Chapter 5
Architecting Next Generation Community Policing Solutions

Gohar Sargsyan, Raymond Binnendijk, and Eltjo Poort

Emerging Trends for Next Generation Community Policing and INSPEC^{2}T

"Community Policing" is a strategy of policing that focuses on police building ties and working closely with members of the communities. With the police innovation, the technology advancement, growing data and need of more complex and advanced analytics, new ways of collaboration between communities and Law Enforcement Agencies (LEA) is needed. "Next Generation Community Policing" (NGCP) introduces this these new ways, which aims at seamless collaboration between police and communities considering also aspects such as Societal, Ethical, Legal & Data Privacy, Criminology and Technological Applications related to the Geographic Information Systems, Smart Apps and integrated Networks.

The INSPEC^{2}T project is a next generation community policing research and innovation project. INSPEC^{2}T project (**I**nspiring Citize**NS** **P**articipation for **E**nhanced **C**ommunity Poli**C**ing Ac**T**ions (http://inspec2t-project.eu/en/) project is funded by the European Commission, under the "H2020-FCT-2014 Ethical/Societal Dimension Topic 2: Enhancing cooperation between law enforcement agencies and citizens – Community policing" call. The INSPEC^{2}T projects' scope is to develop a sustainable framework for Community Policing that effectively addresses and promotes seamless collaboration between the police and the community. In order to encompass the variety of European police cultures and contemporary experiences of police reform the consortium consists of eighteen partners from eight European countries (http://inspec2t-project.eu/en/partners-2). It consists of

The original version of this chapter was revised. An erratum to this chapter can be found at https://doi.org/10.1007/978-3-319-89294-8_13

G. Sargsyan (✉) · R. Binnendijk · E. Poort
CGI Group Inc., Rotterdam, The Netherlands
e-mail: gohar.sargsyan@cgi.com; raymond.binnendijk@cgi.com; eltjo.poort@cgi.com

G. Leventakis, M. R. Haberfeld (eds.), *Community-Oriented Policing and Technological Innovations*, SpringerBriefs in Criminology,
https://doi.org/10.1007/978-3-319-89294-8_5

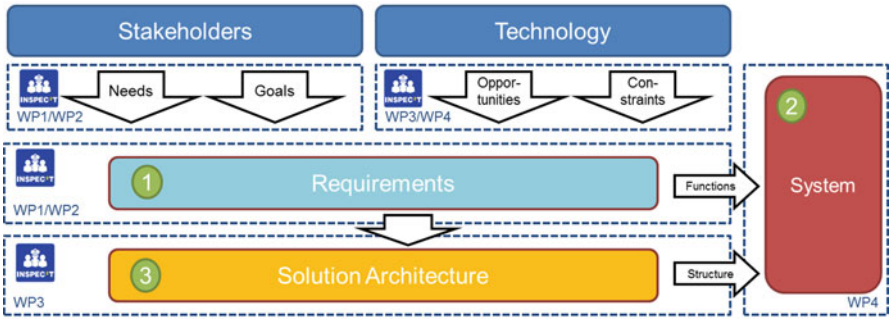
18 partners covering research, academia, business, non-profit and government organizations from 8 European countries and 45 more high level experts and stakeholder advisory group members coming from the worldwide. The INSPEC^{2}T project consists of eight Work Packages (WP's (http://inspec2t-project.eu/en/work-breakdown)), covering various aspects of the problem domain. A number of these WP's are related to the realization of an actual working IT-system. In 2017, this system is being demonstrated and validated in five EU cities (Valencia, Belfast, Engomi, Preston and Groningen) by relevant stakeholders in two phases.

This paper gives an overview of the architectural work done, led by CGI, supported by and in close collaboration with various consortium partners.

Architecture Work Package

Within INSPEC^{2}T, Work Package 3 (WP3) is responsible for architecture, which constitutes the central work package of the INSPEC^{2}T project. Through it the societal and functional requirements, defined in WP1 and WP2, are translated into technology, beyond the state of the art solutions, to be implemented in WP4.

The INSPEC^{2}T architecture process has taken place partly in parallel with the work on functional and non-functional requirements in Work Packages WP1 and WP2. This has allowed the architecture and requirements processes to mutually benefit from each other's progress, and resulted in cohesion between requirements and architecture.



The figure shows how WP3 is positioned among the other WPs. As indicated by the green numbers, architecture work consists of:

1. Making sure system requirements are clear, high-quality, feasible and "ready-for-consumption" for the system development (WP4).
2. Making sure all significant aspects are sufficiently covered, the architecture is aligned with the various technical opportunities and constraints and "technical partners" (the partners which are qualified for technological and technical expertise) are fully aligned and can hit the ground running when starting WP4.
3. Shaping the architecture.

