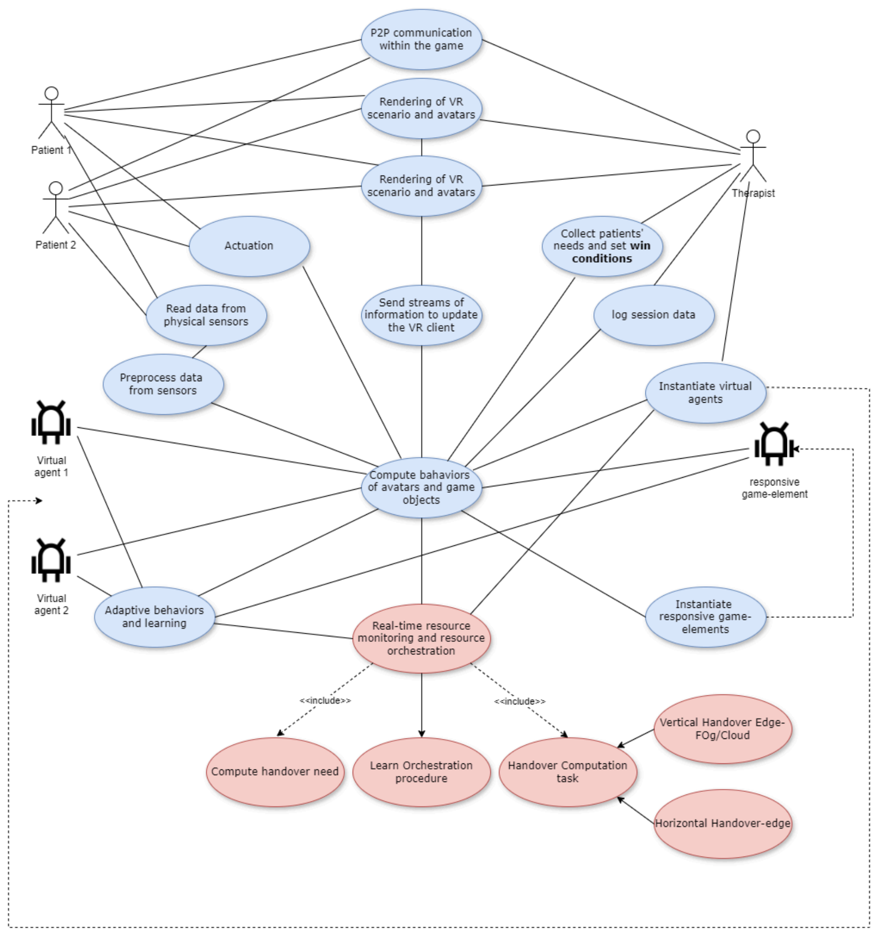
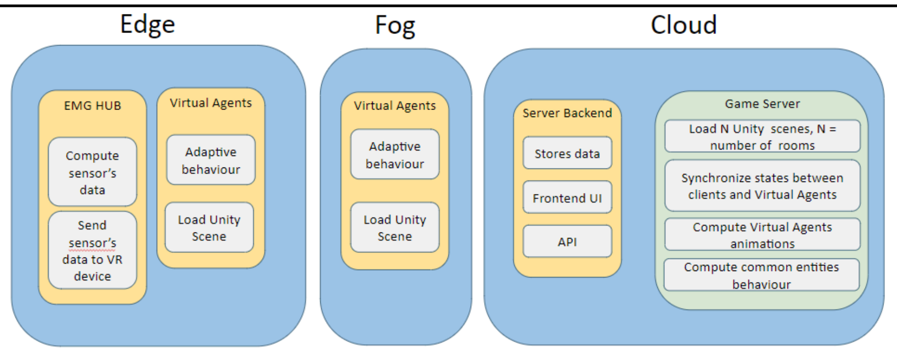
1.1 Healthcare Use Case - Swarm-inspired cooperative telerehabilitation

Healthcare UC	
UC identity data	
UC Owner	UNICA
Involved Partners	**UNICA MeDSP**: The UNICA MeDSP Lab defines the exergames and the key players who are part of it, the interaction models, the body signals acquisition and their integration in the Virtual Reality (VR) scenarios, also modelling pre-defined and probability-driven autonomous swarm behaviour for the Virtual Agents (VAs). It investigates possible exergames scenarios. Finally, it deals with defining the methods and tests for validating the proposed system. **REPLY**: Infinity Reply mainly works closely with UNICA in shaping and developing the Healthcare Use Case, by designing the exergames, defining the solution architecture, producing the 3D assets, and developing the networked VR multi-users layer.
