A Methodology Based on Model-Driven Engineering for IoT Application Development

Claudia Maricela Sosa-Reyna, Edgar Tello-Leal, David Lara-Alabazares, Jonathan Alfonso Mata-Torres, Esmeralda Lopez-Garza

Reynosa-Rodhe Multidisciplinary Academic Unit - Faculty of Engineering and Science
Autonomous University of Tamaulipas
Victoria, Mexico
e-mail: clauqueen1@gmail.com, etello@uat.edu.mx, dlara@uat.edu.mx, mata.jona@gmail.com, elgarza@uat.edu.mx

***Abstract*—The Internet of Things can be understood as an infrastructure of the dynamic global network with a capacity of self-configuration, based on standard communication protocols, where things -physical and virtual- have an identity, physical attributes, and virtual personalities. In this paper, we propose a methodology based on Model-Driven Engineering with different levels of abstraction, points of view, and granularity, with the objective of guiding the development of software applications for Internet of Things. The methodology is supported by methods of model transformation, enabling the generation of the code of the software applications for Internet of Things. In addition, a Service-Oriented Architecture is presented for the deployment of software applications, composed of four layers that allow the identification of the components required for the implementation of the Internet of Things systems.**

Keywords-thing; IoT; MDE; SOA; model; transformation.

I. INTRODUCTION

Nowadays, the advances in research and development of the Information and Communications Technology (ICT) encourages to try to connect all things or objects to the world via the Internet, in order to provide an integrated system to improve their performance in the transmission of information, and offer new services over the Internet. This converts an object into a smart object, allowing to control any tangible object remotely, and has been called the Internet of Things (IoT) [1]. IoT consists of a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies [2].

In the IoT paradigm, a thing is an object of the physical world (physical thing) or the information world (virtual thing), which is capable of being identified and integrated into communication networks [2]. Hence, IoT represents a significant extension of the Internet with a large number of physical objects and devices that show pervasive sensing, detection, actuation, and computational capabilities. IoT is usually related to a physical thing or object, but a fundamental component in IoT is the software used to connect and manage things and as well as to analyze their collected data. This software can be found in different locations, such as embedded, or at the level of middleware, applications, logic in the composition of services, or management tools [3][4].

The advances achieved in automation and intelligent environments have reached important levels, and some contributions in this sense through IoT concepts include smart cities, smart homes, e-health, smart grids, intelligent transportation systems, and intelligent use of water, among others [5]-[9]. While these paradigms have their advantages, their development is complex; that is why new alternatives to facilitate generation processes and implementation of innovative applications are sought. Moreover, the principal characteristic of IoT systems is that heterogeneity between its components, in where things might be totally different among themselves in terms of both hardware and software. However, the very same software functionalities are expected to be deployable on different devices having only a limited set of core common features [10]. Therefore, having models that conceptualize the domain of a specific problem, and with which the elements that integrate it can be identified, classified and abstracted, represents the possibility of achieving efficient automated implementation, as well as support the complexity of the heterogeneous things.

In this respect, Model-Driven Engineering (MDE) is based on models that in an early stage in the development minimize the technological aspects, so that communication between users, analysts, and system developers, can be more efficient, allowing the selection of the technological platform until the end of the process. With the use of MDE, the automatic generation of the code as the product of a set of model transformations represents an increase in productivity, favoring consistency through automation. In the MDE approach, high-level abstraction models are transformed into lower-level models, where the relationship between both models results in a dependence that keeps the process that has been followed until a technological solution, helping to understand the consequences of the changes at any point in the development process [11]. When developing systems based on models, it is possible to achieve an easy adaptation to the changes, both technological and the business requirements that may appear in the development process, turning the models into reusable and enduring units. As part of the development process, where models are productive units from which automated implementations emanate, we find as core points: 1) the abstraction, represented by high-level modeling languages; 2) the automation, that allows to

 ISBN: 978-1-61208-615-6

transform the models in computer programs; and 3) the standard, or complementary development tools; with the aim of obtaining formal models or software artifacts that can be understood by a computer [12].

Moreover, considering the ubiquity and particularity existing in intelligent environments in IoT systems, MDE allows the management of heterogeneous technologies through automatic transformation methods and generation of code for specific platforms. In MDE, the transformation of models can be vertical, where it refines abstract models in more specific models, or horizontal form, defining mappings between models of the same level of abstraction, and in this way to identify the best solution.

