PAPER

QSL: A Specification Language for E-Questionnaire, E-Testing, and E-Voting Systems

Yuan ZHOU[†a)], ***Nonmember***, **Yuichi GOTO**[†], ***Member***, ***and*** **Jingde CHENG**[†,††], ***Nonmember***

SUMMARY Many kinds of questionnaires, testing, and voting are performed in some completely electronic ways to do questions and answers on the Internet as Web applications, i.e. e-questionnaire systems, e-testing systems, and e-voting systems. Because there is no unified communication tool among the stakeholders of e-questionnaire, e-testing, and e-voting systems, until now, all the e-questionnaire, e-testing, and e-voting systems are designed, developed, used, and maintained in various ad hoc ways. As a result, the stakeholders are difficult to communicate to implement the systems, because there is neither an exhaustive requirement list to have a grasp of the overall e-questionnaire, e-testing, and e-voting systems nor a standardized terminology for these systems to avoid ambiguity. A general-purpose specification language to provide a unified description way for specifying various e-questionnaire, e-testing, and e-voting systems can solve the problems such that the stakeholders can refer to and use the complete requirements and standardized terminology for better communications, and can easily and unambiguously specify all the requirements of systems and services of e-questionnaire, e-testing, and e-voting, even can implement the systems. In this paper, we propose the first specification language, named "QSL," with a standardized, consistent, and exhaustive list of requirements for specifying various e-questionnaire, e-testing, and e-voting systems such that the specifications can be used as the precondition of automatically generating e-questionnaire, e-testing, and e-voting systems. The paper presents our design addressing that QSL can specify all the requirements of various e-questionnaire, e-testing, and e-voting systems in a structured way, evaluates its effectiveness, performs real applications using QSL in case of e-questionnaire, e-testing, and e-voting systems, and shows various QSL applications for providing convenient QSL services to stakeholders.

key words: *specification language, unified communication tool, e-questionnaire system, e-testing system, e-voting system*

1. Introduction

Questionnaire and voting are the essential activities of the modern communities as the general and indispensable methods for a group of people to express a choice, a preference, or an opinion [1]. Testing also is a necessary activity as an integral way to be widely used to assess people's achievement, ability, and characteristics [2]. Over two decades, many kinds of questionnaires, testing, and voting are performed in some completely electronic ways to do questions and answers on the Internet as Web applications, i.e. e-questionnaire systems, e-testing systems, and e-voting systems (QTV systems for short). QTV systems play an important role in modern society, have a unique value and are indispensable in society. If these systems are unreliable, lower security, strange in use, it will have a serious impact on society. There is still an important research topic of how to design, develop, maintain, and operate reliable, highly secure, and user-friendly QTV systems.

On the one hand, because e-questionnaire, e-testing, and e-voting services (QTV services for short) have common processes, that is, from preparing questions, following by authenticating respondents, through submitting answers, and ending to analyzing, tallying, and declaring results, systems that provide QTV services have common functions to do the processes. In fact, some systems [3]–[12] exist to provide three-in-one service for people all over the world.

On the other hand, the mutual collaboration of the stakeholders is the foundation for development and operation of the information systems [13]. There are 5 kinds of the stakeholders of QTV systems, which are sponsor, evaluator, executor, respondent, and supporter. Firstly, a sponsor launches activities and usually focuses on the strategic goals, return on investment, as well as the costs and time involved in building and operating the systems. Secondly, an evaluator monitors whether the system meets standards, laws, and regulations. Thirdly, an executor performs tasks to help carry out the activities after system deployment that perhaps contains a questioner, a monitor, and an analyst, etc. Fourthly, a respondent is to answer a questionnaire, a test, or a vote. Fifthly, a supporter provides the tech-support services that perhaps contains including a communicator, a developer, a maintainer, a manufacturing engineer, a supplier, and a customer service, etc. From the respective of software engineering, the most important goal of the implementation of QTV systems is to satisfy the requirements of the stakeholders. There are many stakeholders involved in a variety of QTV systems around the world. To address this situation, it is necessary to promote the effective communications among those stakeholders by a communication tool to easily link each stakeholder's requirements for systems and services of QTV.

Because there is no communication tool shared among the stakeholders of systems and services of QTV, all the QTV systems are designed, developed, used, and maintained in various ad hoc ways. As a specific example on communication among the orderer who plays the role of the sponsor to order a new system and the developer who is a role of supporter to develop the system.

Manuscript received September 30, 2018.
Manuscript revised June 1, 2019.
Manuscript publicized August 19, 2019.

[†]The authors are with the Department of Information and Computer Sciences, Saitama University, Saitama-shi, 338–8570 Japan.
[††]The author is with Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China.
a) E-mail: shuugen@aise.ics.saitama-u.ac.jp

DOI: 10.1587/transinf.2018EDP7333

Firstly, it is not so easy for the orderer to define exhaustive service requirements and for the developer to define exhaustive system requirements because the developer is not the professional expert in the business fields, and the orderer is not a technical specialist. They only possess knowledge of their own requirements but have a shallow understanding of the overall requirements. It lacks an exhaustive requirement list to guide and assist them to know overall requirements and seek a common understanding for these systems and services of QTV. Secondly, it is difficult to avoid ambiguity for them because they do not have the standardized terminology on the systems and the services of QTV. The orderer could use different or vague terminology for defining a common service requirement. It may lead the developer misinterpreted to provide a different and even incorrect solution for the corresponding system. It lacks a standardized terminology to unify the common requirements and to normalize the different requirements for QTV services and the corresponding systems. Not only the above-mentioned specific example but also others exist similar communication problems among other kinds of stakeholders that are easy to misunderstand.

A specification language [14] is a formal language in computer science used during systems analysis, requirement analysis, and system design to describe a system. From the viewpoint of software engineering, requirements specification is the first and main step of developing a software, and the specification language is the most direct way to specify requirement specifications, after all, the requirement specification is the result by communicating among the stakeholders. In addition, owing to the demands for exhaustive requirement list and the terminology, a specification language is a better communication tool to solve communication problems.

Thus, it is necessary to provide a specification language as a communication tool to provide exhaustive requirement list and standardized terminology for systems and services of QTV so that the stakeholders can communicate easily and unambiguously, and deal with and describe the requirement specifications for three kinds of systems and services. Figure 1 shows the relationship between the specification language and the stakeholders. The specification language allows the sponsor, the executor, the evaluator, and even the respondent to have an overall consciousness owing to the exhaustive requirement list, and then to use it to easily communicate with each other as well specify the requirement specifications for three kinds of services. Because of the standardized terminology to avoid ambiguity, they can communicate clearly with the supporter. Referring to the service requirement specifications, the supporter can specify the system requirement specification based on the specification language and even implement the system. Meanwhile, it is convenient that the specification language provides the sponsor, the executor, and the respondent with data by a unique format that can be reused. Besides, when the stakeholders are using and working with the updating system, which needs to add items, change functions, and add constraints, they are looking forward to a clever and automatic way to easily deal with the changes. The ideal state is that a tool as a generator converting the formalized specifications to automatically generate QTV systems.

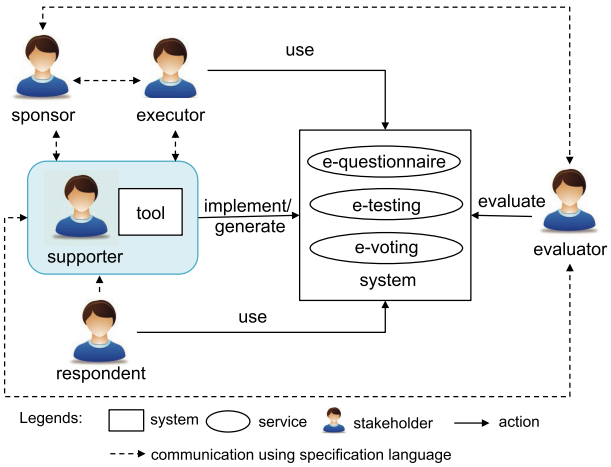

Fig. 1 Relationships of specification language and stakeholders.

However, the existing specification languages Simple Survey System (SSS) [15], IMS Question and Test Interoperability Specification (QTI) [16], and OASIS Election Markup Language (EML) [17] cannot solve the above problems, because based on their own motivations, they can neither cover QTV systems nor specify exhaustive requirements of QTV systems. And that means, until now, there is no general-purpose specification language can provide precise and essential description ways for specifying various QTV systems in a unified format such that the stakeholders can communicate, understand, and accept each other to create the specifications of these essentially similar systems.

This paper proposes QSL [18]–[23], the first specification language for QTV systems that serves as a unified communication tool for specifying various QTV systems with a standardized, consistent, and exhaustive list of requirements. As the advantages of QSL, firstly, QSL can specify various systems and services of QTV. It satisfies public demands to unify QTV systems, and promotes the communications of stakeholders among these essentially similar systems. Secondly, QSL can be used as a unified format for portable data among different QTV systems. It contributes to improve data portability, which right is very important for our modern society and accepted by General Data Protection Regulation [24] since 2016 for providing conveniences for the stakeholders to reuse data. Thirdly, QSL can be used for the precondition to automatically generate QTV systems. It is conducive to improving efficiency by automating and streamlining stakeholders' work procedures.

The rest of the paper is organized as follows: Sect. 2 presents an analysis of exhaustive requirements of QTV systems. Section 3 shows the QSL definitions addressing that it can specify all the requirements of various QTV systems in a structured way for satisfying its stability and extensibility. Section 4 presents the evaluation about the descrip-

tive power of QSL manifesting in specifying various QTV systems. In Sect. 5, we present two real applications of QSL in case of QTV systems we implemented. Various QSL applications for providing convenient QSL services are presented in Sect. 6. Finally, some concluding remarks are given in Sect. 7.

2. Exhaustive Requirements of QTV Systems

2.1 Collecting Requirements of QTV Systems

QSL is proposed to provide exhaustive requirements list of QTV systems to the stakeholders so that they can design, develop, and evaluate their demanded systems easily. In order to get exhaustive requirements of QTV systems, at first, we investigated 29 e-questionnaire systems [3]–[12], [25]–[43], 22 e-testing systems [3]–[12], [36], [40], [44]–[53], and 24 e-voting systems [3]–[12], [54]–[67], which are representative systems seizing a large number of high quality customers all over the world for serving a relatively long time. Most of the investigated systems can optionally execute anonymous, roll-call, open, and closed questionnaires, exams, and votes by switching functions. We also enumerated and summarized the requirements from these systems.