Application Domain	*Healthcare*
UC overview	
Application Diagram *(Note that the high quality version of this diagram is attached in Appendix 1)*	 P2P communication within the game Rendering of VR scenario and avatars Rendering of VR scenario and avatars Patient 1 Patient 2 Therapist Actuation Collect patients' needs and set **win conditions** Read data from physical sensors Send streams of information to update the VR client log session data Preprocess data from sensors Instantiate virtual agents Virtual agent 1 responsive game-element Compute bahaviors of avatars and game objects Virtual agent 2 Adaptive behaviors and learning Real-time resource monitoring and resource orchestration Instantiate responsive game-elements <<include>> <<include>> Vertical Handover Edge-FOg/Cloud Compute handover need Learn Orchestration procedure Handover Computation task Horizontal Handover-edge

Application Diagram description	*[Note that just the main actors/blocks are described below]* The diagram considers both the physical/real world and the virtual environment/entities. Actors in this diagram are both physical (patients, P, and therapist, T) and virtual (virtual agents, VAs, and responsive game-element). ● **Patients** may be more than one, Patient$_{1, 2,\ldots N}$. ● The **therapist** may participate in the online training session along with the patients, but not necessarily, assuming a dual role: he/she can act as a patient (e.g. any Patient$_x$) or just see the scenes in “God” mode and selectively interact to set win conditions and instantiate VAs (therapist in the schema). The **win conditions** are the conditions/parameters that the patient must meet during the exercise to enable a progression toward the goal. For instance, in a dragon boat exergame, the patient should be able to correctly synchronise with the other rowers, beyond further movement-quality conditions, to push the boat efficiently along the course of a river, and reach the end of a maze: thresholds on the behaviour of the boat with respect to the actual synchronisation level to achieve movement, turning, etc, are examples of win conditions. The therapist can record and store data generated during a therapy session ("log session data'' block in the schema). This block is crucial for monitoring and analysing activities performed, patient progress, and therapy effectiveness. This information will be leveraged and appropriately synchronised by the “compute behaviours of avatar and game objects'' block in the schema. ● In contrast to physical users, the **VA** is an entity already integrated into virtual reality as a 3D model, instantiated by the game server according to the therapist's indication to support a specific patient. Indeed, the patients, because of their condition, may have mobility or interaction problems, thus exhibiting difficulties in performing the exergame. ● The **Responsive game-element** can be considered as another type of VA that is integrated into the virtual scene to be responsible for sensing and intervening in the evolution of the game by creating attractors, distractors, and in general environmental changes able to provide reward or penalty and to foster the development of specific behaviours in the group. The users (physical agents) wear the sensors and VR headset. All the sensors and devices worn by a user communicate with a hub (which can be represented simply by the VR headset in case of a reduced set of sensors) to provide information exploited to act in the virtual reality scene, such as translating the user's movement in the real environment into the avatar movement in the virtual environment. This function is implemented by the "read data from physical sensor" block in the schema. When a large number of sensors is used, an external hardware hub could be necessary to collect and preprocess sensors’ data in order to reduce the bandwidth for the communication with the headset and its computational burden. This function (either implemented on the hub or not) is implemented by the