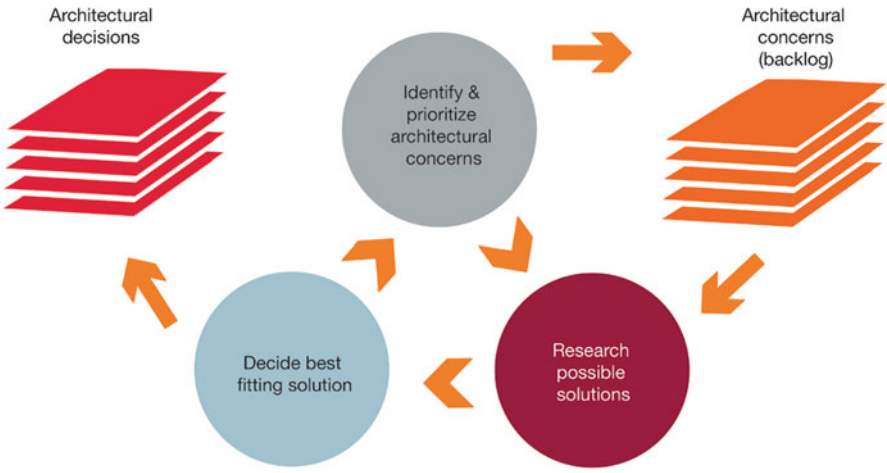
Risk and Cost Driven Architecture

The architecture approach followed in INSPEC^{2}T is CGI's Risk- and Cost-Driven (RCDA (Poort E./CGI), which is recognized in the Open Group Certified Architect program. RCDA was developed to close the gaps between enterprise and software architecture.

Existing software architecture practices often are too limited in scope for the solutions that need to be architected. However, enterprise architecture practices are too heavy for the agility required to manage time pressures and frequently occurring changes and uncertainty. RCDA incorporates a number of aspects from agile software development practices, such as the use of a backlog of architectural concerns, to be frequently reprioritized based on economic factors like risk and cost.

During the INSPEC^{2}T system architecture design, RCDA method supported the architects throughout the process of interpreting stakeholders' requirements, and subsequently designing and delivering the best fitting solution in a lean, mean and agile manner. Architectural concerns and architectural decisions are weighed throughout the process, and stakeholder requirements are constantly taken into account (Poort and van Vliet 2012).

The figure below describes the architectural design process:



When viewed as a risk and cost management discipline, architecture does not need to obstruct agility. RCDA offers a proven approach to solution architecture that is well-suited to today's agile end-users needs which is the base for INSPEC^{2}T.

Applying the RCDA method, setting up the INSPEC^{2}T architecture different practice sets are identified (requirement analysis, solution shaping, architecture validation and architecture fulfilment).

The application of RCDA brings various advantages to INSPEC^{2}T:

- It **smoothens communication** between solution architects and business stakeholders – RCDA-trained architects communicate about architectural decisions and trade-offs.
- A **clear and agreed set of architectural requirements** for design decisions, using objective and economically oriented trade-offs, rather than hypes or personal preferences.
- It reduces the risk of **delayed delivery and budget overruns** – RCDA sees architecture as a risk- and cost management discipline with economic awareness in the design process and avoiding "gold-plating".
- It **enhances the quality of solutions** – RCDA practices are CMMI (Capability Maturity Model Integration) compliant, and contain guidance for early and effective evaluation of quality attributes.
- It **creates transparency in costing structures** – RCDA provides traceability from architectural requirements to the costing model for the whole solution and its parts.

According to RCDA principles, the architecture work starts with identifying architectural concerns with the highest impact in terms of risk and cost, and addressing those concerns by making architectural decisions. Hence, the architecture contains the results of the most impactful architectural decisions made in the INSPEC^{2}T project and related various system requirements. These results have been documented in multiple *views*, where each view shows how the architecture addresses key stakeholder concerns.

Goals, Challenges and Opportunities

During the implementation of INSPEC^{2}T system design and architecture, the project partners agreed on the following approach: to facilitate a successful community policing solution the architecture should provide a foundation for an IT-system capable of connecting police and citizens (bi-directional). Enabling, managing and exploiting an information overload is another key success factor. In addition, the system has to be integratable in various, continually changing, IT-environments.

The architecture is shaped in close collaboration with eight EU technical partners. Each partner brings specific expertise (being brainpower and software products) to the project. Forging these building blocks together, leveraging and integrating the software and facilitating efficient collaboration, is both the primary challenge and a major opportunity to be addressed.