Furthermore, in the Service-Oriented Architecture (SOA) approach, a complex system is treated as a set of well-defined objects or subsystems [13]. These subsystems can be reused, maintaining their individual form. Hence, software and hardware components in an IoT architecture, implemented with SOA, can be efficiently reused and updated. Therefore, when SOA is applied in IoT, the generated design can provide extensibility, scalability, modularity, and interoperability between heterogeneous things, as well as the functionalities and capabilities that were encapsulated in a set of services.

In this paper, we propose a methodology supported by the MDE for solving the challenges in the IoT system developments. The methodology is composed of four phases with different levels of abstraction, viewpoint, and granularity. The phases of the methodology are supported by methods of transformation of models based on MDE. In addition, an architecture for IoT systems composed of four layers is presented, which is based on the SOA approach. This architecture allows the interoperability between heterogeneous devices to be guaranteed in multiple ways, establishing a bridge between the digital and physical world of IoT. Therefore, the architecture and methodology enable guides the process of development of software applications oriented to services, which make it possible to satisfy the business requirements of the IoT domain.

The reminder of the paper is organized as follows. In Section II there is a review of the related research which deals with the technology of Internet of Things and MDE and their integration. Section III introduces the architecture based on SOA. In Section IV, we propose the MDE methodology for Internet of Things, together with conclusions, in Section V.

II. Related Work

The development of technological solutions for the specification of software systems for IoT using the principles of MDD has been previously studied. The most relevant proposals that are related to the approach proposed in this research work are discussed in this Section. Nguyen et al. [14] proposed a Framework for Sensor Application Development (FRASAD) based on MDD approach, which aims at improving the re-usability, flexibility, and maintainability of sensor software. In the highest abstraction level of your architecture, a rule-based model and a Domain Specific Language (DSL) are used to describe the applications. It has been elaborated to uncouple the programming language and their execution model from the underlying OS and hardware. The DSL model has been extended for support to the different operating systems such as TinyOS or Contiki.

Pramudianto et al. [15] presented a MDD approach for the development of software components of the IoT domain, using three levels of abstraction. The Platform Specific Model (PSM) can be transformed into a Java code, which requires refinement before implementation. The code generated run as a standalone application that exposes the domain objects through different protocols and serialization formats. Similarly, Conzon et al. [16] proposed a tool using MDD for extending a platform to be used in factory automation and on techniques used for energy consumption optimization and CO2 reduction. This MDD tool allows developers to discover and compose distributed devices and services into mashups, enabling developers to model the integration of IoT components visually and programmatically, transforming the model into the software code.

In [17], Einarsson et al. presented a Domain-Specific Modelling Language (SmartHomeML) for smart home applications, incorporating a metamodel that enables the capture the architecture and specifications of smart home devices, as well as two transformation templates that generate code from instances of SmartHomeML for Alexa and SmartThings. This transformation was designed trough MDD approach, using a model-to-text transformation in a platform-specific model level to code (implementation artifact). Brambilla et al. [18] proposed a model-driven development based approach for the definition of user interface components and design patterns specific to the IoT domain. In addition, a proposed an extension of the standard Interaction Flow Modeling Language (IFML) in order to support the implementation of the user interfaces. The IFML extension designed focuses on mobile applications, enable for expressing the content, user interaction, and control behavior of the front-end of IoT applications.

In our work, we propose a methodology that explicitly describes the phases for the design of software applications for the IoT domain. In this proposal, the abstraction levels and granularity (of each source or target model required) of the phases and stages of the methodology are described in detail, as well as the model transformation methods (based on MDD) that allow giving support each of the phases of the methodology.

III. SOA-based Architecture for IoT Systems

The main requirement of IoT is that things in the network must be interconnected. The architecture of an IoT system must guarantee the operations of things, allowing a bridge between things (physical part) and the virtual world of IoT. The SOA-based architecture for the development of proposed IoT systems is composed of 4 layers, as shown in Figure 1.

 ISBN: 978-1-61208-615-6

Figure 1. Architecture based on SOA for IoT systems.