Afterwards, we found QTV systems have some common and specific requirements. Figure 2 illustrates a Venne diagram to present the relationship of QTV systems. The three primary color circles stand for the sets of requirements of QTV systems, respectively. The union of four numbered regions in a circle presents a complete system. For instance, a set of requirements of a complete e-questionnaire system is a union of requirements in region 1, 2, 4, and 5.

Region 1 lists the intersection of the sets of requirements of QTV systems, i.e., a set of common requirements of QTV systems. The common requirements are core contents to construct a general-purpose QTV system. The common requirements of QTV systems are summarized for two aspects: system and service. Firstly, there are 55 common requirements for the QTV systems. All these three kinds of systems have common phases, which are *setting-up* to prepare QTV; *distributing* to distribute the paper to respondents; *registering* to allow anyone through some authentication check to be an eligible respondent; *submitting* to answer the questions in the paper and send to submitting server, usually called voting phase in e-voting; *collecting* to collect the responses of the respondents; *analyzing* to analyze the collected responses; and *counting* to calculate the analyzed

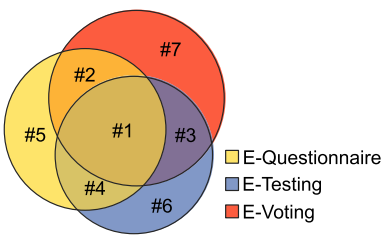

Fig. 2 Relationship of sets of requirements of QTV systems.

responses and get results, usually called tallying phase in e-voting [68]. Based on the phases, they have common security requirements such as *authority* to ensure the access control of the participants. It is worth mentioning at this point that *authentication*, as a necessary security requirement, is used to confirm the participants to whether have the authority to do something like register or login the system. The system can allow anyone to use it without any authentication, and there is another situation required a particular user to use it before being authenticated. For the authentication, there are 3 kinds of basic methods, which are *secrecy* to authenticate what they know such as a password, *token* to authenticate what they have such as ID, and *biometrics* to authenticate what they are such as fingerprints. These methods can be combined to get complex methods for e-voting. They also have common functional requirements such as *launching the paper* or *stopping launching* during setting up phase, as well as environmental requirements such as *server* to store the collected results after submitting phase and server to provide registration services for respondents in registering phase. Secondly, there are 66 common requirements for the QTV services. These three kinds of services should provide common questionnaire structure, which is a question sheet on a form to express a choice preference, usually called ballot in e-voting. It consists of section, question, answer, and some pictures and videos to represent it well recorded as *media*, etc. There are some common requirements for question types such as multiple choices, validation common requirements to limit the responses and control the required response, etc., and common setting requirements such as *language* to support the representation of multilanguage. The services are inseparable from the common roles of the participants, which are *sponsor* who organizes and supports an event; *questioner* who designs a paper and settings, usually called *examiner* in e-testing; *analyst* who processes the collected responses; *monitor* who monitors whether illegal or dishonest behavior occurs or not; *respondent* who answers the questions in the paper, usually called *examinee* in e-testing and *voter* in e-voting [2]; and *analyst* who analyze the collected responses. In addition, these services have common requirements for recording information such as responses from respondents, and analyzed reports.

Region 2, region 3, and region 4 list the intersections of two systems. In region 2, both e-questionnaire and e-voting systems have same question types such as *contact information* to record the name, address, fax, phone number, etc. It is an advanced question type of open-ended text, which derives from single row text to present in multiple lines. They both have same question type named *data reference* to collect or validate such as zip code data against "standardized" database. Region 3 lists the similar requirements of e-testing and e-voting systems. They both have similar security requirements considering on ensure fairness of respondents. For example, *respondent authentication by token* is a security requirement in registering phase to provide respondents to register on the system by authenticating his ID information. According to the security requirements,

they have some common environmental requirements such as *security software* to provide a security system and sealing mechanism so that trust can be placed in the seal and hence the sealed data. This implies that the seal should be performed as close to the user submission of the responses as technically possible. E-testing and e-voting systems also have some common functional requirements such as *confirm response* to provide a confirmation to the respondent regarding the status of his paper, at least the information that his response has been successfully stored. As to region 4, both e-questionnaire and e-testing have some common question types such as *image hotspot* to provide hotspots selection in an image to let respondent select, and *matrix multiple choices select many/one answers* to provide a collection of various individual questions together for respondent to select multiple options for a line marked as checkbox or radio button. They both have common logic requirements like *skip logic* to provide respondents to answer relevant, and *point logic* to compute points in real-time and provide the scores for the options, and the branching on points. They both have common setting requirements about analysis types based on different question types, and distribution methods, etc. For instance, *distribution by live URL* is a common setting requirement for distribution method to provide a unique link for a paper, and the questioner can post this link to let respondents to response, but the responses collected will be anonymous. *Text analysis* is a common setting requirement for analysis to set up text categories with some keywords. E-questionnaire and e-testing systems have some common functional requirements such as *internal stop* to allow respondents to internally stop to answer if the questioner has already set to allow them to do that is decided by the frequency of the stopping during the submitting phase.

Region 5, region 6, and region 7 list the specific requirements of QTV systems. In region 5, e-questionnaire differs with others in its complex question types and complex logic types. E-questionnaire has four basic question types, which are multiple choice, open-ended text, matrix, and ranking, as well as two possible combination question types. Each question type corresponds with limitation and arrangements derive widely variety of question types that e-testing and e-voting are unnecessary. For example, *side-by-side matrix* is a specific question type to collect data on 2 dimensions or more dimensions for the same options. It has a representation that is a matrix with dropdown list, radio button, or checkbox in each grid. In addition, e-questionnaire ensures that only relevant questions are displayed to the appropriate respondents according to its complex question types. E-questionnaire has four basic logic types, which are skipping, piping, extraction, and randomization, as well as three kinds of possible combinations of basic logic types. For instance, *looping* is a specific logic type to loop option, which allows to dynamically looping through a set of questions base on the response to a multiple-choice question. E-questionnaire has some specific validation requirements such as *option quota limitation* to limit responses quota for each option of the selections, and *response quota limitation* to limit total responses of the event. Because e-questionnaire is mostly involved in business surveys, it has some specific setting requirements for dealing with complex statistic and analysis method, such as *trend analysis* to provide a look at data over time for a long-running event. And it is necessary to provide *analyst-oriented device and software* for dealing with that. In region 6, *marking* is a phase to mark the responses and give the scores referring to the sample answers. *marker* is a participant role to mark the responses. In addition, after marking, the *score* of each answer and total score of answers are demanded. Besides, questions in e-testing involve wider and more professional field, such as mathematical and chemical formula, etc. The representative question type for e-testing is *connect points* to provide a question type for connecting listed items from different groups. According to marking phase, there are some requirements like *marking server* to get the response data, identify and authenticate markers, mark the responses and give the score, store the marked data, and send it to counting before deadline, and *blink mark response* to blink mark responses, usually mark for text question type. As to region 7, the differences are around security, because e-voting is mainly used in governmental elections for the universal, equal, free, and secret suffrage. For example, a security requirement *separation of duty for anonymity* to provide a separation of duty approach working with at least two submitting servers, one of which is inspecting the right to submit and another is storing the eligible ballots. E-voting has *auditing* phase to check whether the result is correct. E-voting also needs *option nomination server* to store the candidate information during the setting up phase for option nomination, a *certification server* to validate the respondents to prevent any possibility of affecting results, and an *auditing software* to communicate with submitting system. In addition, e-voting needs a list of *candidates*, and provides a candidate nomination and candidate registration. As to the functional requirements, for example, *store first paper* is a function to store only the first ballot for per respondent in the e-ballot box at the submitting server.

After the investigation, we classified these requirements for services into 8 groups, which are phase, paper, question type, logic, validation, setting, data, and participant. And the requirements for systems are classified into 3 groups, which are environment, function, and security. The groups are the core models to construct a general QTV system. Table 1 presents brief and pithy requirements mapping to the corresponding regions of QTV systems, respectively, as well mapping to the corresponding requirement groups in each region. Considering limited spaces, all the detailed requirements for systems and services of QTV are listed in [23].

According to the investigation, we summarized 181 requirements for e-questionnaire systems, 177 requirements for e-testing systems, and 168 requirements for e-voting systems. These requirements assist to define descriptions and boundaries of each system and service. There are 121 common requirements for QTV systems. These common re-

Table 1 A list of requirements of QTV systems.

Region	Group	Requirements
#1	phase	setting up, distributing, registering, submitting, collecting, analyzing, counting
	paper	section, question, answer, media, etc.
	question type	multiple selections, open-ended text, drop-down list, image chooser, rank order, etc.
	validation	limitation, response required, etc.
	setting	time, language, numbering, etc.
	environment	device, server, software, database, etc.
	participant	sponsor, questioner, respondent, etc.
	data	response, report, participant information
	function	launch, stop launching, distribute, etc.
	security	participant authority, authentication, blind paper for anonymity, etc.
#2	question type	contact information, data reference
#3	environment	security software, mix net, etc.
	function	confirm response
	security	token authentication, etc.
#4	question type	matrix multiple choices, image hotspot
	logic	skip, point logic, automatic redirect, etc.
	setting	question analysis, distribution method, etc.
	function	import paper, export response, etc.
#5	question type	rating, multiple dimensions matrix, side-by-side matrix, etc.
	validation	quota, weighting, etc.
	logic	looping, extraction, complex piping etc.
	setting	trend analysis, balancing, etc.
	environment	analyst-oriented device
#6	phase	marking
	question type	formula, connect point, etc.
	validation	scoring standard
	environment	marking server, marking software, etc.
	participant	marker
	data	score, marker information
	function	mark response
	security	blinded mark
#7	phase	auditing
	environment	option nomination server, database, etc.
	participant	auditor, candidate, proposer
	data	candidate information, no-vote reason, etc.
	function	store first vote, launch proposal, etc.
	security	biometric authentication, holomorphic encrypts for anonymity, etc.

quirements are the core contents to construct a general QTV system, because a set of a basic system for providing QTV services is included in the set of common requirements.