The INSPEC^{2}T project's work packages are primarily positioned sequentially (albeit partly overlapping and with some iterations included). This classical approach brings important structure to the project, but also reduces agility and the opportunity for including learning cycles.

Development Approach

We addressed these concerns by **communication continually** during weekly WebEx meetings, organizing an early physical meeting between technical partners, and creating a **collaboration platform** Content Management System (CMS) where the emerging architectural decisions and views were shared with all technical partners. These steps allowed us to quickly converge on an architecture. This **architecture** (elaborated on more later in this article) **allows for an incremental and iterative delivery strategy** of which an initial description was provided.

WP4 in INSPEC^{2}T project work-plan is responsible for implementation of INSPEC^{2}T system, as designed in WP3, developing also all back-end modules, by integrating prior implemented components, thus delivering it as a usable platform for performing the pilots. As part of the architecture delivery view a **development approach** was provided to develop or otherwise obtain the deliverable elements that make up the technical solution. This approach is being used by all technical partners during the execution of WP4 tasks directly related to a shared integrated environment. Each technical partner is free to decide on its local development approach.

The development of the integrated INSPEC^{2}T components is a continuous process which will contain all the required steps to assure quality during the entire lifetime of the project. This process can be represented as a virtual circle that contains the following functional components:

1. Source-code-versioning and management,
2. Continuous integration,
3. Quality assurance of generated code,
4. Builds (a.k.a. artefacts) management,
5. Issue tracking.

Each part of the circle is supported by mature pre-setup tools that interoperate smoothly. The tooling is part of the **central Integration test environment** provided by one of the technical partners.

Integration Guidelines

To furthermore address the concern of integration it's crucial to get the interfaces right (*focus on what's going on between the boxes*). This is why the architecture includes **interfaces guidelines**. Each technical partner should design and build its interfaces following these guidelines, or explicitly explain why a guideline is violated. These guidelines are:

1. Use REST (https://en.wikipedia.org/wiki/Representational_state_transfer).
2. Asynchronous chunky (not chatty) dialogs between components to promote loosely coupling and reduce overall bandwidth use.
3. Interfaces should abstract the underlying service complexities. Naming is semantic (not just CRUD (https://en.wikipedia.org/wiki/Create,_read,_update_and_delete) and thereby easy to understand by humans.
4. Transport big chunks of data using references. But no integration over shared databases.
5. Use choreography over orchestration for managing business process communication logic.
6. Avoid big bang breaking of interfaces by potentially supporting (two) coexisting interface versions.
7. Adapt Continuous Integration, as a crucial practice in securing successful integration.

Integration Plan

The development approach and interfacing guidelines provide technical partners guidance in "how" to develop and integrate their various components. In addition, the architecture provides guidance in "what" to develop, focusing on developing, deploying and integrating software components early and often, making sure the important (high impact on Risks or Costs) stuff is covered first. This guidance is provided via an **integration plan**, comprised of the following steps:

1. **Setup Integrated Test Environment**
   The central test environment, including tools support.
2. **UI Mock-up**
   Mock-ups of the UI components will look like the real thing, but will not do useful work beyond what the user sees. Mock-ups are used to discuss requirements and GUI/design-ideas.

3. **Interface Exemplar**
   Working code examples will help to understand written documents better. Start delivering a first draft version of an INSPEC^{2}T service focusing on implementing the interfaces, following the interface guidelines above. This service will be a real-world template for future services to be built by the other technical partners and will be part of and the starting point for the architectural prototype.
4. **Architectural Prototype**
   The INSPEC^{2}T prototype is built using all or the majority of INSPEC^{2}T components integrated and aligned with the architecture. The goal is to validate the architecture as early as possible. The prototype will be based on working software containing basic and preliminary functionality.
5. **Initial Version**
   A first version of the INSPEC^{2}T consisting of all INSPEC^{2}T components delivering at least all mandatory requirements. This version will be used for and will also be improved during system integration and technical testing.
6. **Pilots**
   During five test-case pilots INSPEC^{2}T used and validated in a real-life environment. Each pilot city will have a dedicated hosting environment and if needed a custom version of the INSPEC^{2}T system being a subset of the initial version and/or a configured version. Feedback from the various pilots will be consolidated and used to improve the INSPEC^{2}T system.