A. *Object Layer*

The Object layer is composed of hardware objects available on the network that detect the state of things. In the object layer, the intelligent systems through labels or sensors, are able to automatically detect the environment and perform data exchange between devices. The objects in this layer must have a digital identity (universal identifier, UUID), allowing to trace the object in the digital domain, making it possible to meet IoT's expectation of being a physical interconnected network all over the world, where things are seamlessly connected and can be controlled remotely [13].

B. *Network Layer*

The Network layer consists of the infrastructure that supports wired, wireless or mobile connections between things, allowing to detect their environment, which enables to share data between connected things, enabling event management and intelligent IoT processing. The network layer enables to manage the communication in the IoT environment and to transmit messages between the objects and systems. In the SOA approach, services will be consumed by things that have been enabled in the network layer. The network layer is crucial in any IoT approach, considering QoS functionalities, efficient energy management in the network and in things, signal and data processing, security and privacy, among others.

C. *Service Layer*

In the Service layer, there are created and managed the services required by the users or software applications. The service layer is based on middleware technologies, which is fundamental for consuming services and the execution of IoT applications, where hardware and software platforms can be re-usable. Middleware plays a key role in supporting the development of such IoT enhanced applications and services. The IoT systems introduce significant new challenges for middleware stemming from the vast number of connected objects, the volume and variety of the data produced, the patterns of communication required, the heterogeneity of communicating components, and new challenges in terms of quality-of-service, privacy, and security [7]. It is one of the architecture's critical layers of operation, which operates in bidirectional mode. This layer operates as an interface between the object layer (at the bottom of the architecture), and the application layer (at the top of the architecture). It is responsible for functions such as device management, information management, data filtering, data aggregation, semantic analysis, and information discovery [19]. The services layer consists of: service discovery, service composition, APIs, and reliability management (see Figure 2), among others. The discovery of services allows to find the objects that can provide the required service and the necessary information in an efficient way, through the UUID in the registry of services or repository of services. The composition of services allows the interaction between connected things by combining the available services to perform a specific task, that is, when the services are created and stored in the service repository, they can be combine in services of higher level of complexity from the business logic.

D. *Application Layer*

The Application layer is responsible for delivering the applications to different IoT users. The application layer usually plays the role of providing services or applications that integrate or analyze the information received from the other three layers. The intent of the architecture is to support vertical applications. The development of applications in IoT has focused on the areas of health, agriculture, transportation, intelligent cities, home automation, complex systems for decision making, water use management, etc. [4][5][8][9].

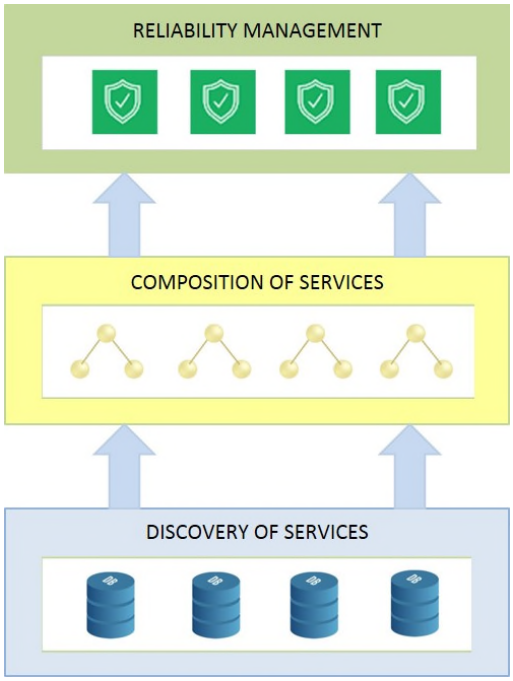

Figure 2. Scheme of the SOA service layer.

 ISBN: 978-1-61208-615-6

IV. MDE METHODOLOGY FOR IOT APPLICATIONS

The proposed methodology is oriented to the use of conceptual models in different points of view, levels of abstraction, and granularity. The output artifacts of the phases of this methodology are represented by models, generated by the application of the principles of MDE. The end result are software implementation artifacts, that is, the code of the applications or software systems for IoT. In Figure 3, the phases that make up this methodology are presented: 1) analysis of business requirements, 2) definition of the business logic, 3) design of the integrated services solution, and 4) generation of the technological solution.