	"preprocess data from sensor" block in the schema. Each VR headset sends the local information (position, orientation, and movements information, …) and receives the information coming from (a limited number of) the other agents, both virtual and real, in order to locally render this information and create a scene where all of them are correctly represented and animated. This function is implemented by the "compute behaviors of avatar and game objects" block in the schema; this is nothing more than the front-end of the game server, which is responsible for synchronising all the data and can be physically hosted in a low-latency cloud. Once the data have been synchronised, an action is generated (e.g., the boat will move down the river with a specific speed and direction). If the resulting action respects the conditions set by the therapist, then a global reward is assigned. In case this resulting action does not respect these rules, a penalty is assigned. This reward and penalty information is managed by the responsive game-element. For example, if the game is not evolving with an acceptable level of complexity even in terms of achieving a neurorehabilitative outcome for patients, the responsive game-element can engage the users by introducing attractors, distractors, etc. This function is implemented by the "instantiate responsive game-elements" block in the schema. The behaviour of the VA adapts to changes in the environment and the action of other users and is able to influence it, see "adaptive behaviour" block in the schema. To realise this behaviour, the VA should exchange information with the front-end of the game server as the headset of every physical user would do. As such, the VA can be computed in the game server, in the user clients, or in other cloud resources. Each user of the game (patients and therapist) can communicate directly with other participants through "P2P communication within the game" direct communication without the need for a central server. The VR environment, as well as all users participating in the game, will be generated through a three-dimensional creation and visualisation process ("Rendering of VR scenario and avatars" block in the schema). To ensure that any changes or information (user positions, movements and interactions) are immediately reflected in the VR environment the rendering block is closely related to the "Send stream of information to update the VR client" block. To handle the instantiation and execution of VAs, a process closely related to the computation request, there is the "Real-time resource monitoring and resource orchestration" block in the schema, which is connected to the game server and adaptive behaviours and learning. Patients and therapists are connected to an edge, and they will always be on edge. The VAs are not bound to a specific location. Therefore, they are connected to an orchestration flow that can decide upon their vertical (cross-layer, "vertical handover" block in the schema) or horizontal (intra-layer "horizontal handover" block in the schema) dispatching, depending on the current execution conditions. Learning and the adaptive behaviour of VAs can take place everywhere (fog or edge).

Deployment Diagram *(Note that the high quality version of this diagram is attached in Appendix 1)*	 Edge Fog Cloud EMG HUB Compute sensor's data Send sensor's data to VR device Virtual Agents Adaptive behaviour Load Unity Scene Virtual Agents Adaptive behaviour Load Unity Scene Server Backend Stores data Frontend UI API Game Server Load N Unity scenes, N = number of rooms Synchronize states between clients and Virtual Agents Compute Virtual Agents animations Compute common entities behaviour
Deployment Diagram description	In the Deployment Diagram, each actor is shown in the layer (edge, fog, cloud) it operates on. The **users** (patient or therapist) wearing the VR headset will have their computation done locally. The **therapists** (acting as an invisible observer) will have their computation done locally on a Windows desktop device. **Sensors** will send data to a local **HUB** at the edge, which processes the data, and sends it back to the VR user. **VAs** run either on the edge or the fog, depending on the decision made by the **MIRTO cognitive engine**. When the orchestrator detects excessive computational complexity on the edge, it might shift part of the computation to the fog. In the cloud, on the game server side, runs the **Network Synchronisation** entity that receives shared data from all users and ensures that each entity has a shared view of the scene. Common entities for all users will also run in the cloud. For example, in the game "Dragon Boat," the behaviour of the **boat** is calculated on the server side and must be the same for all connected clients. There is also a provision for storing patient report data. These can either be kept locally or uploaded to the server at the end of the session.