2.2 Terminology

In order to define the terminology for QSL, we extracted the keywords from the summarized requirements, analyzed the keywords, picked up them who have an independent feature in spite of a variety of representations, and defined them as entities. A set of entities is, most tangibly, a set of real objects that cannot be disintegrated, and can be combined in varying amounts describing a gamut of requirements for QTV systems. This is the essential method used to be intended to elicit the descriptions of diverse requirements for QTV systems. The entities are suitable as terminology for QSL.

Phase	Participant	Func.	Env.	Security	Paper	Setting	Data
setting up	sponsor	export	server	authority	paper	language	sample
distributing	questioner	import	gateway	authentication	description	time	score
registering	respondent	launch	interface	anonymity	section	number	response
submitting	analyst	stop	device	seal	question	limitation	result
collecting	monitor	generate	software	channel	answer	quota	field
analyzing	marker	ping			media		
counting	auditor	integrate			logic		
marking	candidate	remind			alignment		
auditing	proposer				formula		

Fig. 3 A form of entities.

There are 55 entities arranged in Fig. 3. The column stands for a group of entities. The groups listed respectively correspond to *phase*, *participant*, *function*, *environment*, *security*, *paper*, *setting*, and *data*.

The first group lists the entities to describe each phase. Marking is a phase in e-testing, and auditing is a phase in e-voting. Phases relate to service's settings and system's functions. In the second group, it lists the entities to describe each role of participants. Both candidate and proposer are the roles of the participant in e-voting. A candidate relates with a number of proposers in an election. The third group shows the entities to describe functions. Some functions are demanded during the whole phases. For example, the system provide functions for a questioner to export the paper file in the whole phases, to import the paper file in setting up phase, to launch and stop the event after setting up phase, and even generate registration tickets to respondents during registering phase, ping IP for monitors before submitting phase, integrate responses and send to analysts during collecting phase, and remind before counting phase for respondent whom did not take part in submitting when they are approaching deadline. The fourth group lists the entities to describe software and hardware for the environment. In the fifth group, it lists the entities to describe security. The sixth group lists the entities to describe the paper sheet. Considering the question type is a feature for question entity, we defined question type as the representation of the question entity. As same as logic type, it is the representation of the logic entity. The seventh group lists the entities to describe settings. As to validation to describe the limitations for the settings, so we defined it as a limitation entity. The last group lists the entities to describe data. The field is data entry in a record for the system database. More to the point, we define these 8 groups of the entities, but there is no conflict with 11 groups of the summarized requirements. The 8 groups are also suitable as terminology for QSL.

2.3 Summary for Exhaustive Requirements

This section elaborates on investigating and analyzing the exhaustive requirements of QTV systems, summarizing the common requirements and specific requirements of QTV systems, and classifying the requirements into 11 groups. The common requirements and the groups are the core to construct a general QTV system. In addition, in this section,

we present extraction of key words as entities for combining an exhaustive requirements list for various QTV systems. We also show the classifications of the entities in 8 groups. These entities and the groups of entities are suitable as terminology for QSL. In general, owing to the relationships of requirements of QTV systems, and independence of the entities, it builds the foundation for design of a specification language of various QTV systems.

3. QSL: A Specification Language for QTV Systems

3.1 Overview

Questionnaire Specification Language (QSL) serves as a formalized specification for specifying various QTV systems. Current QSL is version 3.1 [23]. QSL is based on Extensible Markup Language (XML) [69]. The grammar of QSL is defined by XML Schema [70]. According to summarized exhaustive requirements, using QSL can specify complete specifications for QTV systems, so that can solve the communication problems among the stakeholders.

QSL is an XML-based language that can be used in three ways. Firstly, QSL can be used to describe non-misunderstanding specifications for systems and services of QTV. In other words, QSL can be used to specify the requirements the system provides. For example, QSL provides tags for the server to register the respondents' information, submit and analyze responses to get results. QSL also can be used to describe restrictions in each phase of QTV services. For example, QSL can be used to describe the distribution method such as e-mail, web link, or other offline methods. Secondly, QSL can be used to provide an exhaustive requirement list for systems and services of QTV. To be precise, QSL can be used to describe all the necessary and desirable requirements and the users do not need to list and numerate by themselves. Thirdly, QSL can be used to provide a data format for the data of the expected QTV systems [71]. Portable data [72] is formatted according to a published syntax and where the metadata is explicit, either included with the data or by reference to an open technical dictionary. For instance, QSL is used to describe various questions, responses, and results.

3.2 QSL Construction

We defined **tags** for QSL that are used to describe requirement specifications of QTV systems. Using the tags, we defined **QSL schemas** to constrain the requirements in a formal way. QSL schema is a collection of requirements formalized by XML schema, and clear definition of the relationship among the requirements. The schemas through the XML parser, we get the **QSL templates** that are the requirements formalized by XML, and the requirements correlate with corresponding necessary requirements. The **data format** is a part of QSL schemas, because we defined the data part in QSL schemas.

Figure 4 presents the usages of QSL. The users choose

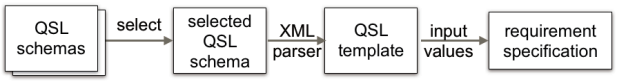

Fig. 4 Usages of QSL.

desirable QSL schema, and through XML parser, they can get the QSL template. The users input appropriate values can get formalized requirement specification.

QSL provides 93 tags as terminology because QSL is an XML-based specification language. We defined two kinds of tags for QSL. Firstly, 55 tags are defined according to entities. Using these tags, we can describe requirements of systems and services of QTV formally. Since XML documents have a hierarchy structure. From the viewpoint of this construction, those tags have some hierarchical relationships. Secondly, 38 tags are defined according to the construction. In addition, QSL provides 52 complex types of the hierarchy of the tags and 34 simple types for the constraint of the tags.

Using the tags, we defined QSL schemas. Because there are nearly 200 requirements for each kind of system and service. We considered that the requirements can be roughly divided into system requirements and service requirements, and both system requirements and service requirements can be are subdivided according to the classification of exhaustive requirements, and then each subdivision of requirements can be divided into common requirements and specific requirements, respectively. Based on the above consideration, we designed the formalized requirements in a hierarchy structure, that are the QSL schemas.

3.3 QSL Hierarchy Structure

QSL hierarchy structure is a regular and core framework indicating and defining the corresponding relationships among QSL schemas. In this structure, each schema provides an intuitive means of requirement navigations for different systems and services for QTV by organizing the corresponding tags in a hierarchical structure. To assist the stakeholders to choose the corresponding schemas, we defined each schema and marked with an exclusive number.

Firstly, 100-series schemas are used to declare QSL document and define system and service requirements. The stakeholder can easily get a requirement list with QSL declaration. Secondly, 200-series schemas are used to define the subdivisions of requirements according to the groups of entities we summarized previously. These schemas help the stakeholders clearly understand what kinds of requirements should be specified. Thirdly, 300-series schemas are used to define common requirements. Fourthly, 400-series, 500-series, and 600-series schemas are used to define specific requirements of systems and services for QTV, respectively. Using the marked numbers, the stakeholders can easily distinguish these three kinds of systems and services. The users can understand the relationships of each schema only according to the numbers. Overall, we defined 65

Table 2 Core schemas configuration.

Code	Schema	Purpose
100	QSL	to declare QSL document
110	System	to specify system template
120	Service	to specify service template
210	Phase	to specify sequence of all the phases
220	Security	to specify security requirements
230	Paper	to specify paper sheet
240	Setting	to specify settings for paper sheet
250	Environment	to specify software, hardware, network
260	Participant	to specify all the participant roles
270	Data	to specify recorded data and DB fields
280	Function	to specify function requirements
310	SettingUp	to specify setting up phase
311	Distributing	to specify distributing phase
312	Registering	to specify registering phase
313	Submitting	to specify submitting phase
314	Collecting	to specify collecting phase
315	Analyzing	to specify analyzing phase
316	Counting	to specify counting phase
320	Sponsor	to specify sponsor information
321	Questioner	to specify questioner information
322	Respondent	to specify respondent information
323	Analyst	to specify analyst information
324	Monitor	to specify monitor information
330	Export	to specify export function
331	Import	to specify import function
332	Launch	to specify launch function
333	Stop	to specify stop launching function
334	Generate	to specify generate function
335	Ping	to specify ping IP function
336	Integrate	to specify integrate function
337	Remind	to specify remind function
340	Server	to specify server information
341	Field	to specify database fields
342	Gateway	to specify network information
343	Interface	to specify interface
344	Device	to specify device information
345	Software	to specify software information
350	Anonymity	to specify anonymous methods
351	Authentication	to specify authenticate methods
352	Authority	to specify authority methods
353	Seal	to specify encryption information
354	Channel	to specify communication channel
360	Section	to specify section for a group of questions
361	Question	to specify question a group of answers
362	Answer	to specify answer contents
363	Description	to specify title, summary, and tips
364	Media	to specify media information for paper
365	Alignment	to specify alignment for paper
366	Limitation	to specify paper limitation
370	Language	to specify multi-languages
371	Time	to specify time settings
372	Number	to specify ordering settings
373	Quota	to specify limitation for response numbers
374	Response	to specify response data
375	Result	to specify result data

schemas in QSL structure. There are 44 schemas (300-series schemas) for specifying common requirements, 10 schemas (400-series, 500-series, and 600-series schemas) for specifying specific requirements, 10 schemas (110, 120, and 200-series schemas) for constructing the hierarchy structure of QSL documents, and 1 schema (100 schema) for QSL document declaration.

Table 3 Specific schemas configuration.

Code	Schema	Purpose
410	Logic	to specify logic methods, conditions, and routes
510	Marking	to specify marking phase
520	Score	to specify score rules
530	Sample	to specify sample answer for each question
540	Formula	to specify formulas of math, chemical, etc.
550	Marker	to specify marker information
610	Auditing	to specify auditing phase
620	Candidate	to specify candidate information
630	Proposer	to specify proposer information
640	Auditor	to specify auditor information

Table 2 shows a list of core schemas. Core schemas marked before reaching 400 are used to describe the requirements for constructing a basic and nondistinctive QTV system.

110 schema is oriented to the supporters and 120 schema is oriented to the executors and the sponsors. The executors and the sponsors choose 120 schema. Through XML parser they can get service template and complete the service requirement specification based on this schema. The supporters choose 110 schema as the structure of system template, and they refer to the service requirement specification to complete the system requirement specification. The 110 schema and 120 schema must declare QSL document that is based on the 100 schema. The remaining schemas are used to construct the 110 schema and 120 schema. In addition, both 110 schema and 120 schema can be used to provide descriptions for data. 110 schema provides the supporters with descriptions for database schema that is defined as 341 schema in QSL. According to 341 schema, the supporters can describe and design database fields. In addition, 270 schema as the child of 120 is used to describe participant information, responses, and results. And it is the most commonly reused data.

