Meta-Design with Safe and Secure Embedded System Networking

Miroslav Sveda
Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic
sveda@fit.vutbr.cz

Radimir Vrba
Faculty of Electrical Engineering and Communication
Brno University of Technology
Brno, Czech Republic
vrbar@feec.vutbr.cz

Abstract—**The paper presents several validated principles of a meta-design support for end-user development of safe and secure embedded system networking. The devised approach offers a reusable framework for Internet-based embedded system applications. Such resulting framework provides a flexible development environment kernel that can be adapted for various safety/security critical embedded system application domains. It stems from the IEEE 1451.1 smart transducer interface standard that provides an object-based networking model mediating efficient and unified access to distributed components through both wired and wireless networks. The paper discusses this framework not only from the viewpoint of framework builders, but also end-user developers. In this context, it demonstrates how to use that approach for a safety and security-critical application with Internet and ZigBee.**

Keywords-embedded system application; networking; security; safety; meta-design

I. INTRODUCTION

Not surprisingly, meta-design relates to design in the similar way as meta-modeling relates to modeling. But, while modeling and meta-modeling are identical activities with the only difference of interpretation, designing and meta-designing are targeted differently. Model, which is the object of modeling, remains an abstract notion in the similar sense as meta-model with the only exception of abstraction level. On the contrary, objects of design and meta-design differ: in the former case the process produces an artifact, in the letter case it is design of a design process including related development environment.

This paper [1] discusses a deployment of meta-design principles for building up a flexible design framework focused on embedded systems and their components inter-connected by Ethernet-based wired Internet and wireless ZigBee. Necessarily under-designed open source tools and techniques create design spaces for end-user developers. Hence, the paper demonstrates both the use of this framework for implementation of a development environment aimed at Internet-supported smart sensor applications and, concurrently, the utilization of this framework for development of pressure and temperature measurements and safety and security management along gas pipes.

The following section discusses end-user roles and deployment of meta-design principles in an embedded application development process, which is introduced from the end-user viewpoints. After that, the section 3 restates dependability principles characteristic for the considered application domain containing also industrial, safety and security critical applications.

The section 4 deals with related standards and standard structures that create not only solution constraints, but also design support. The subsections discuss subsequently the family of standards IEEE 1451 and, in more detail, standard IEEE 1451.1, which creates a framework foundation for logical design structures, standard ZigBee/IEEE 802.15.4 protocol profile, and TCP/IP-ZigBee interconnection structure.

The next section describes a case study based on a real industrial application that demonstrates utilization and refinement of the introduced design approach aiming at Internet-based, tiered architecture with wireless sensor networks. The applied design framework, which is conceptually based on the IEEE 1451.1 environment, is refined and extended to deal with pressure and temperature measurement and safety and security management along gas pipes. This section presents especially communication, safety and security issues, and resulting structure of the 1451.1 implementation providing efficient and unified access to distributed components through both wire and wireless networks.

The case study discusses the developed framework not only as a meta-design tool from the viewpoint of framework builders, but also from the viewpoint of end-user developers. In this context, it demonstrates how to use that approach for a safety and security-critical application with Internet and ZigBee.

II. META-DESIGN WITH END-USER

Typical system development process involves multiple participant roles [2]. Framework builders create the infrastructure for system components to interact; developers

[1] The current manuscript extends and updates the paper [1].

identify suitable domains and develop new components for them; application assemblers select domain-specific components and assemble them into applications; and end users employ component-based applications to perform daily tasks. Obviously, the fifth role can be included in this pipe-line: end-user developers positioned between application assemblers and end users. These end-user developers are able to tailor applications at runtime because they have both domain expertise and technical know-how. They can interact with applications to adjust individual components, and modify existing assemblies of components to create new functionality. Furthermore, they can play a critical role when component-based systems have to be redesigned for new requirements. End-user development activities can range from customization to component configuration and programming.

Meta-design offers techniques and processes for creating new environments allowing end users to act as designers [3]. In all design processes, two basic stages can be distinguished: design time and use time. At design time, system developers create environments and tools. In conventional design they create complete systems. Because the needs, objectives, and situational contexts of users can only be anticipated at design time, users often find the system unfit for their tasks at use time. Thus, they require adaptation of the existing environment and tools for new applications. Meta-design extends the traditional notion of system development to include users in an ongoing process as co-designers, not only at design time but throughout the entire life-cycle of the development process. Rather than presenting users with closed development systems, meta-design provides them with concepts and tools to extend the system to fit their needs. Hence, meta-design promotes designing the design process.