A. Phase 1. Analysis of business requirements.

In this phase, the problem domain is analyzed and the business requirements are identified. This is done considering the functional and non-functional requirements of the system. In order to define the business requirements model, the UML language is used, capturing the flow of the software process through use case diagrams and activity diagrams, generating a model defined in a Platform Independent Model (PIM) level. A PIM is a platform-independent system view, that is, a model with a high level of abstraction independent of any technology or implementation language that exhibits a sufficient degree of platform independence to allow mapping to one or more platforms.

B. Phase 2. Definition of the business logic.

This phase focuses on the design of the business processes required to support the business logic and business requirements. Then, the pre-generated business requirements model is used as input to the phase, and is complemented by the definition of the business process logic, using the Business Process Model and Notation language (BPMN), which allows to generate a model of the business solution defined in a PIM level of the MDE, describing the behavior and interactions of the business process from a global viewpoint.

C. Phase 3. Design of the integrated services solution.

In this phase, a model of the IT architecture is defined, at a platform independent level to separate the business logic solution from the technical aspects of implementation (IT), which allows that this type of implementation can be generated on different target platforms. The IT architecture model is derived from the model of the business solution generated in the previous phase, the generated model remains unchanged on any platform. In this case, the IT architecture model is generated following the approach oriented to SOA services. The output of this stage is a model defined in a PIM level.

D. Phase 4. Generation of the technological solution.

In this phase, specific concepts of the implementation platform are used in order to convert this solution into an executable code of a particular software application. This phase is accomplished through the implementation of two stages: 1) design of the IT platform specific solution, and 2) generation of the specifications or code of the software system. The first stage consists of the definition of the specifications model based on a standard or specific technology (for example, TinyOS 2.0 or WSN Operating Systems), using as input to the phase, the IT architecture model, previously generated. The output of this stage is a model defined in a Platform-Specific Model (PSM) level. A PSM presents a view of the system from the perspective of a specific technological platform, that is, an associated solution model to a platform that includes the details of the PIM and describes how the implementation is performed on that platform. The technology solution model contains the information required for the specific platform (specific messages in the send or receive format for things or objects, transport protocols used, sensor UUID, sender or receiver). The second stage consists of a transformation of the PSM to text, which represents the code skeleton or executable code of an application, usually in XML-based specifications.

E. Model transformation methods.

The methodology is supported by methods based on the MDE approach, to reduce costs and development time, allowing automatic and semi-automated model transformations to generate the output models of each phase. A model transformation consists of a set of transformation rules, which define how an input model is mapped to one or more models or executable code. In order to support the necessary model transformations for the methodology, it is proposed the application of different model transformation methods.

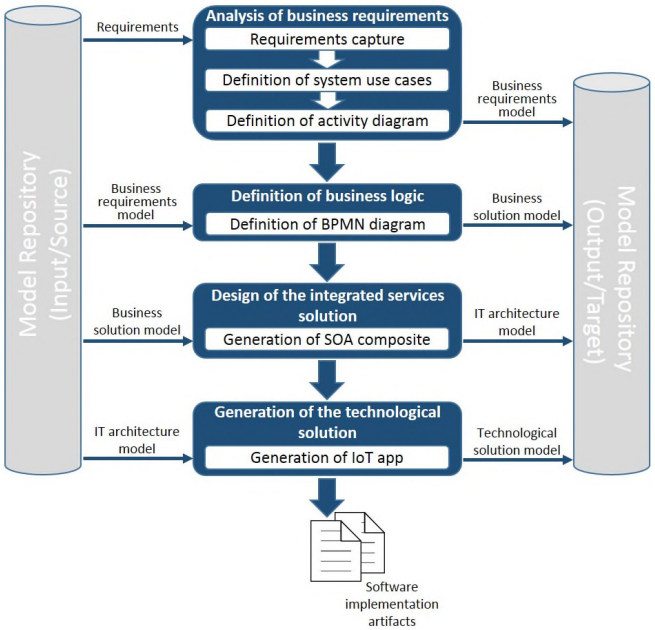

Figure 3. MDE based methodology to IoT applications deployment.