Table 3 shows a list of specific schemas. Specific schemas are used to describe the specific requirements of systems and services for QTV. The marked numbers for schemas are used to ensure recognition in which fields, even if the stakeholders were not familiar with the associated business.

When we version up QSL, it is easily revise the schemas without changing the general structure. Because according to the entities and the groups of entities, the schemas defined separately and independently are easy to be extended, revised, and added. The schemas has a stable structure and a well-designed extensibility mechanism.

Figure 5 illustrates the inclusion relations of schemas in QSL hierarchy structure. The black rectangle stands for QSL schemas marked with numbers. The schemas in the gray rectangle are used to specify the data of systems and services of QTV for data portability. 100 schema is used to declare QSL document that must sets a fixed version number. In these schemas, we defined some attributes to coordinate clear descriptions. On account of the repeatability of attribute values, we defined schema for some simpleType elements [70] to specify the constraints for them. This schema

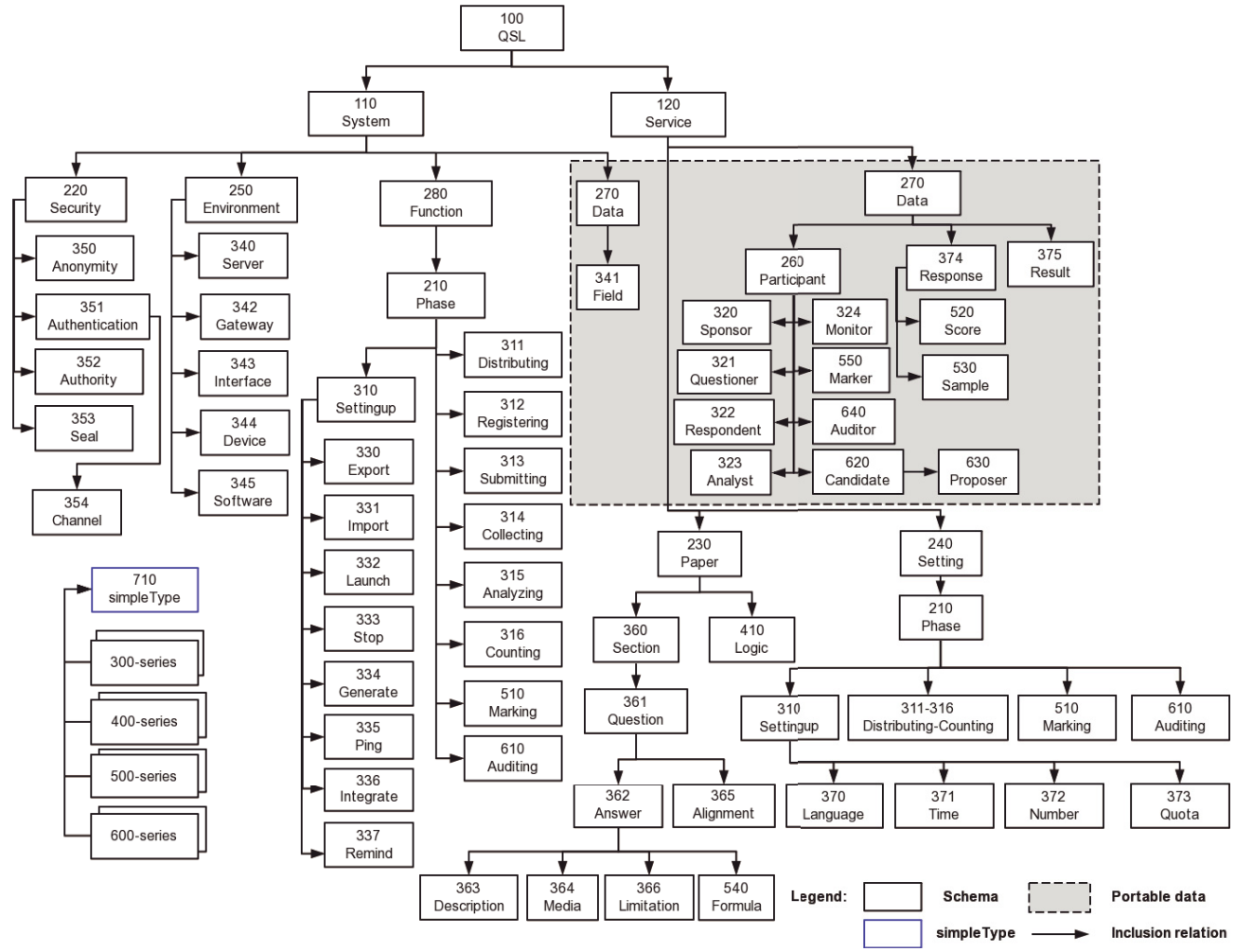

Fig. 5 Inclusion relation among schemas.

marked as 710 are used and included with 300-series, 400-series, 500-series, and 600-series schemas. Most visibly, both 240 schema and 280 schema include 210 schema. Previously discussing, phases are essential for systems and services of QTV. The stakeholders can easily identify three kinds of systems and services from the whole phases, and easily recognize functions or settings during a certain phase. Each phase schema should include function schemas and setting schemas. Considering validity and repeatability of schemas, 310 schema is defined to include these schemas, and other phase schemas add 310 schema. According to the relationship, the users can choose their demanded schema from the top level, which involve systematical requirements, and they cannot miss out any related requirements. In addition, it is easy to choose the desirable ones. Thus, as the proposed QSL template is combined by identifiable and distinct schemas that the stakeholders can specify independently. QSL structure has well-formed stability and extensibility.

Table 4 shows the QSL versions. Each version not only contains the schema files, but also the manual for using QSL. We are still improving QSL. From this table, it also proves that QSL has extensibility.

Table 4 QSL versions.

Date	Version	Contents and changes
2014.02	1.0-1.2	First QSL proposal and prototype of QSL for e-questionnaire systems
2014.09	1.3	QSL foundation in a frame
2014.11	1.4	Add some complex question and logic types
2014.12	1.5	Add formula, interface, functions
2015.01	1.6	Change QSL structure into 3 parts: system, paper, and security
2015.05	1.7	Add some details of phases, security, etc.
2015.11	1.8	Extend for e-testing systems
2016.08	2.0	Change structure for common and specific
2016.12	2.1	Extend some details for e-voting systems
2017.07	3.0	Change new QSL structure to desperate the simpleType and ComplexType for reuse
2018.08	3.1	Extract entities as 300-series schemas, and cover more than 30 requirements

In summary, QSL schemas can be used to help to specify exhaustive requirements owing to stability and extensibility of QSL's well-formed structure that contributes to the stakeholders to easily communicate with each other

by using QSL to describe the requirements of systems and services of QTV. In addition, QSL provides data template and as a unified data format for portable data among different QTV systems that contributes to improving data portability for providing conveniences for the stakeholders to reuse data.

3.4 Examples of QSL

Firstly, we give an example about snippet specification using QSL system template. It specifies a general-purpose offline e-testing system we developed [19]. The specification lists the functions in each phase. Our system can import paper data and setting data described by QSL. This file is distributed using a USB flash memory. The respondents login and are verified to ensure whether he is eligible or not. And the respondent answers the questions and submits his response. During the submitting phase, the monitor can monitor the whole phase by ping the respondents' IP address and monitor their test states. After submitting, the monitor can use the USB to store the collected responses by a zip method. The marker can mark the responses. In addition, the corresponding components are listed such as servers, devices, database, network, and interface.

```
<?xml version="1.0" encoding="UTF-8"?>
<QSL version="3.1" xml:lang="en" xmlns="
    http://www.aise.ics.saitama-u.ac.jp/
    qsl" xmlns:xs="http://www.w3.org/2001/
    XMLSchema" xmlns:xsi="http://www.w3.
    org/2001/XMLSchema-instance">
 <System>
   <Security/>
   <Environment>
     <Server id="server001" purpose="
         registering"/>
     <Server id="server002" purpose="
         submitting"/>...
     <Gateway id="ip001" ref="device001" ip
         ="10.0.5.1"/>
     <Interface id="interface001" ref="
         device003"/>
     <Device id="device001" type="PC"
         memory="8G" cpu="Intel Core i5"
         ref="soft001"/>
     <Device id="device002" type="access
         point" encryption="WPA2-PSK"
         max_connection="50"/>
     <Device id="device003" type="usb"
         edition="3.0" capacity="8G"/>...
     <Software id="soft001" ref="server002"
          purpose="submitting" role="
         respondent"><Solution type="
         browser" name="chrome"/></
         Software>
     <Software id="soft002" ref="server002"
          purpose="submitting" role="
         respondent"><Solution type="
         database" name="DB2" edition="
         8.1.6" data="participant"/></
         Software>
   </Environment>
   <Function>
     <Phase>
       <SettingUp>
         <Import id="func001" role="
             questioner" scope="paper and
             setting" format="qsl"/>
       </SettingUp>
       <Distributing>
         <Distribute id="func002" role="
             questioner" channel="usb"/>
       </Distributing>
       <Registering>
         <Authenticate id="func003" role="
             respondent" method="token"/>
       </Registering>
       <Submitting>
         <Ping id="func004" role="monitor"
             scope="ip"/>
         <Observe id="func005" role="monitor
             " scope="state"/>
         <Reply id="func006" role="
             respondent"/>
         <Submit id="func007" role="
             respondent"/>
         <Stop id="func008" role="respondent
             " method="interval" frequency=
             "1" auto_save="yes"/>
       </Submitting>
       <Collecting>
         <Collect id="func009" channel="usb"
             />
         <Integrate id="func010" role="
             questioner" scope="response"
             format="zip"/>
       </Collecting>
       <Marking>
         <Mark id="func011" role="marker"
             scope="response"/>
       </Marking>
     </Phase>
   </Function>
 </System>
</QSL>
```