Evidently, meta-design not only promotes designing the design process, but also can support the end-user development of dependable embedded applications.

III. DEPENDABILITY

Dependability [4] is that property of a system that allows reliance to be justifiably placed on the service it delivers. A failure occurs when the delivered service deviates from the specified service. Dependability measures consist namely of reliability, availability, security, safety and survivability. Availability is the ability to deliver shared service under given conditions for a given time, which means namely elimination of denial-of-service vulnerabilities. Security is the ability to deliver service under given conditions without unauthorized disclosure or alteration of sensitive information. It includes privacy as assurances about disclosure and authenticity of senders and recipients. Security attributes add requirements to detect and avoid intentional faults. Safety is the ability to deliver service under given conditions with no catastrophic affects. Safety attributes add requirements to detect and avoid catastrophic failures.

A failure occurs when the delivered service deviates from the specified service. The failure occurred because the system was erroneous: an error is that part of the system state which is liable to lead to failure. The cause of an error is a fault. Failures can be classified according to consequences upon the environment of the system. While for benign failures the consequences are of the same order of magnitude (e.g. cost) as those of the service delivered in the absence of failure, for malign or catastrophic failures the consequences are not comparable.

A fail-safe system attempts to limit the amount of damage caused by a failure [5]. No attempt is made to satisfy the functional specifications except where necessary to ensure safety. A mishap is an unplanned event (e.g. failure or deliberate violation of maintenance procedures) or series of events that results in damage to or loss of property or equipment. A hazard is a set of conditions within a state from which there is a path to a mishap.

A fail-stop system never performs an erroneous state transformation due to a fault. Instead, the system halts and its state is irretrievably lost. The fail stop model, originally developed for theoretical purposes, appears as a simple and useful conception supporting the implementation of some kinds of fail-safe systems. Since any real solution can only approximate the fail-stop behavior and, moreover, the halted system offers no services for its environment, some fault-avoidance techniques must support all such implementations.

Obviously, design of any safe system requires deploying security to avoid intentional catastrophic failures. And vice versa, system's security can be attacked using a safety flaw. The greater the assurance, the greater the confidence that a security system will protect against threads, with an acceptable level of risk [6].

The above statement deals with trust, which is assured reliance on the character, ability, strength, or truth of someone or something [7]. Trust can be defined as the belief that an entity is capable of acting reliably, dependably, and securely in a particular case. In frame of network systems, trust is a complex subject that should be managed. Trust management entails collecting the information necessary to establish a trust relationship and dynamically monitoring and adjusting the existing trust relationship. Principally, security represents the combination of confidentiality, integrity and availability, and necessarily complements safety in safety-critical industrial applications.

IV. ENVIRONMENT

While preceding sections of this paper review methodology, i.e. meta-design, and required properties, i.e. safety and security, this section discusses architectural means that enable refining the framework into straightforward design and implementation of the networked sensor-based systems.

Embedded system networking concepts stem from hierarchically interconnected networks of various kinds: Internet, local area wire and wireless networks, and wireless

sensor networks. The logical hierarchy is usually based on tiered architectures that bring cost-effectiveness and scalability, and allow adapting straightforwardly to various application requirements.

Actually, networking appears a basic feature of promising embedded systems architectures. Internet access to individual components of distributed embedded systems can be based on both wire and wireless LAN technologies, predominantly on IEEE 802.3 and related Ethernet standards, and on IEEE 802.11b WiFi and associated wireless LAN protocols. Embedded systems and their components can be attached directly to Ethernet with TCP/IP protocol stack, but also indirectly or exclusively through various wired Fieldbuses or wireless technologies such as IEEE 802.11b WiFi, IEEE 802.15.1 Bluetooth, and IEEE 802.15.4 with related ZigBee. Sensor networks bring an important pattern with single base station connected to a wired network on one side and wirelessly to transducers, i.e. sensors and actuators, on the other side. When sensors are clustered, the base station communicates to cluster heads and through them to individual sensors. Moreover, the applications can use also ad-hoc wireless network architectures that enable to extend wireless part of the system network over physical limits and bring new dimension to fulfill application requirements.