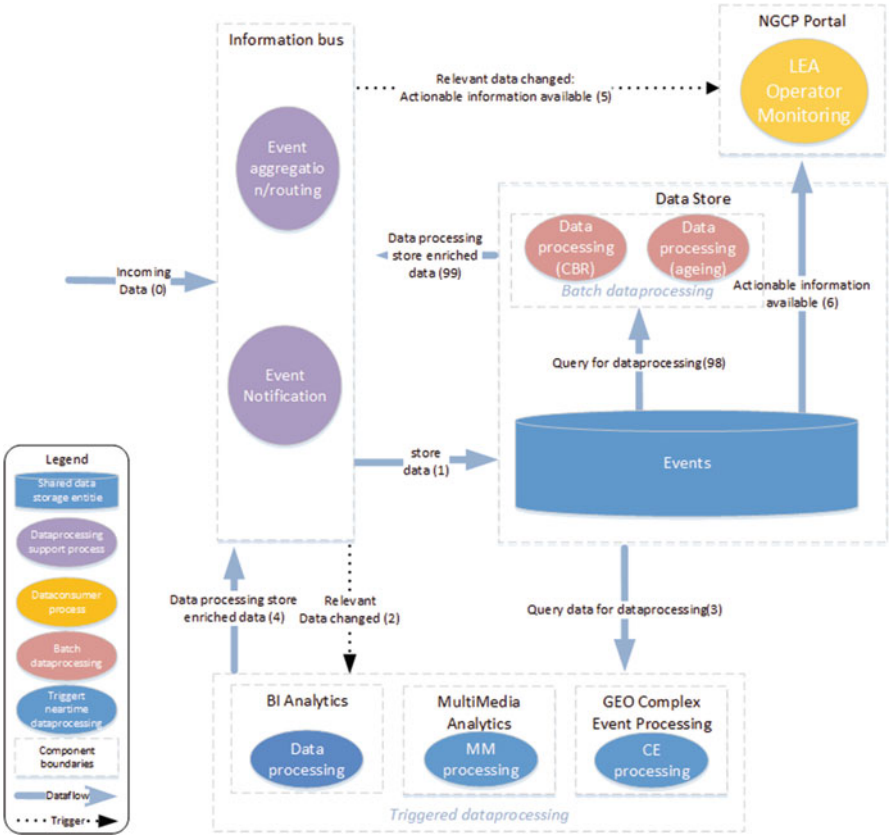
Steps 1, 2, 3 and 4 are part of WP3. This means the installation of environments, validation of the architecture and development and delivery of (preliminary versions of) the system already started during the architecture phase.

Open, Modular and Loosely Coupled Architecture

So, guidelines and steps, at the end of the day architecture is all about IT-system design. For Next Generation Community Policing the architecture should primarily address bi-directional information flow between people and systems and dealing with potentially large, changing and various flows of data. And last but not least, the architecture should facilitate flexible integration in a continually changing IT-landscape.

An open, modular and loosely coupled architecture meets these requirements.

This architecture is documented via a set of multiple views. The view below visualizes how components are loosely intercoupled dealing with information flowing through the system.



Incoming data is channelled through the information bus and stored centrally.

A **generic meta-data domain model** is in place as the base for data storage, transportation and message exchange schema's.

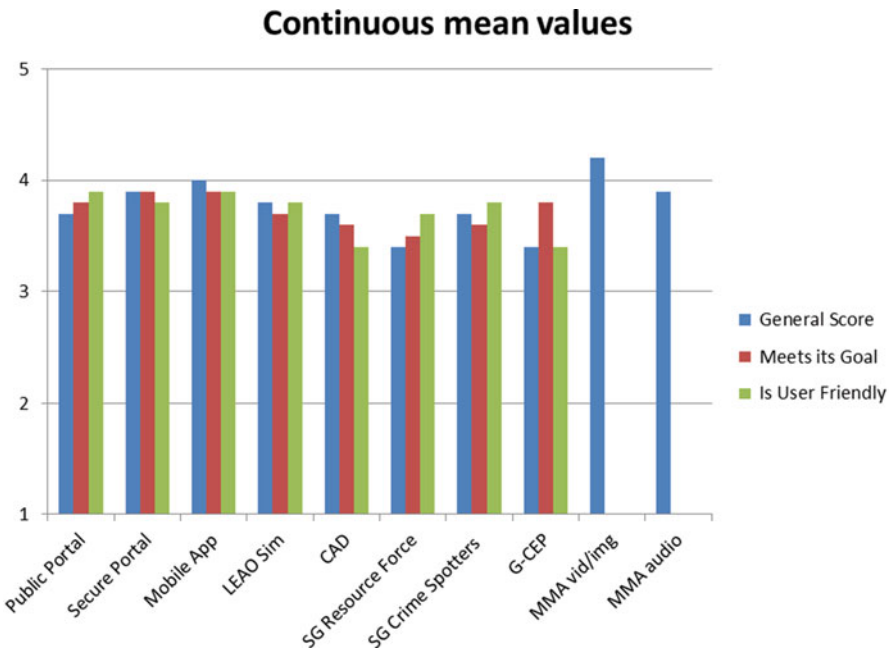
To provide more decoupling and modifiability a **choreographed** (instead of orchestrated), **publish/subscribe** design is used. This means the business process logic is not centralized but instead individual data processing components know what will trigger their data processing process. This way, if data processing components need to be added or removed they can just (un-)subscribe. No direct dependencies between data processing components and no need for code or configuration changes to a central component.