Offered services	**For the patient**: home-based or hospital/clinic-based telerehabilitation by advanced immersive and haptic virtual reality environment, with the challenge of collaborative tasks, no matter the number of simultaneously connected patients, for ● Cognitive and behavioural rehabilitation ● Motor rehabilitation ● Neuromotor rehabilitation **For the therapist**: possibility to monitor the patients' progress, interacting in the virtual environment with them, supporting or challenging them by the inclusion of VAs with specific goals.
Expected human role	**Patients are the main human actors**: they perform rehabilitation training in a virtual environment by interacting with other users in order to carry out a collaborative task, which involves both motor and cognitive loads. Patients can only control their own activity.

	Therapists are other actors in the system: they can control and change the parameters associated with specific patients to motivate them more. To support or challenge specific patients, they can instantiate VAs programmed to emulate specific behaviours (distractors, supporters, …). Moreover, they could directly communicate with individual patients by voice or video calls, or text messages, to provide feedback, or help and encourage them in a difficult situation.
UC Requirements	
Requested functionalities	Real-time multiplayer games rely on prediction to maintain the illusion of real-time synchronisation across all clients. The most common prediction algorithms used are: **Client-Side Interpolation**: This method makes movements smoother by guessing between known states. For example, if there's a delay in receiving an object's next position, the game engine can guess its movement, updating the position until new data arrives. The server periodically sends state updates to the client. Instead of immediately updating the positions of objects to the new values received, the client keeps a small "cache" of these states. The client then interpolates between these states, calculating intermediate positions based on the received data. For example, if the server sends updates every 100ms, the client can use this data to calculate intermediate positions every frame (e.g., 16ms for a game running at 60fps). Some examples of Client-Side interpolation algorithm are: ● **Linear Interpolation (LERP):** It calculates the intermediate position between two points (states) in a straight line. Formula: P(t)=(1−t)*P1 + t * P2 where P(t) is the interpolated position, P1 and P2 are the known positions, and t is the interpolation factor (0<=t<=1). ● **Cubic Interpolation:** This method uses a cubic function to calculate the intermediate position, providing smoother transitions than linear interpolation, especially during changes in direction. Formula: the formula can vary, but involves a cubic polynomial. an example is: P(t) = a * t^3 + b * t^2 + c * t + d where a,b,c and d are coefficients determined by the known points and their derivatives. ensuring a very smooth and natural transition. **Client-Side Prediction and Correction of Prediction**: This allows the game to predict the result of actions without waiting for server confirmation. It's useful for actions like movement, but requires a way to correct differences between the game's predictions and the server's state. Some example of Client-Side Prediction are: ● **Dead Reckoning**: It predicts the future position of an object based on its last known position, velocity, and direction. This method assumes that the object will continue moving in the same manner until new data is received from the server. Formula: $P_{predicted} = P_{last} + v + \Delta t$ where $P_{predicted}$ is the predicted position, P_{last} is the last known position, v is the velocity and Δt is the time elapsed since the last known position. ● **Input Prediction:** It involves the client predicting the game state by simulating the effects of user inputs before receiving confirmation from the server. Formula: $S_{predicted} = S_{current} + f(I)$ where $S_{predicted}$ is the

	predicted state, $S_{current}$ is the current state and f(I) is the function that applies input I. ● **Reconciliation:** It corrects the client's game state based on updates received from the server while maintaining a smooth gameplay experience. Step(s): i) Detects the discrepancy between the predicted state and the server's state. ii) Applies a correction over several frames to avoid a jarring transition. ● **Time Warp:** It is a method that involves rewinding the game state to a previous point in time to handle late-arriving inputs. Step(s): i) Saves a history of game states. ii) When a delayed input is received, rewinds to the appropriate state. iii) Applies the input and re-simulates all actions up to the current state. **Lag Compensation**: Servers can use techniques to account for a player's latency. This ensures that actions are evaluated as if they occurred immediately, despite the delay in data reception.