The next subsections provide a brief outline of the IEEE 1451 and ZigBee communication architectures aimed at smart sensors and fitting embedded system networks of which smart sensor networks constitute essential subset.

A. IEEE 1451 Architecture

The IEEE 1451 consists of the family of standards for a networked smart transducer interface that include namely (i) a smart transducer software architecture, 1451.1, targeting software-based, network independent, transducer applications, and (ii) a standard digital interface and communication protocol, 1451.2, for accessing the transducer or the group of transducers via a microprocessor modeled by the 1451.1. The next three standards extend the original hard-wired parallel interface 1451.2 to serial multi-drop 1451.3, mixed-mode (i.e. both digital and analog) 1451.4, and wireless 1451.5 interfaces, see Figure 1.

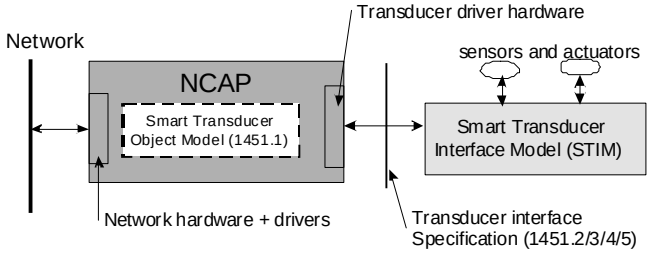

Figure 1. Smart transducer networking

The document 1451.0 complements the above standard set defining the structure of Transducer Electronic Data Sheets (TEDS), and associations between 1451.1 on one side and 1451.2/3/4/5 on the other side with message exchange protocols and command set for transducers.

The IEEE 1451.1 software architecture [8] provides three models of the transducer device environment: (i) the object model of a network capable application processor (NCAP), which is the object-oriented embodiment of a smart networked device; (ii) the data model, which specifies information encoding rules for transmitting information across both local and remote object interfaces; and (iii) the network communication model, which supports client/server and publish/subscribe paradigms for communicating information between NCAPs. The standard defines a network and transducer hardware neutral environment in which a concrete sensor/actuator application can be developed with respect to a concrete network.

The object model definition encompasses the set of object classes, attributes, methods, and behaviors that specify a transducer and a network environment to which it may connect. This model uses block and base classes offering patterns for one Physical Block, one or more Transducer Blocks, Function Blocks, and Network Blocks. Each block class may include specific base classes from the model. The base classes include Parameters, Actions, Events, and Files, and provide component classes.

Block classes form the major blocks of functionality that can be plugged into an abstract card-cage to create various types of devices. One Physical Block is mandatory as it defines the card-cage and abstracts the hardware and software resources that are used by the device. All other block and base classes can be referenced from the Physical Block.

The Transducer Block abstracts all the capabilities of each transducer that is physically connected to the NCAP I/O system. During the device configuration phase, the description is read from the hardware device what kind of sensors and actuators are connected to the system. The Transducer Block includes an I/O device driver style interface for communication with the hardware. The I/O interface includes methods for reading and writing to the transducer from the application-based Function Block using a standardized interface. The I/O device driver provides both plug-and-play capability and hot-swap feature for transducers.

The Function Block provides a skeletal area in which to place application-specific code. The interface does not specify any restrictions on how an application is developed. In addition to a State variable that all block classes maintain, the Function Block contains several lists of parameters that are typically used to access network-visible data or to make internal data available remotely.

The Network Block abstracts all access to a network employing network-neutral programming interface supporting both client-server and publish-subscribe patterns for configuration and data distribution.

B. ZigBee Architecture and Security

The ZigBee/IEEE 802.15.4 protocol profile [9], [10] is intended as a specification for low-powered wireless networks. ZigBee is a published specification set of high level communication protocols designed to use small low power digital radios based on the IEEE 802.15.4 standard for wireless personal area networks. The document 802.15.4 specifies two lower layers: physical layer and medium access control sub-layer. The ZigBee Alliance builds on this foundation by providing the network layer and the framework for application layer, which includes application support sub-layer covering ZigBee device objects and manufacturer-defined application objects.

