1)*	*1. Update the cardinality on the {source_class} side from "*" to "1..*"* *OR* *1. Update the cardinality on the {target_class} side from "*" to "1..*"* *2. Set the confidence for the cardinality to high (0.9)*

Domain: Car Service		**Domain: Robot tasks**	
			
Variables:		**Variables:**	
{source_class}: Service {target_class}: Garage		{source_class}: Robot {target_class}: Task	
Question		**Question**	
In the context of a garage, does it always involve at least one service, or can it exist without any?. For example: 1. A garage can exist without any service. 2. A garage must have at least one service.		In the context of a robot, does it always involve at least one task, or can it exist without any?. For example: 1. A robot can exist without any task. 2. A robot must have at least one task.	
Answer Options		**Answer Options**	
Choice 1: A garage can exist without any service.	Choice 2: A garage must have at least one service.	Choice 1: A robot can exist without any task.	Choice 2: A robot must have at least one task.

Association Class vs Class		
Decision	A concept is a class and association at the same time (association class) or it is only a class.	
Patterns	1. A relationship with many to many cardinality on both sides: Alternative_1: Association class, Alternative_2: Class	
Template	*A {source_class} has multiple {association_class} at different {target_class}. Should we think of the link between a {source_class} and a {target_class} as something that only exists when a {association_class} happens? Or would you say a {source_class} and a {target_class} should have a link even outside of specific {association_class}?. For example:* *1. A {source_class} has a direct relationship with a {target_class}. The {association_class} is a separate event that connects the {source_class} and {target_class} temporarily.* *2. The {association_class} is the key event that establishes the link between a {source_class} and a {target_class}. The {source_class} and {target_class} are only linked when a {association_class} occurs.* *Which option is better for you:* ***<Answers>***	
Answers	*Choice 2: A {source_class} and a {target_class} are always connected, even when no {association_class} happens.*	*Choice 2: The {source_class} and the {target_class} have a relationship only when a {association_class} happens.*
Actions	*1. Update class {association_class} to association class* *2. Set confidence for the association class to high (Threshold x 1.1)* *3. Create association from {source_class} to {target_class}* *4. Update the cardinality to "*" in the {source_class} side and "*" in the {target_class} side* *5. Set confidence for the associations to low (Threshold / 2)* *6. Set confidence for the cardinalities "*" to low (Threshold / 2)* *7. Remove the relationship from {source_class} to {association_class}* *8. Remove the relationship from {association_class} to {target_class}*	*1. Update association class {association_class} to class* *2. Set confidence for the class to high (Threshold x 1.1)* *3. Create association from {source_class} to {association_class}* *4. Update the cardinality to "1" in the {source_class} side and "*" in the {association_class} side* *5. Create association from { association_class } to {target_class}* *6. Update the cardinality to "1" in the {association_class} side and "*" in the {target_class} side* *7. Set confidence for the associations to low (Threshold / 2)* *8. Set confidence for the cardinalities "1" to low (Threshold / 2)* *9. Set confidence for the cardinalities "*" to high (Threshold x 1.1)* *10. Remove association between {source_class} and {target_class}*

Domain: Car Service		**Domain: Robot tasks**	
			
Variables:		**Variables:**	
{source_class}: Car {target_class}: Garage {association_class}: Service		{source_class}: Robot {target_class}: Task {association_class}: TaskExecution	
Question		**Question**	
A car has multiple services at different garages. Should we think of the relationship between a car and a garage as something that only exists when a service happens? Or would you say a car and a garage should have a connection even outside of specific services?. For example: 1. A car has a direct relationship with a garage. The service is a separate event that connects the car and garage temporarily. 2. The service is the key event that establishes the connection between a car and a garage. The car and garage are only linked when a service occurs.		A robot has multiple task execution at different tasks. Should we think of the relationship between a robot and a task as something that only exists when a task execution happens? Or would you say a robot and a task should have a connection even outside of specific task execution? . For example: 1. A robot has a direct relationship with a task. The task execution is a separate event that connects the robot and task temporarily. 2. The task execution is the key event that establishes the connection between a robot and a task. The robot and task are only linked when a task execution occurs.	
Answer Options		**Answer Options**	
Choice 1: A car and a garage are always connected, even when no service happens.	Choice 2: The car and the garage have a relationship only when a service happens.	Choice 1: A robot and a task are always connected, even when no task execution happens.	Choice 2: The robot and the task have a relationship only when a task execution happens.