Expected Operating modes - User Stories	Flavio Rossi, a post-stroke patient followed by Dr. Andrea Giuliani, needs a neurorehabilitation session. The therapist suggested a one-hour training session three times a week; therefore, Flavio starts the first session this evening from the living room of his home. First, he wears the head-mounted display of the VR headset, and some wearable sensors controlling his muscle activity, switching on a sensor hub on his desk. He stands in an upright position grasping the VR controllers. In the virtual environment, his avatar, *Big*, is ready to start! In the virtual scenario, Flavio can see 3 other avatars, which represent other players available to complete the task with him. He is not able to distinguish whether the other players are real patients like him or VAs acting as patients. The therapist suggested playing the Dragon Boat game, which involves exploring a maze river by paddling a dragon boat. So, Flavio selects this training and presses "start". Initially, Flavio is requested to perform a reaching movement in front of him, to paddle with his avatar: for him, this is very difficult, due to his clinical condition that impaired his proper mobility in his upper limbs. At some point in the session, Flavio perceives from the scene that he is paddling against the will of the other rowers. So, he tries to change the direction of the arm movement and go in the opposite direction from the previous action. Nonetheless, he continues to find the movement complicated and is getting tired quickly. So, he slows down a bit. At a certain point, a message appears in front of Flavio, suggesting that he leave the session to move to a simpler one and better understand the movements he needs to make. As he accepts, he is loaded into a safe room simpler from the previous one. Flavio tries to synchronise with the rest of the avatars and realises that the motor task is easier for him on this boat. He starts to pay attention to the extent and quality of the movement too,

	discovering that larger movements are more effective while co-contractions of antagonistic muscles reduce effectiveness. Quality of movement is displayed as feedback, enabling Flavio to perceive his contribution to the boat movement, compared to the others, and to improve movement control. If he reaches the threshold, a message appears saying “you have completed the training session”, and he can choose to return to the homepage in order to start another game. In the next session, Flavio perceives that in this boat the pace is faster. The boat is moving faster now and proceeding in the right direction. At some point, Flavio realises that further ahead there is a deviation to the left of the maze, but the boat is not turning yet. Suddenly, a vortex attracts the boat and slows down its proper progress along the course of the river. Flavio then hears a voice, "Flavio! Good job so far, try to increase the speed of movement. Try to synchronise with the other rowers to get the boat to the end of the maze". Then Flavio stops the arm movement to raise the paddle on the left side of the boat and allows the other rowers to paddle only on the right side. Flavio then perceives that the boat is beginning to move in the right direction and the obstacle has been correctly avoided. Finally, Flavio sees that the end of the maze is approaching. When the game is over, Flavio sees "successfully completed training session" on the screen. The therapist would like to assist in Flavio's training session, so he decides to connect remotely. The therapist then sees the virtual scene evolving on his computer monitor and Flavio's avatar highlighted by a label with his name. The therapist had suggested to Flavio to play Dragon Boat in multiplayer mode. For each patient, the therapist fills out game conditions to be observed as minimum speed below which the boat cannot go, maximum deviation from the ideal path, etc. And so, knowing Flavio's clinical history, he had done the same for Flavio, figuring out an individualised rehabilitation program for him. But at one point the therapist notices that Flavio has difficulty keeping up with the pace of the boat driven by a set of 4 other patients further along in rehabilitation than him. So, the therapist decides that Flavio needs a guided session to get better ready before a new session. Later, the therapist decides that Flavio, having become familiar with the game, can return to the level he originally assigned him. Further later, as a game increases in the level of complexity of the task by adding elements to avoid, in order to increase the pacing or get a rest. To motivate the patient to complete the task despite the added difficulty, the therapist sends a vocal message to Flavio suggesting to increase the speed of arm movement and to cooperate and synchronise with the other rowing avatars to get around the obstacle.