In addition, QSL also can be used to specify the corresponding service. As an instance about a snippet specification for a questionnaire sheet and its setting shown in Fig. 6, we use QSL service template to fill the values.

```
<?xml version="1.0" encoding="UTF-8"?>
<QSL version="3.1" xml:lang="en" xmlns="
    http://www.aise.ics.saitama-u.ac.jp/
    qsl" xmlns:xs="http://www.w3.org/2001/
    XMLSchema" xmlns:xsi="http://www.w3.
    org/2001/XMLSchema-instance">
 <Service>
   <Paper id="paper001" type="
       questionnaire">
     <Description id="des001" type="header
         " value="A Questionnaire about
         Hangzhou City in China"/>
     <Description id="des002" type="
         paragraph" value="Special
         Insurance Policies of China Life
          Ltd."/>
     <Section id="section001">
       <Question id="question001" type="
           multiple choices" isMandatory=
           "yes">
         <Limitation minOccur="1" maxOccur
             ="1"/>
```

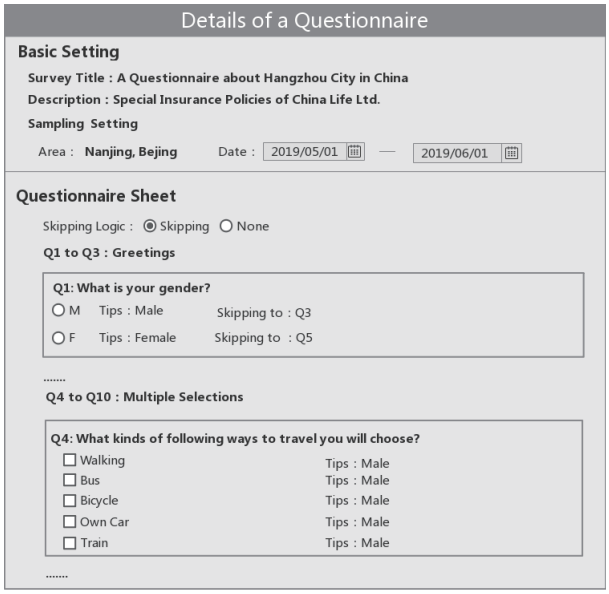

Fig. 6 A sample questionnaire.

```
10              <Description id="des003" type="
                    header" value="What is
                    your gender?"/>
11              <Answer id="answer001">
12                <Description id="des004" type
                      ="header" value="M"/>
13                <Description id="des005" type
                      ="tips" value="Male"/>
14              </Answer>
15              <Answer id="answer002">
16                <Description id="des006" type
                      ="header" value="F"/>
17                <Description id="des007" type
                      ="tips" value="Female"/>
18              </Answer>
19          </Question>
20          </Section>
21          <Section id="section002">
22          <Question id="question004" type="
                multiple choices" isMandatory=
                "yes">
23            <Limitation minOccur="1" maxOccur
                  ="6"/>
24              <Description id="des008" type="
                    header" value="What kinds
                    of following ways to
                    travel you will choose?"/>
25              <Answer id="answer003">
26                <Description id="des009" type
                      ="header" value="Walking
                      "/>
27              </Answer>
28              <Answer id="answer004">
29                <Description id="des010" type
                      ="header" value="Bus"/>
30              </Answer>
31              <Answer id="answer005">
32                <Description id="des011" type
                      ="header" value="Bicycle
                      "/>
33              </Answer>
34              <Answer id="answer006">
35                <Description id="des012" type
                      ="header" value="Own Car
                      "/>
36              </Answer>
37              <Answer id="answer007">
38                <Description id="des013" type
                      ="header" value="Train"/
                      >
39              </Answer>
40          </Question>
41          </Section>
42          <Logic>
43            <Route id="route001" type="
                  skipping">
44              <Condition answerId="answer001"
                    isChecked="yes"/>
45              <Action questionId="question005
                    "/>
46            </Route>
47            <Route id="route002" type="
                  skipping">
48              <Condition answerId="answer002"
                    isChecked="yes"/>
49              <Action questionId="question005
                    "/>
50            </Route>
51          </Logic>
52    </Paper>
53    <Setting id="setting001" ref="paper001">
54      <Phase>
55        <SettingUp>
56          <Language original="EN"/>
57          <Time enable="yes" zone="utc+8"
                start="2019-05-01 00:00:00"
                end="2019-06-01 00:00:00"/>
58          <Number enable="yes" order="
                ascending"/>
59        </SettingUp>
60         <Distributing>
61          <IP scope="Nanjing"/>
62          <IP scope="Beijing"/>
63        </Distributing>
64      </Setting>
65    </Phase>
66  </Service>
67  </QSL>
```

The last but not the least, QSL can be used as the data format to specify the data. The above-mentioned example is also a specification for data. Due to place constraints, the complete specifications please refer to [23].

3.5 Summary for QSL

Using QSL, it is helpful to easily and systematically choose desirable requirements, and describe machine-readable specifications for systems and services of QTV. According to the formalized requirement specifications, there is not an ambiguous presentation, so that solve the communication problems among the stakeholders. In addition, QSL can be used as a unified data format for portable data among different QTV systems that can improve data portability for providing conveniences for the stakeholders to reuse data. Moreover, when we version up QSL, we can easily revise the schemas without changing the general structure. The schemas have a stable structure and a well-designed exten-

sibility mechanism.

4. Evaluation

In order to approve the descriptive power of QSL that means QSL can be used to specify various QTV systems, we evaluated QSL about the descriptive power from following 5 aspects: system, service, data format, comparative study, and standard.

The system aspect is aimed to evaluate specifications by QSL for QTV systems. The service aspect is aimed to evaluate specifications by QSL for QTV services on the systems. The aspect of data format is aimed to evaluate specifications by QSL for portable data. The evaluation steps of these 3 aspects are as follows: to investigate the target systems and the services, to enumerate requirements of the target systems, services, or results of the services, to summarize enumerated requirements, to specify the requirements by QSL, and to evaluate whether QSL provides enough tags to specify the requirements or not.

The aspect of comparative study is aimed to compare with the related works to evaluate the coverage of QSL for QTV services, and the corresponding systems. The standard aspect is aimed to compare with the standards, laws, and ordinance to evaluate the compatibility of QSL. The evaluation steps of these 2 aspects are to investigate existing specification languages and standard for QTV systems, and enumerate their requirements, and compare with QSL to evaluate the coverage and compatibility of QSL.

4.1 Specifications for QTV Systems

In order to verify that QSL can be used to specify various QTV systems, at first, we investigated 29 e-questionnaire systems [3]–[12], [25]–[43], 22 e-testing systems [3]–[12], [36], [40], [44]–[53], and 24 e-voting systems [3]–[12], [54]–[67]. There are 10 systems providing QTV services. We enumerated requirements of each system. The 29 e-questionnaire systems have 60 requirements, the 22 e-testing systems have 78 requirements, and the 24 e-voting systems have 77 requirements. In addition, these systems have some common requirements for QTV systems. We used QSL to specify the summarized requirements by category of common and specific requirements for three kinds of the systems.

Current QSL is enough to specify 57 requirements for e-questionnaire systems, 75 requirements for e-testing systems, and 74 requirements for e-testing systems. The rest of the requirements that QSL cannot cover are listed below.

- **Error recovery**: the submitting server of QTV systems shall run a self-check before a resuming is possible. In case of irreversible problems the server shall prevent a resuming of the submitting phase.
- **Response database encryption**: QTV systems shall encrypt the response database.
- **Retrieve question**: QTV systems shall be capable of finding and getting back questions before submitting the responses.

From the result, QSL can be used to specify more than 95% requirements and is easy to extend to specify the rest of the requirements caused by well-formed structure. The above-listed requirements are the common requirements that will be added and defined in 300-series schemas. In general, it proves that QSL can specify various QTV systems. In the other words, QSL has the descriptive power for QTV systems.

4.2 Specifications for QTV Services on the Systems

In order to evaluate that QSL can be used to specify various QTV services on the systems, we respectively chose 10 representative e-questionnaires on the Statistic Japan [73], 10 e-testing on National Education Examination Authority (NEEA) [74], and e-voting based on the Europe and US standards [68] launched on 10 popular systems [3]–[12] to provide QTV services in the world. We enumerated the requirements for each service. We summarized the enumerated requirements. The 10 e-questionnaire have 65 requirements, the 10 e-testing have 55 requirements, and the 10 e-voting have 48 requirements. We also found these three kinds of e-services have in common such as the phases and paper structure. We used QSL to specify the summarized requirements by category of common and specific requirements for three kinds of services.

Current QSL can specify 62 requirements for e-questionnaire, 53 requirements for e-testing, and 46 requirements for e-testing. The rest of the requirements that QSL cannot cover are listed below.

- **CSS style**: QTV services shall provide css style design for the paper design.
- **Dynamic multi-tier lookup table**: e-questionnaire shall provide a text box to represent hierarchies of data likes parent to child.
- **Theme library**: QTV services shall provide a theme library for design the paper.

In addition, the reasons of the rest requirements QSL cannot cover, on the one hand is not necessary requirements for QSL because these requirements concern paper design but QSL focus on the functions of the services to support all the phases of services so that it does not need to consider, such as “css style” and “theme library;” on the other hand is easy to extend QSL to contain it, such as the “dynamic multi-tier lookup table.”

From the result, QSL can be used to specify more than 95.4% requirements and is easy to extend to specify the rest of the requirements caused by well-formed structure. The requirements named “dynamic multi-tier lookup table” will be added and defined in 300-series schemas. In summary, It proves that QSL can specify various QTV services. In the other words, QSL has the descriptive power for QTV services on the systems.

4.3 Specifications for Portable Data

In order to verify that QSL can be used as format of portable data of QTV services, that means to check whether QSL can provide enough tags and notations to describe the results of QTV, we referred to the method listed up previously. We launched QTV, and imitated the respondents to answer so that obtained the results of QTV. We enumerated the requirements of the results from each service, and summarized the enumerated requirements. The 10 e-questionnaire have 13 requirements, the 10 e-testing have 16 requirements, and the 10 e-voting have 11 requirements for specifications of data. We also found some requirements of data have in common.

In addition, we used QSL to specify the results. Current QSL can specify all the requirements.

From the result, QSL can specify the whole requirements. In conclusion, it proves that QSL can specify various data of QTV services. In the other words, QSL has descriptive power for portable data.

4.4 Comparison with Related Works

In order to verify that the descriptive power of QSL can cover various QTV systems, we compared with existing specification languages with different motivations for specifying QTV systems, respectively. All the existing languages are based on XML.