Responsibilities of the ZigBee network layer include mechanisms used to join and leave a network, to apply security to frames and to route frames to their intended destinations. In addition to discovery and maintenance of routes between devices, including discovery of one-hop neighbors, it stores pertinent neighbor information. The ZigBee network layer supports star, tree and mesh topologies. Star topology network is controlled by one single device called ZigBee coordinator, which is responsible for initiating and maintaining devices on the network. Those devices, known as end devices, directly communicate with the ZigBee coordinator. In mesh and tree topologies, the ZigBee coordinator is responsible for starting the network and for choosing key network parameters.

The ZigBee application layer includes application support sub-layer, ZigBee device objects and manufacturer-defined application objects. The application support sub-layer maintains tables for binding, which is the ability to match two devices together based on their services and their needs, and forwards messages between bound devices. The responsibilities of the ZigBee device objects include defining the role of the device within the network (e.g., ZigBee coordinator or end device), initiating and/or responding to binding requests and establishing a secure relationship between network devices. The ZigBee device object is also responsible for discovering devices on the network and determining which application services they provide.

The ZigBee specification defines also techniques and services for safety and security support [11]. To increase reliability of data transmission, DSSS (Direct Sequence Spread Spectrum) is used. For keeping message integrity, it is possible to use 0, 32, 64, 128 bits data integrity options. Sequential data transmission guarantees the freshness that protects the network against replay attack when the attacker replays an older message to obtain answer. Each ZigBee device manages counters for maintaining data freshness.

Authentication on the network level using a common network cipher key protects the system against attacks from outside of the network with minimal memory requirements. Authentication on the device level is achieved by a unique link key shared by two communicating devices. This is prevention against attacks both from inside and outside of the network, but it leads to higher memory requirements. Confidence provides protection of the information in face of unauthorized users by data encryption. Data encryption prevents to learn only a part of message; we talk about semantic security. One of the features of semantic security is initial value, which is used with encryption. When the same message is encrypted two times, two different crypto texts are produced. ZigBee employs Advanced Encryption Standard (AES) with 128 bits key. Encryption and decryption is performed on the network level or on the device level. The encryption on the network level employs network key, which ensures protection against outside of the network. The second option is a protection on the device level. In this case we used a link key is used between two devices, what ensures protection against attacks both from inside and outside of the network.

C. TCP/IP-ZigBee Interconnection

According to the ISO Open Systems Interconnection vocabulary, two or more sub-networks can be interconnected using equipment called as intermediate system whose primary function is to relay selectively information from one sub-network to another and to perform protocol conversion where necessary. A bridge or a router provides the means for interconnecting two physically distinct networks, which differ occasionally in two or three lower layers respectively. The bridge converts frames with consistent addressing schemes at the data-link layer while the router deals with packets at the network layer. Lower layers of these intermediate systems are implemented according to the proper architectures of interconnected networks. When sub-networks differ in their higher layer protocols, especially in the application layer, or when the communication functions of the bottom three layers are not sufficient for coupling, the intermediate system, called in this case as gateway, contains all layers of the networks involved and converts application messages between appropriate formats.

An intermediate system represents typically a node that belongs simultaneously to two or more interconnected networks. The backbone network interconnects more intermediate systems that enable to access different networks. If two segments of a network are interconnected through another network, the technique called tunneling enables to transfer protocol data units of the end segments nested in the proper protocol data units of the interconnecting network.

Gateways and bridges offer two different ways how to provide connectivity in between TCP/IP and ZigBee networks. In context of ZigBee, gateways provide a full featured connectivity and allow a greater diversity of devices and applications that can be interconnected by ZigBee networks. Bridges are much simpler than gateways but serve a smaller application space. Gateway is a device that allows disparate networks to exchange information. Gateways allow wireless sensor networks to use wireless protocols such as ZigBee that are well suited for the harsh RF environment as well as battery powered applications and allow them to be integrated into existing applications. Gateways convert the

wireless protocols and sensor data into various formats necessary for industrial, commercial, and residential systems.