Expected Operating modes - System Expected	The proposed system involves instantiating VAs based on the therapist's request or based on rules the therapist established. MIRTO cognitive engine will then have to make decisions about if and how to move the loads (the VAs). Since these are purely software entities, they could

Behaviour behind the User Stories	potentially run anywhere within the MYRTUS layered computing continuum infrastructure. Here follows different possible scenarios, considering higher complexity and computational demands. At the beginning Flavio is the only patient, P1, being initially unable to perform the task with other patients and needs to familiarise himself with the neurorehabilitation task on his own. After training with a single VA in a 2-seat boat, the system decides to pair him with 7 VAs, in the role of facilitators (normal boats have 8 seats). These VAs will be well-synchronised among them, and properly moving, well responsive to the environmental changes so that P1 can try to "follow" their behaviour. Normally, VAs could run on the sensor hub at the edge. However, such a high number of VAs might be too much for a single processing core. Therefore, MIRTO could operate vertical or horizontal orchestration to run them in the fog or in other edge devices. As Flavio, still P1, gets more experienced and progresses, he starts exercising with other real patients. Let’s consider the case that P1 is exercising, and also 4 other patients are rehabilitating at the same moment. According to the boat size, if more than 3 VAs are needed for the patients, they will be allocated to different boats in different room games. If the workload for any given edge machine is not acceptable, MIRTO could operate horizontal orchestration to better distribute the workload according to the available resources. Clearly, vertical orchestration is still an option also in this second scenario. Let’s assume a third case, due to a level switch of a patient, demonstrating more capabilities in managing the exercise, the game server reduces the number of VAs for both P1 and P2 to 2. Then, P1 and P2 can be transferred on the same boat with 4 VAs. Two further VAs will be added to fill the boat: also in this case, a workload reallocation between the edge and fog resources can be implemented.. Now, consider a sudden internet shutdown, which is why P2 exits the game. P2 must be excluded from the exercise, but the other patients must continue the exercise. In this case, it is necessary to instantiate 3 VAs on the fog, one replacing P2 and the others replacing his/her VAs. Similarly, if P2 exits the exercise, other VAs could be needed to enable exercising for the other patients.
Evolution and Learning capabilities	All agents, real and virtual, can learn to move in the virtual environment and participate in the collaborative task. The behaviour of VAs acting as patients is constrained by a predetermined set of local rules, which can evolve depending on the actions of patients and therapists, as well as the needs of the task (for example the level of difficulty). Some parameters associated to these behaviours can be tuned according to the experience of the VAs with patients. For instance, a rower can have a predefined paddling style (frequency, ...) that can be modulated to deal with the patients’ typical movements. Such adaptive behaviour during the exergame can be consolidated potentially through a learning strategy, so that the default behaviour is more adequate to a given population, with the possibility for the therapist to overload those self-tuned parameters.

	To this end, all nodes will have to collect and use both local information and the result of the actions performed by the rest of the involved agents to adapt their strategy. Beyond the VAs acting as patients, there will be VAs controlled by a specific module in the game, which is in charge of guessing the group behaviour, insert attractors and distractors, etc. Some learning could be implemented also at this level, with very simple AI models, in order to optimise the recognition of situations and the selection of the proper element to instantiate in order to overcome the situation. NOTE: Evolution and learning capabilities of the VAs are not the primary goal of this assessment scenario. Depending on the level of maturity of the project technologies at M18 and their adoption within the Use Case, the possibility of transferring some learning approaches developed for the MIRTO cognitive engine at the UC level may be exploited.