At first, Bethke [15] proposed a questionnaire specification language in Simple Survey Systems (SSS) to provide a straight-forward way to present questionnaires to untrained users for survey research. The language can be use to specify structure, content, and a part of question types of questionnaires, as well as two kinds of dynamic logic types [75]. SSS focuses on presenting what a simple questionnaire has. Secondly, IMS Question and Test Interoperability Specification (QTI) [16] is aimed at providing a standard format for the representation of assessment content and results, supporting the exchange of this material between authoring and delivery systems, repositories and other learning management systems. QTI focuses on presenting what a test paper and a result report have, and how to process response and operation for data exchange. Thirdly, OASIS Election Markup Language (EML) [17] is a standardized XML language for interchange of data among hardware, software, and service providers who engage in any aspect of providing election or voter services to public or private organizations. EML focuses on describing what and how the data interchange in a secure election process in details.

Compared with the above-mentioned languages, QSL is aimed at providing a unique and common method for the communications of the stakeholders among QTV systems. Because of this motivation, QSL should have the descriptive power to specify what QTV systems have. In other words, QSL is able to specify the requirements for various QTV systems, which are listed in Table 5. This table also shows the requirements what QTI and EML can specify. Because the terminologies of QSL and the related works are different, the requirements are described according to the terminology of QSL.

Table 5 A list of the requirements with related works.

Region	Requirements
#1	paper: section, question, answer, etc.
	question type: single choice, multiple choices
#2	question type: multiple-row box, single-row text
#3	logic: simple skipping, compound skipping
#4	question type: drag, connect the points, slide, etc.
	logic: randomization
	setting: media, weighting, etc.
	data: sample, score, comment, etc.
#5	question type: formula, drop-down list, matrix, etc.
	setting: ordering, numbering, etc.
	data: respondent information, result
#6	setting: multi-language, launch/finish time. etc.
	participants: candidate, executor
	phase: setting up, distributing, registering, submitting, collecting, analyzing, counting, auditing
	security: authentication, anonymity, channel, etc.
	environment: interface and it communicates with security
#7	logic: compound skipping, piping, extraction
	setting: spelling checking, automatic lookup, etc.
	participant: sponsor, executor (marker), auditor
	phase: marking
	environment: server, database, device, software, etc.
	function: export, import, monitor, etc.
#8	setting: css, attempt times for incorrect answer, etc.
	function: tester feedback sample correctness
#9	data: bureaucracy information, etc.
	environment: processing unit, identifier with election, etc.

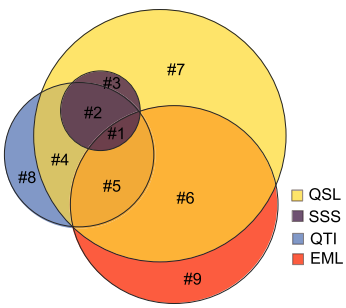

Fig. 7 Descriptive power of QSL with related works.

Figure 7 illustrates a straight-forward relation overview of descriptive power of QSL, SSS, QTI, and EML. From this figure, QSL can specify all the requirements to satisfy the motivation. As we found that QSL can specify most of the requirements except them in region 8 and region 9.

Until now QSL is not considering e-voting for e-learning services, and the css style. Hence, the items in region 8 are not the necessary requirements of QSL. Moreover, the identifier is used to refer to the EML-based program for connect with the EML schemas. The bureaucracy information such as some office hours are not necessary for QSL. QSL focuses on what the systems have, what the phases have, so the requirements how to process and the detailed information having no concerns with systems and phases are also not necessary requirements for QSL. The

above-mentioned requirements in region 8 and region 9 are the optional requirements that QSL can be slightly and easily extended.

From the result, QSL is a better tool than SSS, QTI, and EML. Because QSL has enough coverage for specifying various QTV systems, as well the stakeholders can learn only QSL to deal with the three kinds of systems. After all, when the stakeholders use QTI, SSS, or EML, they have to learn three languages to deal with three systems. In addition, There are existing 10 systems providing QTV services we mentioned and investigated above. It is necessary to provide specifications for QTV services, as well as the corresponding systems. QSL can satisfy the demand in tend and can be used to specify these three kinds of services and the corresponding systems.

4.5 Comparison with Standard

There are existing standards which are about guidelines defined by various official organizations for how to do QTV. It is very useful for the sponsors, the executors, and the supporters if there is a specification language which can create the specification of QTV systems conforming to the standards of QTV. In order to verify the compatibility of QSL with the standards, we evaluated whether QSL is suitable for European Electronic Voting Recommendation Standard (EEVRS) [76].

EEVRS is the legal, operational and technical recommendation standard for electronic voting, adopted by the Committee of Ministers of the Council of Europe. It listed 112 requirements from the standard in the first recommendation Rec(2004)11. In 2017, for adherence to the democratic principles, recommendation standard in Rec(2017)5 has been developed that includes 49 additional requirements [77]. A total of 161 requirements consists of 82 legal, 25 operational, and 54 technical requirements.

Comparing with the requirement list by QSL schemas, in the 161 requirements, current QSL can satisfy 89 requirements. And QSL does not need to cover 67 requirements because they are about the formality, the staffs how to operate the system, and irrelevant to the system itself. There are 5 requirements that QSL does not meet. We list satisfied, unsatisfied, and irrelevant examples as follows.

Firstly, QSL is satisfied EEVRS **S15** "the e-voting system shall prevent the changing of a vote once that vote has been cast." Because QSL can be used to specify the requirements that only one ballot by one respondent is saved to the server and the votes cannot access and change the voted ballot. And the corresponding requirements of QSL are Q74, Q75 listed in [23]. Secondly, an irrelevant example is EEVRS **S20** "member states shall take steps to ensure that voters understand and have confidence in the e-voting system in use." The reason is that this is not a system requirement. As the last example about an unsatisfied requirement is EEVRS **S89** "the integrity of data communicated from the pre-voting stage (e.g. voters' registers and lists of candidates) shall be maintained. Data-origin authentication shall be carried out." Because QSL cannot be used to specify the requirements to guarantee the integrity of the list of candidates generated at the pre-voting stage by digital signatures.

The rest of the 5 unsatisfied requirements are listed as follows. For S86, S89, S97, and S160, QSL cannot be used to describe the requirements to guarantee the integrity of the list of candidates. As to S104, there is no description for proving e-voting process follows the legal provisions.

- **S86**: The authenticity, availability and integrity of the voters' registers and lists of candidates shall be maintained. The source of the data shall be authenticated. Provisions on data protection shall be respected.
- **S89**: The integrity of data communicated from the pre-voting stage (e.g. voters' registers and lists of candidates) shall be maintained. Data-origin authentication shall be carried out.
- **S97**: The integrity of data communicated during the voting stage (e.g., votes, voters' registers, lists of candidates) shall be maintained. Data-origin authentication shall be carried out.
- **S104**: The audit system shall provide the ability to oversee the election or referendum and to verify that the results and procedures are in accordance with the applicable legal provisions.
- **S160**: The authenticity, availability and integrity of voters' registers and lists of candidates shall be maintained. The source of the data shall be authenticated. Provisions on data protection shall be respected.

Except for irrelevant requirements, QSL can cover 94.7% requirements for EEVRS. According to the result of comparison with EEVRS, it proves that QSL is useful. Even if QSL does not satisfy these 5 requirements, it is without prejudice to the reliability and security of e-voting because human beings can check and confirm them. In addition, because the requirements listed by QSL is fulfilled with EEVRS, QSL can be expected to be used flexibly and adapted to other e-voting systems. After all, it is convenient and efficient to have a ready-made formalized requirement list that satisfies the standards.

4.6 Summary for Evaluation

Overall, according to the above evaluation results, it can be concluded that: first of all, QSL can be used to specify various QTV services, as well as the corresponding data and systems. Secondly, QSL is a better tool because of its coverage for the specifications of QTV systems. At last, QSL can be expected to be used flexibly and adapted to other systems because of the compatibility of it.

5. The Cases for Using QSL

5.1 ENQUETE-BAISE: a General-Purpose E-Questionnaire Server for Ubiquitous Questionnaire

ENQUETE-BAISE [1] is a general-purpose e-questionnaire

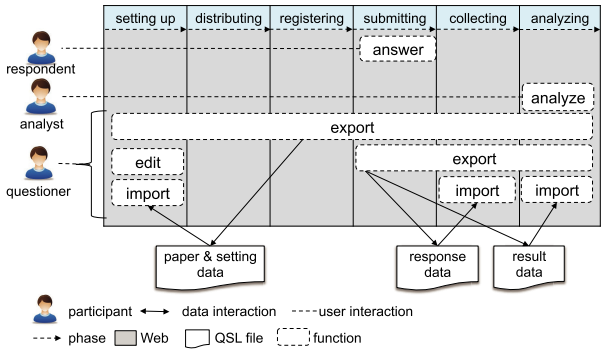

Fig. 8 Overview of ENQUETE-BAISE for portable data.

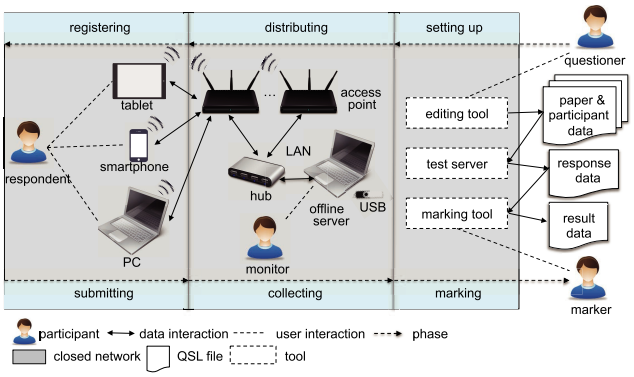

Fig. 9 Overview of general-purpose offline e-testing environment.

server developing for ubiquitous questionnaire that can be used as a readymade e-questionnaire server component in various web service systems as well as an alone e-questionnaire server with general-purpose for various questionnaires. It can also be used as an e-testing server and an e-voting server with general-purpose by restricting its general functions and strengthening its security functions [2]. ENQUETE-BAISE [78] has been used to support students in Saitama University to access, login, and use it to do QTV since 2007.

ENQUETE-BAISE lacks data portability because it did not have a format of portable data. QSL is used as a data format for improving data portability of ENQUETE-BAISE [71]. It helps to conduct an e-questionnaire on different e-questionnaire systems. The data of paper, setting, responses, and result have been specified by QSL.