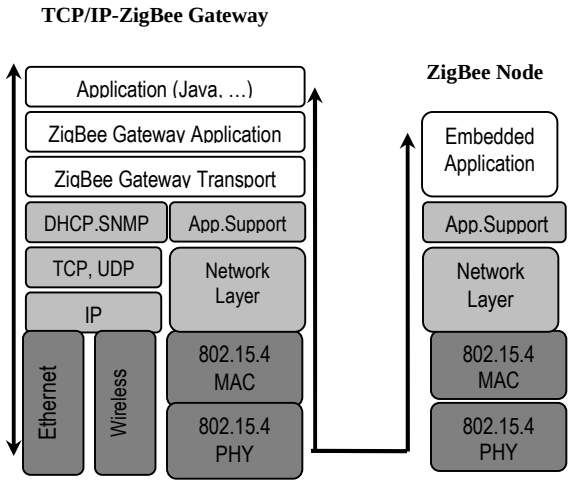

Figure 2. Gateway layered architecture

The ZigBee Gateway on Figure 2 provides an interface between ZigBee and IP devices through an abstracted interface on IP side. The ZigBee Gateway translates both addresses and commands between ZigBee and IP.

V. CASE STUDY

This section describes a case study based on a real industrial application that demonstrates utilization and refinement of the introduced framework aiming at Internet-based, tiered architecture with wireless sensor networks. The framework, which is conceptually based on the IEEE 1451.1 environment, is refined and extended to deal with pressure and temperature measurement and safety and security management along gas pipes. The related implementation stems directly from the IEEE 1451.1 model with Internet and the IEEE 1451.5 wireless communication based on ZigBee running over the IEEE 802.15.4.

The case study presents the framework both as a product of meta-design and as a design means including its refinement providing the final implementation.

A. Safety and Security Issues

The application architecture comprises several groups of wireless pressure and temperature sensors with safety valve controllers as base stations connected to wire intranets that dedicated clients can access effectively through Internet, see Figure 3. The WWW server supports each sensor group by an active web page with Java applets that, after downloading, provide clients with transparent and efficient access to pressure and temperature measurement services through controllers. Controllers provide clients not only with secure access to measurement services over systems of gas pipes, but also communicate to each other and cooperate so that the system can resolve safety and security-critical situations by shutting off some of the valves.

Each wireless sensor group is supported by its controller providing Internet-based clients with secure and efficient access to application-related services over the associated part of gas pipes. In this case, clients communicate to controllers using a messaging protocol based on client-server and publish-subscribe patterns employing 1451.1 Network Block functions. A typical configuration includes a set of sensors generating pressure and temperature values for the related controller that computes profiles and checks limits for users of those or derived values. When a limit is reached, the safety procedure, which is derived from the fail-stop model discussed above, closes valves in charge depending on safety service specifications. Examples include too low pressure for a pipeline segment, which means a gas leakage, or too high temperature, which means a danger of explosion. In both cases the safety procedure provides a pipeline reconfiguration by shutting off/opening some of the valves.

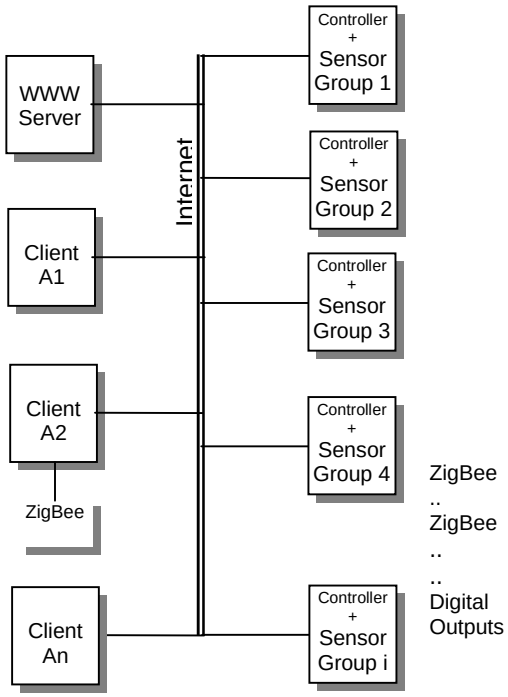

Figure 3. Network configuration

Security configurations in this case can follow the tiered architecture mentioned previously. To keep the system maintenance simple, all wireless communication uses standard ZigBee hop-by-hop encryption based on single network-wide key because separate pressure and/or temperature values, which can be eavesdropped, appear useless without the overall context. Not surprisingly, the application deploys standard ZigBee authentication.

Security support in frame of Intranet subnets stems from current virtual private network concepts. The discussed application utilizes ciphered channels based on tunneling between a client and a group of safety valve controllers. The tunnels are created with the support of associated authentications of each client.