The proposed MDE methods to generate the technological solutions in IoT environments, through software applications oriented to services, consist of 4

transformations. Through the T1 transformation, a conceptual model of the business solution is generated based on a business requirements model, complemented with the business logic and the business process designed through a horizontal PIM-to-PIM transformation. This transformation is derived from the business solution model, using the concepts of the Service Oriented Architecture (SOA), maintaining an independence of the implementation platform. Figure 4 shows an example of a transformation rule (T2), where the BPMN language meta-model is used as input and the SOA architecture meta-model is used as a destination (Rule 1). The transformation methods and proposed rules were defined using the Eclipse Atlas Transformation Language (ATL).

```
rule participant2scope {
      from
            participant: MMbpmn2!Participant (
                  not
participant.processRef.oclIsUndefined()
            )
      to
            scope: MMUML4SOA!Receive (
                  lnk <- participant.name,
                  rvc <- participant.document
            )
}
```

Rule 1

```
rule scope2vehicle {
      from
            participant: MMUML4SOA!Recieve (
                  not
recieve.processRef.oclIsUndefined()
            )
      to
            vehicle: MMSmartVehicle!sensor (
                  sensorData <- receive.lnk,
                  command <- receive.rvc
            )
}
```

Rule 2

Figure 4. Example of transformation rules for method T2 (Rule 1) and T3 (Rule 2).

The technological solution is generated by two model-driven methods. The first method applies a model-to-model transformation T3, generating an output model based on the specific implementation platform that is selected, using a model of the IT architecture as input. The generated model is defined at a specific level of the PSM platform, using concepts from the IoT implementation platform. Figure 4 shows an example of a transformation rule (Rule 2) of the SOA meta-model (PIM level) to a meta-model at a PSM level. In the example, the data to be sent to a sensor of a software application of a smart vehicle simulator is generated. The second method is done by the direct model-to-text transformation T4, which consists of the generation of a document with the source code that represents the structure and behavior of the object when an event occurs.

V. CONCLUSIONS

In this paper, a methodology was presented for the development of software applications for IoT. The methodology is based on the principles of Model-Driven Engineering (MDE), where a set of model transformation methods were defined and specified with different viewpoints, abstraction levels, and granularity. The proposed methodology allows guiding the process of developing software applications oriented to services, from conceptual models to the code of a specific application and a selected technology platform.

The objective of the proposed approach is to reduce the time and costs in software development by implementing automatic and semi-automatic model transformations. In addition, an architecture to support the applications or software systems for IoT was proposed. The architecture describes, in a generic way, the different layers required for the deployment of software applications in IoT, using the concepts of Service-Oriented Architecture (SOA).

ACKNOWLEDGMENT

This work was supported by the National Council of Science and Technology (CONACYT) of Mexico under Grant 256922.

REFERENCES

[1] C.-W. Tsai, C.-F. Lai, and A. V. Vasilakos, “Future internet of things: open issues and challenges,” Wireless Networks, vol. 20, no. 8, pp. 2201–2217, 2014.

[2] ITU-T, “Overview of the internet of things,” Telecommunication Standardization Sector of ITU, Specification ITU-T Y.2060, January 2013. [Online]. Available: http://handle.itu.int/11.1002/1000/11559

[3] L. Atzori, A. Iera, and G. Morabito, “The internet of things: A survey,” Comput. Netw., vol. 54, no. 15, pp. 2787–2805, 2010.

[4] S. Stastny, B. A. Farshchian, and T. Vilarinho, “Designing an application store for the Internet of Things: Requirements and challenges,” Proc. Ambient Intelligence: 12th European Conference (AmI 2015), Springer International Publishing, Nov 2015, pp. 313–327, doi:10.1007/978-3-319-26005-1_21

[5] C. Yin, Z. Xiong, H. Chen, J. Wang, D. Cooper, and B. David, “A literature survey on smart cities,” Science China Information Sciences, vol. 58, no. 10, Oct 2015, pp. 1–18. [Online]. Available: https://doi.org/10.1007/s11432-015-5397-4

[6] E. Al Nuaimi, H. Al Neyadi, N. Mohamed, and J. Al-Jaroodi, “Applications of big data to smart cities,” Journal of Internet Services and Applications, vol. 6, no. 1, Dec 2015, pp. 1-15. [Online]. Available: https://doi.org/10.1186/s13174-015-0041-5