Figure 8 illustrates the overview of ENQUETE-BAISE for portable data. We implemented and improved ENQUETE-BAISE for data portability. All the data are exchanged in a QSL specification file. A questioner can edit a paper and set up settings for the paper. The paper data and setting data are saved in the database in real-time. The questioner also can import a QSL format requirement specification recording paper data and setting data during setting up phase. The questioner can export paper data and setting data by QSL format during the whole phases. The respondents answer questions, and the response data are saved in the database during submitting phase. The questioner can export and download the submitted response data. After collecting the responses, all the response data are integrated and send to analyst to analyze and get the result. ENQUETE-BAISE can automatically analyze and get result. The result data is saved in the database and the questioner can export the result data. Sometimes, the questioner can import response data to analyze by ENQUETE-BAISE, and import the result data to record in database.

5.2 A General-Purpose Offline E-Testing Environment

We implemented a general-purpose offline e-testing environment [19] based on QSL to provide users with offline e-testing service to execute various offline e-testing. This environment has been applied to execute final test of Discrete Mathematics in University of Japan since 2015 [79].

Considering it is difficult for teachers to manage test data when to execute offline e-testing in classrooms, QSL is used as a data format to solve the difficulty about management such that the environment provides users with offline e-testing service to execute various offline e-testing. The data of exam paper, participant, responses, and result have been developed by QSL, and the QSL format data has been used directly. Moreover, we used QSL to define all the requirements of this system in the phase of pre-development.

The overview of the general-purpose offline e-testing environment is illustrated in Fig. 9. An editing tool is set into the USB flash memory. A questioner uses the editing tool to specify all the participants by QSL. After that, the editing tool will generate the admission ticket for each respondent automatically. The questioner uses the editing tool to prepare exam paper with complex question types and logic, which files are specified by QSL. An offline test server distributes exam paper to respondents through wireless LAN. The environment uses access points to support to test a large number of respondents easily and conveniently. The respondents confirm the exam paper from the offline server. During the test, the monitor monitors the connection states of all the respondents through the offline server for avoiding online cheating activities, and the closed network will also block accesses from outside to connect to the offline server. After submitting, all the responses will be collected and integrated by the offline server as QSL format files. The offline server distributes and collects offline e-testing. The questioner only needs to stick the USB flash memory into a PC. The PC can be the offline server to execute offline e-testing. The marking tool is also set into the USB flash memory. A marker uses the marking tool to mark the collected responses and give a result of the test as a QSL format file. An analyst uses the marking tool to analyze responses and get the test result automatically. The functions of the tools and phases, as well as the environments, are defined by QSL.

6. Various QSL Applications

6.1 Essential Application for QSL

Hundreds of requirements and their relationships and constraints for systems and services of QTV specified in a QSL format are quite complicated. From the viewpoints of the stakeholders, they are unwilling to learn the details of QSL such as what tags have and what meanings are. In fact, it is a waste of time and cost to learn QSL and write QSL format requirement specifications according to QSL definition. Improving efficiency and reducing the costs of using QSL should be taken into careful consideration. In addition, adapting QSL format requirement specifications to other systems is desirable. Simplifying the writing and processing procedures for using QSL, is much easier and more flexible to be used and adapted in other systems.

Thus, we propose to implement a QSL structure editor that the users can easily create a QSL format requirement specification and they are not necessary to know the structure and terminology of QSL.

Figure 10 shows the usage of QSL structure editor. The users can use the editor to guide them to choose the corresponding requirement templates according to their suitable identities and even import the QSL format requirement specifications. The editor performs the requirement list with a graphical user interface (GUI) that hide the code in the background and present the content in more user-friendly forms for easy understanding, and guide them to fill the appropriate values. In the background processing, the editor parses the selected QSL schema into the corresponding QSL template, when the users select a requirement, it can real-time correlate with the corresponding requirement in the GUI. After filling all the values, the editor is used to verify the values are correct or not in conformance with QSL schemas, and replace the terminology for the tags, and then output the formalized requirement specification.

Using QSL structure editor can assist the stakeholders to save the time and cost to easily read and write QSL format requirement specifications. Moreover, QSL can be expected to be used and adapted flexibly into other systems.

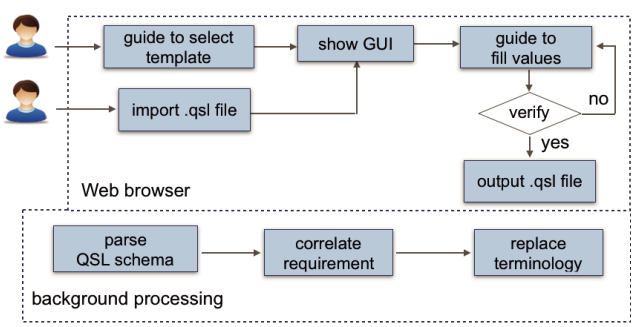

Fig. 10 Usage of QSL structure editor.

6.2 Expectable Applications for QSL

QSL is an XML-based language, differing with the natural language, it is machine-readable. There are two expectable applications for QSL:

- **QSL verifier** to check whether the requirements in specifications are enough or not, and check whether the specification conforms to the standards, laws, and regulations.
- **System generator** to convert from QSL format requirement specifications to generate QTV systems easily and automatically.

Figure 11 shows the expectable applications for QSL. The stakeholders (usually sponsor and executor) discuss and use QSL structure editor to input values to get a service requirement specification. Based on the service requirement specification, the supporter can design a system requirement specification by this editor. This editor can help stakeholders to write specification easily. The requirement specification is as the input data into a QSL verifier, and the verifier output the result of the verification. If the result is yes, then output the reliable requirement specification and store the output data into the QSL database. If there are errors or mistakes, then output the feedback, and the stakeholders can improve the specification by checking and according to the feedback. If the stakeholders want to reuse QSL document, it is necessary to provide the database. And the editor and verifier can also use the data in the QSL database. Because QSL format requirement specifications are machine-readable data, the system generator inputs the reliable requirement specification and can generate some encapsulated code. The code through the processor and run can get a system to do QTV services.

Previously, we compared QSL with the standards. Using the QSL verifier can assist the stakeholders to mechanically verify the standards with QSL format requirement specifications. We advocate that QSL can be expected to be used and adapted flexibly into other systems, after all, this is a plus point for other systems. The system generator is a great advantage for the stakeholders that automatically generates desirable systems as the ideal goal. The various applications for QSL are aimed to provide one-stop service of easily creating, verification, and generation in order to make convenience for using QSL, essentially to solve the

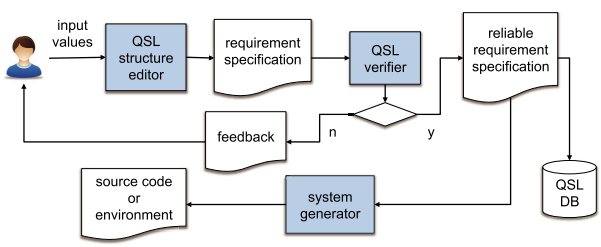

Fig. 11 QSL verifier and system generator.

communication problems.

7. Concluding Remarks

This paper proposes QSL, the first specification language, with a standardized, consistent, and exhaustive list of requirements for specifying various QTV systems such that the specifications can be used as the precondition of automatically generating QTV systems.

As the contributions, firstly, we collected, analyzed, and summarized exhaustive requirements for systems and services of QTV. Secondly, we proposed QSL as a unified method to provide a standardized terminology and an exhaustive requirement list to solve the communication problems among the stakeholders. Thirdly, we evaluated the proposed QSL about description power to ensure its completeness manifesting in specifying various QTV systems. Lastly, we proposed an alternative usage of QSL for data portability and implemented two real applications for QSL to prove it is practical and available to be used to provide data format for data portability. And we proposed applications for QSL to simplify stakeholders' works and streamline their procedures, and even solve communication problems.

The ultimate goal of this research is to support all kinds of stakeholders to easily communicate with each other by QSL. In the future, we continue improving QSL through case studies in order to provide a complete requirement list and satisfy standards of e-questionnaire, e-testing, e-learning, and e-voting. Moreover, a series of supporting tools for QSL such as QSL structure editor, QSL verifier, and system generator should be developed. These tools can be used to save cost and improve efficiency.