B. Structure of the 1451.1 Implementation

The 1451.1 network model provides an application interaction mechanism supporting both client-server and publish-subscribe paradigms for event and message generation and distribution. Controllers play the role of clients or subscribers for the wireless part of the system network, and the role of servers or publishers for the wired part. Moreover, they compute temperature and pressure profiles, check the limit values and handle the safety valves.

In the transducer's 1451.1 object model, basic Network Block functions initialize communication between a client, which passed an authentication procedure, and the controller identified by a unique unicast IP address. The client-server style communication, which in this application covers both the configuration of controllers and initialization actions, is provided by two basic Network Block functions: *execute* and *perform*. The standard defines a unique ID for every function and data item of each class. If the client wants to call some function on server side, it uses command *execute* with appropriate parameters. On server side, this request is decoded and used by the function *perform*. That function evaluates the requested function with the given arguments and, in addition, it returns the resulting values to the client. Those data are delivered by requested variables in *execute* arguments.

The publish-subscribe style of communication, which in this application covers primarily distribution of measured data, but also distribution of group configuration commands, employs IP multicasting. All regular clients wishing to receive messages from a controller, which is joined with an IP multicast address, register themselves to this group. After that, when this controller generates a message by Block function *publish*, this message is delivered to all members of this group, without unnecessary replications.

Each controller communicates wirelessly with its sensors through 1451.5 interfaces by proper communication protocol. In the discussed case the P1451.5-ZigBee protocol, which means ZigBee over IEEE 802.15.4, was selected because it fits application requirements, namely those dealing with power consumption, response timing, and management.

C. Sensor Node Implementation

A typical node configuration, depicted on Figure 4, consists of STIM (Smart Transducer Interface Module) connected with a PSD sensor for differential pressure measurements, and with auxiliary temperature sensor for signal conditioning.

Of course, NCAP can be either embedded in a complex smart sensor, or shared among more simple smart sensors. On the other hand, from the viewpoint of Internet, only NCAP is directly addressable being equipped by its own IP address. Therefore, we can also denote as smart sensor the device consisting of an NCAP accessing one or more STIMs with connected sensors.

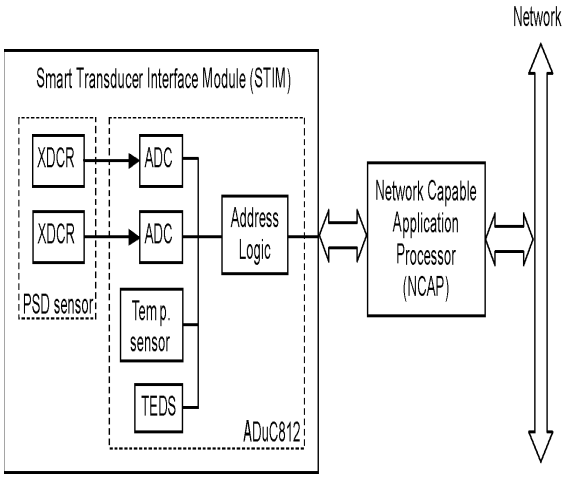

Figure 4. Sensor node example

This example discusses the pressure sensors with reflected laser beam and diffractive lens. The sensitive pressure sensor is based on a nitride membrane and an optoelectronic read-out subsystem. Measured pressure values are transformed into related thick-layer nitride membrane deflections. The nitride membrane serves as a mirror for laser beam, and it can move the related reflected laser mark. The mark's position is sensed using position-sensing device, which is a photo-lateral diode. Diode double current signal is amplified and conditioned digitally by the ADuC812 microcontroller. This microcontroller provides also the IEEE 1451.5 interface.

The sensing subsystem combines two principles that provide both high precision and wide range pressure measurements. Large displacements are measured by the position of reflected focused laser beam. Small position changes are measured by one-side layer diffractive lens principle. Sensor output signal is conditioned in digital by the ADuC812 single-chip microcontroller, which provides, with a simple hardware support, the IEEE1451.5 interface as one of its communication ports. This microcontroller calculates the position of the light spot and converts that position on the measured pressure using an internal table. Figure 4 depicts principles of the implementation of that smart sensor. The STIM contains (1) a PSD sensor with two analog differential transducers (XDCR), (2) a microcontroller ADuC812 with nonvolatile memory containing a TEDS field (Transducer Electronic Data Sheet) that props IEEE 1451.5 storing sensor specifications, (3) a TII (Transducer Independent Interface), (4) a temperature sensor necessary

for signal conditioning, (5) an analogue-to-digital conversion units (ADC), and (6) a logic circuitry to facilitate communication between the STIM and related NCAP.