Technical Constraints	**Very low latency**: Visual feedback requires less than 50 ms in collaborative virtual environments, halved for haptic feedback and further reduced as the number of participants grows, as latencies introduce cross-effects. *Message latency*: < 50 ms **Bandwidth capacity**: It limits the number of participants locally represented in the virtual space. *Network bandwidth*: min 100 kbps per player, desiderate 1 Mbps per player **Power/Energy**: Low-power near-sensor processing and edge-to-fog data exchange. **Frame per second (FPS)**: Most video games today are developed to hit a frame rate of 60 FPS, but anywhere between 30 FPS and 60 FPS is considered acceptable. That's not to say that games cannot exceed 60 FPS, many do, but for anything below 30 FPS, animations may become choppy and show a lack of fluid motion. *Frame duration*: min 16.67 ms (60 FPS)
UC Evaluation	
Baseline	**Qualitative baseline**: Collaborative rehabilitation systems allow people with motor, cognitive and neurological deficits to perform recovery protocols in a stimulating environment, populated by subjects who can play different roles (patients, doctors/physiotherapists, family members, etc). The paramount value of this paradigm lies in the scientific evidence that subjects have a much higher performance if they interact with a partner, even intermittently, than those who perform the exercise alone. Both partners' performance and the nature of their behaviour influence both cognitive and motor improvements and can provide neuromotor adaptations in interacting humans, even when they are unaware of the partner's presence. The current state of the art in collaborative training aimed at rehabilitation is intended for pairs of individuals, typically patient-therapists. Only one multiplayer setup has been found in the scientific literature [TNG17], in which participants play in teams but each performing their own task independently (contributing individually to their team's score, but not cooperating in a collaborative task); furthermore, in this exercise the patient is required to perform reaching movements for a cognitive

	rehabilitation exercise based on an association image-writing rehabilitation pattern. This, at a motor level, can also be used to perform basic training for the upper limb. Also, player satisfaction with real-time multiplayer games is correlated with performance of the communications network. The network is the most dynamic component of such games; congestion and channel loss figure prominently in achieving bounds required for real-time response. If messages cannot be delivered on-time, game developers must use predictive techniques to maintain the game experience. The ability to tune this prediction with in-context measurements allows game developers to tune their prediction model for network conditions rather than designing around a single assumed level of network quality, thereby delivering even more convincing real-time experiences. **Quantitative Baseline**: Evaluation KPIs to measure: ● *User's acceptance/system usability*: e.g., Simulator Sickness Questionnaire (<10), System Usability Scale (>70%). ● *Treatment effectiveness* (on healthy volunteers): e.g., verification of the interactive initiative (exploration of motor strategy, by varying motor performance in terms of speed, fluidity, etc), evaluation of the effectiveness of the collaboration (task completion, completion time). There's no quantitative baseline value as the application is completely novel.
Expectation and needs	**Scalability**: it is necessary to use a multiplayer approach based on the creation of small groups, with a variable size from 2 to 6 players per group, in order to ensure that all participants play an active role in the collaborative task and that the coordination of participants is always efficient and stable. However, multiple groups can share the same playground, so that a global behaviour can be pursued. **Adaptivity**: the evolution of both the scenario and the collaborative task and the computational workload associated with them is variable, since it depends on the number of participants, on the number of the active nodes operating in a certain area of the virtual environment at a given time, on the approaches of artificial intelligence used to model (and possibly evolve) the behaviours of VAs, on the activation/deactivation of some computational resources, and on the number of sensors and actuators used at a given moment. **Cooperativeness**: this scenario is a system of systems, since each patient/therapist node is a system that includes a body network of physiological/biomechanical sensors and is part of a cluster of nodes, which in turn is a system, which can also interact with the other clusters. **Dependability**: the system protects the management of sensitive data. No relevant physiological signals will be shared by the system. **Qualitative goals**: ● Collaborative rehabilitation training used by more than 2 people at the same time without appreciable latency issues;

	● Collaborative rehabilitation training involving both physical patients and VAs; ● Collaborative rehabilitation paradigm based on local information and visual-haptic interaction only, following for example swarm intelligence principles; ● Possibility of integrating in the virtual experience information coming from multiple unconventional sensors in real-time; ● Low latency game server for an effective user experience. **Quantitative goals**: ● Increase of 50% of the number of physical and/or virtual actors involved in a collaborative rehabilitation training at the same time (more than 2 people); ● Successful integration in the game of at least 3 VAs; ● Reduction in the frequency of prediction algorithms intervention due to high latency >20%.