References

[1] J. Cheng, Y. Goto, M. Koide, K. Nagahama, M. Someya, Y. Utsumi, and A. Shionoiri, "ENQUETE-BAISE: a general-purpose e-questionnaire server for ubiquitous questionnaire," Proc. IEEE Asia-Pacific Services Computing Conference, IEEE-CS, Tsukuba, Japan, pp.187–194, 2007.
[2] Y. Goto and J. Cheng, "Information assurance, privacy, and security in ubiquitous questionnaire," Proc. 4th International Conference on Frontier of Computer Science and Technology, IEEE-CS, Shanghai, China, pp.619–624, 2009.
[3] AddPoll, http://www.addpoll.com, accessed Aug. 25. 2018.
[4] Constant Contact, http://www.constantcontact.com, accessed Aug. 25. 2018.
[5] D.C. Quan, http://www.diaochaquan.cn, accessed Aug. 25. 2018.
[6] eSurv, http://eSurv.org, accessed Aug. 25. 2018.
[7] Mobo Survey, http://www.mobosurvey.com, accessed Aug. 25. 2018.
[8] MySurveyLab, https://www.mysurveylab.com, accessed Aug. 25. 2018.
[9] ProProfs, http://www.proprofs.com, accessed Aug. 25. 2018.
[10] QuestionPro, http://www.questionpro.com, accessed Aug. 25. 2018.
[11] So Jump, http://www.sojump.com, accessed Aug. 25. 2018.
[12] SurveyMonkey, http://www.surveymonkey.com, accessed Aug. 25. 2018.
[13] N. Rozanski and E. Woods, Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives, Addison-Wesley, 2012.
[14] S. Donald and M. Wirsing, "Specification languages," in Algebraic Foundations of Systems Specification, pp.243–272, Springer, Berlin, Heidelberg, 1999.
[15] A.D. Bethke, "Using XML as a questionnaire specification language," Proc. IEEE Southeast Conference 2007, Richmond, pp.127–131, 2007.
[16] IMS Global Learning Consortium, "IMS question and test interoperability (QTI) specification," http://www.imsglobal.org/question/index.html, accessed Aug. 25. 2018.
[17] OASIS, "Election markup language (EML) specification version 7.0," http://docs.oasis-open.org/election/eml/v7.0/cs01/eml-v7.0-cs01.html, accessed Aug. 25. 2018.
[18] Y. Zhou, Y. Goto, and J. Cheng, "QSL: a specification language for e-questionnaire systems," Proc. 5th IEEE International Conference on Software Engineering and Service Science (ICSESS 2014), IEEE, Beijing, China, pp.224–230, 2014.
[19] Z. Wang, Y. Zhou, B. Wang, Y. Goto, and J. Cheng, "An extension of QSL for e-testing and its application in an offline e-testing environment," in J.J. Park, et al. (Eds.), Advanced Multimedia and Ubiquitous Engineering, Lecture Notes in Electrical Engineering, vol.352, pp.7–14, Springer, 2015.
[20] Y. Zhou, H. Gao, and J. Cheng, "QSL: a specification language for e-questionnaire, e-testing, e-voting systems," Y. Zhou, H. Gao, and J. Cheng, "QSL: a specification language for e-questionnaire, e-testing, e-voting systems," in J.J. Park, et al. (Eds.), Advanced Multimedia and Ubiquitous Engineering, Lecture Notes in Electrical Engineering, vol.393, pp.255–261, Springer, 2016.
[21] Y. Zhou, H. Gao, and J. Cheng, "An extension of QSL for e-voting systems," in J.J. Park, et al. (Eds.), Advances in Computer Science and Ubiquitous Computing, Lecture Notes in Electrical Engineering, vol.421, pp.87–96, Springer, 2016.
[22] Y. Zhou, D. Matsuura, Y. Goto, and J. Cheng, "Evaluation about the descriptive power of QSL: a specification language for e-questionnaire, e-testing, and e-voting systems," Proc. 4th IEEE Smart World Congress (SmartWorld 2018), IEEE-CS, Guangzhou, China, pp.198–203, 2018.
[23] AISE Lab, "QSL specification," http://www.aise.ics.saitama-u.ac.jp/qsl/, accessed April 22. 2018.
[24] European Commission, "Data protection," https://ec.europa.eu/info/law/law-topic/data-protection_en, accessed June 12, 2018.
[25] A.D. Yan, http://www.idiaoyan.com, accessed Aug. 25. 2018.
[26] AskForm, http://www.askform.cn, accessed Aug. 25. 2018.
[27] CubeQuery, http://cubequery.jp, accessed Aug. 25. 2018.
[28] Dounano, http://www.dounano.jp, accessed Aug. 25. 2018.
[29] EnableQ, http://enableq.chinaedu.net/System/Login.php, accessed March 10. 2019.
[30] Enq-maker, https://enq-maker.softonic.jp/web, accessed March 10. 2019.
[31] Fluidsurveys, http://fluidsurveys.com, accessed Aug. 25. 2018.
[32] FormSite, http://www.formsite.com, accessed Aug. 25. 2018.
[33] Key Survey, https://www.keysurvey.com, accessed Aug. 25. 2018.
[34] Moodle, http://www.moodle.org, accessed Aug. 25. 2018.
[35] My3Q, http://www.my3q.com, accessed Aug. 25. 2018.
[36] OQSS, http://www.oqss.com, accessed Aug. 25. 2018.
[37] Qualtrics, http://www.qualtrics.com, accessed Aug. 25. 2018.
[38] Smart Survey, http://www.smartsurvey.co.uk, accessed Aug. 25. 2018.
[39] Smaster, http://www.smaster.jp, accessed Aug. 25. 2018.
[40] Sogo Survey, http://www.sogosurvey.com, accessed Aug. 25. 2018.
[41] Survey Moz, http://www.surveymoz.com, accessed Aug. 25. 2018.
[42] Surveyi, http://www.surveyi.com, accessed Aug. 25. 2018.
[43] T.H.D.Y. Bao, http://www.diaoyanbao.com, accessed Aug. 25. 2018.
[44] ExamSoft, https://learn.examsoft.com, accessed March 10. 2019.
[45] Examcoo, http://www.examcoo.com, accessed Aug. 25. 2018.
[46] Ischool Random Testing System (ver. 3.7.1), http://down.chinaz.

com/soft/26226.htm, accessed Aug. 25. 2018.
[47] LoveKao, http://down.chinaz.com/soft/29038.htm, accessed Aug. 25. 2018.
[48] OCR, https://www.ocr.org.uk, accessed Aug. 25. 2018.
[49] oExam, http://www.orivon.com, accessed Aug. 25. 2018.
[50] PHPEMS Online Testing System, http://www.phpems.net, accessed Aug. 25. 2018.
[51] Qi Bo Testing System, http://down.chinaz.com/soft/29998.htm, accessed Aug. 25. 2018.
[52] Tomexam Online Testing System, http://www.tomexam.com, accessed Aug. 25. 2018.
[53] Yong Dao Offline Testing, http://www.onlinedown.net/soft/49716.htm, accessed Aug. 25. 2018.
[54] U. Madise and T. Martens, "E-voting in Estonia 2005," The First Practice of Country-wide Binding Internet Voting in The World, In: R. Krimmer (Eds.), Electronic Voting 2006, LNI, Bonn, Germany, pp.15–26, 2006.
[55] FC2 Vote, http://vote.fc2.com, accessed Aug. 25. 2018.
[56] Google Form, http://www.google.cn/intl/zh_cn/forms/about/, accessed March 10. 2019.
[57] Helio, https://vote.heliosvoting.org, accessed Aug. 25. 2018.
[58] Opinion Stage, https://www.opinionstage.com, accessed Aug. 25. 2018.
[59] Poll Every Where, https://www.polleverywhere.com, accessed Aug. 25. 2018.
[60] POLYAS, https://www.polyas.com, accessed Aug. 25. 2018.
[61] Simple Voting, https://www.simplyvoting.com, accessed Aug. 25. 2018.
[62] Snappy Poll, https://www.snappypoll.com, accessed Aug. 25. 2018.
[63] Stone Poll, http://www.stonepoll.com, accessed Aug. 25. 2018.
[64] T.P. Wang, https://www.toutoupiao.com, accessed Aug. 25. 2018.
[65] Vizzual Forms, http://www.vizzualforms.com, accessed Aug. 25. 2018.
[66] Votenet, http://www.votenet.com, accessed Aug. 25. 2018.
[67] YouPoll, http://www.youpolls.com, accessed Aug. 25. 2018.
[68] M. Volkamer, Evaluation of Electronic Voting: Requirements and Evaluation Procedures to Support Responsible Election Authorities, Lecture Notes in Business Information Processing, vol.30, Springer, 2009.
[69] W3C, "Extensible markup language (XML) 1.0 (fifth edition)," http://www.w3.org/TR/2008/REC-xml-20081126/, accessed Aug. 25. 2018.
[70] E. van der Vlist, XML Schema, O' Reilly, New York, 2002.
[71] Y. Kamata and Y. Goto, "Improvement of data portability of ENQUETE-BAISE: a general-purpose e-questionnaire server for ubiquitous questionnaire," Proc. 4th IEEE Smart World Congress (SmartWorld 2018), IEEE-CS, Guangzhou, China, pp.174–179, 2018.
[72] P.R. Benson, "Data portability: it is about the data; the quality of the data," Proc. MIT Information Quality Industry Symposium, pp.614–618, 2009.
[73] Statistic Bureau, Ministry of Internal Affairs and Communications, http://www.stat.go.jp, accessed Aug. 25. 2018.
[74] National Education Examinations Authority, http://www.neea.edu.cn, accessed Aug. 25. 2018.
[75] A.D. Bethke, "Representing procedural logic in XML," Journal of Software, vol.3, no.2, pp.33–40, 2008.
[76] Council of Europe, "Legal, operational and technical standards for e-voting, Recommendation Rec(2004)11 adopted by Committee of Ministers of Council of Europe on 30 Sept. 2004 and explanatory memorandum," http://www.eods.eu/library/CoE_Recommentation%20on%20Legal,%20Operational%20and%20Technical%20Standards%20for%20E-voting_2004_EN.pdf, accessed Aug. 3. 2018.
[77] A.D. Maurer, "Updated European standards for e-voting," Proc. International Joint Conference on Electronic Voting, Springer, Cham, pp.146–162, 2017.
[78] AISE Lab, "ENQUETE-BAISE: a general-purpose e-questionnaire server for ubiquitous questionnaire," http://www.aise.ics.saitama-u.ac.jp/enquete/, accessed Aug. 25. 2018.
[79] H. Gao, Z. Wang, Y. Zhou, and J. Cheng, "Development of a general-purpose offline e-testing environment," Proc. 12th International Conference on Computational Intelligence and Security (CIS 2016), IEEE, Wuxi, China, pp.603–607, 2016.

Yuan Zhou received the Bachelor of Technology degree in computer science and information technology, University of Mysore in India in 2011, the degree of Master of Engineering and Doctor of Engineering in computer science from Saitama University in Japan in 2015 and 2019, respectively. Her current research interests include Web services, safety engineering, and Internet of Things.

Yuichi Goto is an associate professor of computer science at Graduate School of Science and Engineering, Saitama University in Japan. He received the degree of Bachelor of Engineering in computer science, the degree of Master of Engineering in computer science, and the degree of Doctor of Engineering in computer science from Saitama University in 2001, 2003, and 2005, respectively. His current research interests include relevant reasoning and its applications, automated theorem finding, epistemic programming, anticipatory reasoning reacting systems, and Web services. He is a member of ACM, IEEE-CS, IEICE, IPSJ, and JSAI.

Jingde Cheng is a Professor Emeritus at Saitama University, and now a Professor at Southern University of Science and Technology, China. He received the Bachelor of Engineering degree in computer science and the Best Graduate Award from Tsinghua University in 1982, and the Master of Engineering degree and the Doctor of Engineering degree, both in computer science from Kyushu University in 1986 and 1989 respectively. Before he joined Southern University of Science and Technology in 2018, he was a research associate (1989-1991), an associate professor (1991-1996), a professor (1996-1999, as the first foreign professor in Japanese national universities) at Kyushu University, and a professor (1999-2018) at Saitama University. He is a senior member of ACM, and a member of IEEE-CS, IEEE-SMC, IEEE, and IPSJ. His current research interests include relevant reasoning and its applications, automated theorem finding, epistemic programming, anticipatory reasoning reacting systems, and information security engineering environment.