George Mason University’s Smart-Space Research Group, led by João Pedro Sousa, has a strong agenda on several fronts, ranging from applications to foundations of software for smart spaces.

CPSS (Cyber-Physical Safety Systems) looks at the spaces where we live, travel and work as caretakers of our safety. Current evacuation plans are static: signs posted on walls inform occupants of the nearest exit, or maybe of primary and secondary routes, but how is someone to know if that route is safe when a hazard strikes? What if the planned route is blocked by fire? What if the hazard is not easily recognisable to human eyes and the nearest stairwell is filling up with poisonous gas? These issues are especially relevant to large buildings, mass transit facilities and other installations that may attract malicious actions.

CPSS extends techniques developed in the autonomic computing community towards building cyber-physical safety systems that select safe routes dynamically based on awareness of the hazard – and even better, based on forecasting how the hazard is going to evolve over the next few minutes – and then communicates customised directions to occupants at each location in the building. Because a hazard may cripple or destroy parts of safety systems, CPSSs are self-healing, reconfiguring themselves in reaction to and in advance of failures, thanks to self-awareness and to hazard forecasting.

TeC empowers end users to define, personalise and evolve applications for smart spaces. TeC addresses a cost equation for software development that is impossible to solve with traditional methods: some applications are too personalised, too specific to circumstances, and may change too often to make it feasible to hire professionals to make every change. For example, users may want to configure their own solutions for home surveillance, for enhancing the independence of an elderly relative, or for controlling appliances and improving energy usage. However, today’s mainstream methods and tools for software development are inaccessible to end users.

TeC offers a spreadsheet-like declarative programming paradigm, where end users can build rules for the local behaviour of small devices, which coalesce into system-wide behaviours by extending virtual “wires” between those device abstractions.

Buddy (an architectural framework for peer-to-peer applications) enables teams of humans to gather and share contextual information over wide areas without relying on the availability of (communications to remote) servers or other infrastructural components. Application domains include disaster response, wildfire management, military and security operations in remote areas, search and rescue, crowd organisation, etc.

Both TeC and Buddy build on Team Computing, a new paradigm for distributed applications where computation is carried out by teams of small devices, such as phones and sensors, without assuming the availability of infrastructural components. Team Computing includes lightweight distributed protocols for self-healing of applications, and formal operational semantics that support model-checking techniques for the automated verification of desired behaviours.