The ADuC812 microcontroller, the basic building block of the smart pressure sensor electronics, includes on-chip high performance multiplexers, ADCs, FLASH program and data storage memory, an industrial standard 8052 microcontroller core, and supports several serial ports. The microcontroller may also utilize nonvolatile memory containing a TEDS field.

In this case, STIM represents a smart sensor serially interconnected with NCAP that provides controller functions and accesses Internet. Of course, each NCAP may be connected to many smart sensors, can access dedicated WWW server, and can be accessed by clients registered in related multicast group.

D. Design Pattern

In conclusion of this case study it could be helpful to restate the crucial architectural refinements of the proposed framework through building up a design pattern based on its basic abstract components, i.e. IEEE 1451 architecture, communication procedures and IP multicasting on the Internet side, and ZigBee Gateway on ZigBee side, which introduce a more detailed structure reusable for similar applications.

The 1451.1 object model provides skeleton supporting individual components. Its Network Block is refined so that it enables to access data-link layer communication services through unicast on IP with client-server procedure for start-up configuration and run-time maintenance, or through IP multicast with publish-subscribe procedure for run-time process measurements on both application data users and transducers sides. This refinement covers also selection of the most appropriate multicast routing protocol for local Internet traffic in case when the relevant parts of the network are accessible through routers.

The Transducer Block includes methods for reading and writing to transducers from the application-based Function Block using the standardized interfaces. The I/O device driver provides both plug-and-play capability and hot-swap feature for each transducer. It enables run-time reconfiguration of sensors that can support robustness of the system or improve measurement efficiency.

The Function Block contains a measurement application code. In the current case it prescribes sampling times, data filtering, linearization, conversions and transformations improving measurement accuracy and stability. The Function Block contains lists of parameters used to access net-work-visible data and, concurrently, to make internal data available remotely. The Function Block enables in this case to compute pressure profiles along the pipeline, pressure or temperature gradients, and the speed of pressure or temperature changes in time.

The ZigBee Gateway provides a prototypical interface between ZigBee and IP devices through an abstracted interface on IP side. The gadget translates both addresses and commands between ZigBee and IP standard architectures. Instead of that gateway, much more simple bridge can be used; unfortunately, such type of interconnection restricts substantially functionality of the application.

From more general viewpoint, the design pattern provides embodiment of meta-design principles for creating flexible design environments that can support development of various dependable applications respecting not only special functional requirements, but also requirements on system's safety and security. Necessarily under-designed open source tools and techniques create an open design space for end-user developers in various application domains of networked embedded systems.

VI. CONCLUSION

Internet technologies complemented by wireless access networks are rapidly becoming the preferred choice for building next generation distributed measurement and control systems. The framework, which can provide for current trends, stems from the IEEE 1451.1 standard specifying smart transducer interface architecture that enables to unify interconnecting smart sensors and sensor-based embedded systems with various wireless networks, and their direct coupling to recent Ethernet-based Intranets. Supporting techniques included in the frame-work, namely publish-subscribe messaging by IP multicast and security maintenance, offer scalable and traffic-saving solution important from viewpoint of the contemporary Internet. The schemes discussed can properly interplay with each other and can supply suitable support for design of networked, sensor–based embedded system applications.

This paper discusses a deployment of meta-design principles for building up an adaptable design framework focused on embedded systems and their components interconnected by Internet and ZigBee. Necessarily under-designed open source tools and techniques create design spaces for end-user developers. In that way the meta-design approach supports reusability among various application domains. Explicitly, the paper demonstrates both the use of this framework for implementation of development environment aimed at Internet-based smart sensor applications and, concurrently, the utilization of this framework for development of pressure and temperature measurement and safety and security management along gas pipes. The paper brings this design approach in manner suitable not only for framework builders, but also for end-user developers in a variety of application domains.

The current manuscript, which extends and updates the paper [1], stems partly from the previous research tasks published in [12], [13], and [14]. As a main contribution it delivers meta-design approach to a real world example of the design framework's stepwise refinement for a safety and security-critical sensor networking application.