[7] G. Blair, D. Schmidt, and C. Taconet, “Middleware for internet distribution in the context of cloud computing and the internet of things,” Annals of Telecommunications, vol. 71, no. 3, pp. 87–92, 2016. [Online]. Available: https://doi.org/10.1007/s12243-016-0493-z

[8] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of things (IoT): A vision, architectural elements, and future directions,” Future Generation Computer Systems, vol. 29, no. 7, 2013, pp. 1645-1660. [Online].Available:https://doi.org/10.1016/j.future.2013.01.010

[9] B. d. T. Pereira, L. C. Melo, F. J. da Silva, L. E. Talavera, and M. Endler, “A comprehensive and scalable middleware for ambient assisted living based on cloud computing and internet of things,” Concurrency and Computation: Practice and Experience, vol. 29, no. 11, 2017, pp. 1-19. [Online]. Available: http://dx.doi.org/10.1002/cpe.4043

[10] F. Ciccozzi, and R. Spalazzese, “MDE4IoT: Supporting the Internet of Things with Model-Driven Engineering,” Proc. 10th International Symposium on Intelligent Distributed Computing (IDC 2016), Springer International Publishing, October 2016, pp. 67–76, doi: 10.1007/978-3-319-48829-5_7

[11] E. Tello-Leal, A. B. Rios-Alvarado, I. Lopez-Arevalo, O. Chiotti, and P. D. Villarreal, Methodology based on Model-Driven Development for the execution of collaborative processes through software agents, 1st ed. México, Ediciones UAT - Plaza y Valdes, 2016.

[12] C. Pons, R. Giandini, and G. Perez, Model-Driven Software Development, 1st ed. Argentina, Editorial Universidad Nacional La Plata - McGraw Hill, 2010.

[13] L. Shancang, X. L. Da, and Z. Shanshan, “The internet of things: a survey,” Information Systems Frontiers, vol. 17, no. 2, pp. 243–259, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10796-014-9492-7

[14] X. T. Nguyen, H. T. Tran, H. Baraki, and K. Geihs, “FRASAD: A Framework for Model-Driven IoT Application Development,” Proc. IEEE 2nd World Forum on Internet of Things (WF-IoT 2015), 2015, pp. 387–392. Available: doi:10.1109/WF-IoT.2015.7389085

 ISBN: 978-1-61208-615-6

[15] F. Pramudianto, C. A. Kamienski, E. Souto, F. Borelli, L. L. Gomes, D. Sadok, and M. Jarke, "IoT Link: An Internet of Things prototyping toolkit," Proc. 2014 IEEE 11th Intl Conf on Ubiquitous Intelligence and Computing, and IEEE 11th Intl Conf on Autonomic and Trusted Computing, and IEEE 14th Intl Conf on Scalable Computing and Communications (UTC-ATC-ScalCom), 2014, pp. 1–9. [Online]. Available: doi:10.1109/UIC-ATC-ScalCom.2014.95

[16] D. Conzon, P. Brizzi, P. Kasinathan, C. Pastrone, F. Pramudianto, and P. Cultrona, "Industrial application development exploiting IoT vision and Model-Driven programming," Proc. 18th International Conference on Intelligence in Next Generation Networks (ICIN 2015), 2015, pp. 168–175. [Online]. Available: doi:10.1109/ICIN.2015.7073828

[17] A. F. Einarsson, P. Patreksson, M. Hamdaqa, and A. Hamou-Lhadj, "SmarthomeML: Towards a domain-specific modeling language for creating smart home applications," Proc. IEEE International Congress on Internet of Things (ICIOT 2017), June 2017, pp. 82–88. [Online]. Available: doi:10.1109/IEEE.ICIOT.2017.35

[18] M. Brambilla, E. Umuhoza, and R. Acerbis, "Model-Driven Development of user interfaces for IoT systems via domain-specific components and patterns," Journal of Internet Services and Applications, vol. 8, no. 1, pp. 1-21, 2017. [Online]. Available: https://doi.org/10.1186/s13174-017-0064-1

[19] D. Bandyopadhyay and J. Sen, "Internet of things: Applications and challenges in technology and standardization," Wireless Personal Communications, vol. 58, no. 1, pp. 49–69, 2011. [Online]. Available: https://doi.org/10.1007/s11277-011-0288-5

 ISBN: 978-1-61208-615-6