The various components are **connected loosely**, via **REST** interfaces.

Each component preferably is hosted using **virtual machines**, making physical **deployment flexible** and enhances **scalability**.

Architecture Validation Approach

Referring to the chapter "Integration plan", steps 1, 2, 3, 4 and 5 were demonstrated and validated within WP3 during the stakeholders advisory group and external expert group's meeting of the INSPEC^{2}T project on April 19, 2016. The system's rapid prototype has been demonstrated: after UI mock up presentations and architecture prototype demonstration, a questionnaire was offered to the senior LEA, Government and Community representatives to validate the architecture to allow us to move forward with the development of the system.



As illustrated in Figure, the rapid prototype validation containing all modules has been validated by the senior end-users, who are the project's (Stakeholder Advisory Board (SAG) / External Expert Group (EEG) members. For the criteria "General Score", 1 is low, or a bad score, and 5 is high, or a perfect score. For the criteria "Meets its Goal" and "Is User Friendly" selecting 1 means "Totally Disagree" and selecting 5 means "Totally Agree" (NSPEC^{2}T project's deliverable – D3.4 – 2nd SAG Report). Although overall assessment had positive scores,, however improvements and suggestions were provided which were taken on board for the development.

After the development of the system the Step 6 was conducted to validate the system during each of the 5 pilots in two phases. It is important to note, that the conclusion was that the architecture was solid and no breaks have been identified.

Designed for Integration and Evolution, More than a Set of Views

Architecture in today's world is not just about drawing a set of views, aimed to address current requirements. A sustainable NGCP-solution should be able to integrate in numerous heterogeneous environments and evolve over time. This is facilitated by an open, modular and loosely coupled ecosystem, designed in close collaboration with the various expert partners.

Acknowledgements This paper is based on the collaborative work conducted within the project INSPEC2T, the CGI's already existing practices and experience and the authors' vision on the subject. Having central role in setting up the framework (tools, architecture and methodologies) of the INSPEC^{2}T system, as part of the joint work plan, the authors received essential input from technical partners on the components under their development or their central development role, other partners with other valuable input which made the work complete. In particular, the authors would like to thank the following individuals and project partners: INTRASOFT – Sofia TSEKERIDOU, and Dane Vergeti, ADITESS –Elisavet Charalambous, Nikos Koutras, VICOMTECH – Peter Leskovsky, Santiago Prieto Calero, SATWAYS – Antonis Kostaridis, Leonidas Perlepes, KEMEA – George Leventakis, George Kokkinis, George Papalandrapolos, IMC – Spyros Tzavellas, Panos Georgolios, EXUS – Dimitris Katsaros, PLAYGEN – Kam Star.

References

CRUD. https://en.wikipedia.org/wiki/Create,_read,_update_and_delete

INSPEC^{2}T project (**I**nspiring Citize**NS** **P**articipation for **E**nhanced **C**ommunity Poli**C**ing Ac**T**ions). http://inspec2t-project.eu/en/

INSPEC^{2}T project (**I**nspiring Citize**NS** **P**articipation for **E**nhanced **C**ommunity Poli**C**ing Ac**T**ions). *Project partners*. http://inspec2t-project.eu/en/partners-2

INSPEC^{2}T project (**I**nspiring Citize**NS** **P**articipation for **E**nhanced **C**ommunity Poli**C**ing Ac**T**ions). *Work breakdown*. http://inspec2t-project.eu/en/work-breakdown

Poort, E., & CGI. *RCDA: Risk- and cost- driven architecture: Solution architecture for the agile age*. http://www.cs.vu.nl/~hans/publications/y2012/JSS-RCDA.pdf

Poort, E., & van Vliet, H. (2012). *RCDA: Architecting as a risk- and cost management discipline*. https://www.cgi.com/sites/default/files/white-papers/risk-and-cost-driven-architecture.pdf

REST – Representational State Transfer. https://en.wikipedia.org/wiki/Representational_state_transfer