A. Future Work

The next related research will be focused on the deployment of formal specification techniques including related tools for the industrial embedded systems domain, in more detail mentioned in the paper [15]. We will strive to create a design and development environment supporting automatic or semi-automatic generation of executable prototypes with behaviors satisfying not only functional requirements, but also safety and security constraints, from verified formal specifications.

ACKNOWLEDGMENT

The research has been supported by the Czech Ministry of Education in frame of the Research Intentions MSM 0021630528: Security-Oriented Research in Information Technology and MSM 0021630503: MIKROSYN -- New Trends in Microelectronic Systems and Nanotechnologies, and by the Grant Agency of the Czech Republic through the grant GACR 102/08/1429: Safety and Security of Networked Embedded System Applications.

The authors acknowledge also contributions to this research by their colleagues from the Networks and Embedded Systems Research Group and from the Secure and Reliable Network Architectures Research Group at the Department of Information Systems, Faculty of Information Technology of the Brno University of Technology, and from the Department of Microelectronics, Faculty of Electrical Engineering and Communication of the Brno University of Technology.

Initially, this work was supported also by the Grant Agency of the Czech Republic through the grants GACR 102/05/0723: A Framework for Formal Specifications and Prototyping of Information System's Network Applications and GACR 102/05/0467: Architectures of Embedded Systems Networks.

REFERENCES

[1] M. Sveda and R. Vrba, "Meta-Design Support for Safe and Secure Networked Embedded Systems", Proceedings Third International Conference on Systems, ICONS 2008, IARIA, Published by the IEEE Computer Society, 2008, pp. 69-74.

[2] A.I. Morch, et al., "Component-Based Technologies for End-User Development", Communications of the ACM, Vol.47, No.9, 2004, pp.59-62.

[3] G. Fischer, et. al., "Meta-Design: A Manifesto for End-User Development", Communications of the ACM, Vol.47, No.9, 2004, pp.33-37

[4] B. Melhart and S. White, "Issues in Defining, Analyzing, Refining, and Specifying System Dependability Requirements", Proceedings of the IEEE Conference and Workshop ECBS'2000, IEEE Computer Society, Edinburgh, Scotland, 2000, pp.334-340.

[5] N.G. Leveson, "Software Safety in Computer-Controlled Systems", IEEE Computer, February 1984, pp.48-55.

[6] I.-G. Kim, et al., "Formal Verification of Security Model using SPR Tool", Computing and Informatics, Vol.25, No.5, 2006, pp.353-368.

[7] Li, H. and M. Singhal, "Trust Management in Distributed Systems", IEEE Computer, Vol.40, No.2, 2007, pp.45-53.

[8] IEEE 1451.1, Standard for a Smart Transducer Interface for Sensors and Actuators -- Network Capable Application Processor (NCAP) Information Model, IEEE, New York, USA, 2000.

[9] IEEE 802.15.4, Wireless Medium Access Control and Physical Layer Specification for Low-Rate Wireless Personal Area Networks, IEEE, New York, USA, 2003.

[10] ZigBee: ZigBee Specification. ZigBee Alliance Board of Directors, 2006, Website http://www.zigbee.org/.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[11] P. Baronti, et al., "Wireless Sensor Networks: A Survey on the State of the Art and the 802.15.4 and ZigBee Standards", Computer Communications, Vol. 30, 2007, pp.1655-1695.

[12] M. Sveda, "End-User Development Framework for Embedded System Applications", Proceedings 14th IEEE International Conference on the Engineering of Computer-Based Systems ECBS07, IEEE Computer Society, Tucson, Arizona, USA, 2007, pp.186-192.

[13] M. Sveda and R. Vrba, "Dependability-driven Embedded Systems Networking", Proceedings 6th International Conference on Networking ICN 2007, IARIA, Published by the IEEE Computer Society, 2007, pp.483-488.

[14] M. Sveda and R. Trchalik, "Safety and Security-driven Design of Networked Embedded Systems", Proceedings 10th Euromicro Conference on Digital Systems, Digital System Design Architectures, Methods and Tools, IEEE Computer Society, Lübeck, Germany, 2007, pp.420-423.

[15] M. Sveda, O. Ryšavý, and R. Vrba, "Pattern-driven Reuse of Behavioral Specifications in Embedded Control System Design", Frontiers in Robotics, Automation and Control, Vienna, AT, IN-TECH, 2008, pp.151-164.