KINGDOMS
A Societal Model for Putting Humans Back in Charge

Miroslaw Malek, Peter Ibach, Nikola Milanovic, Jan Richling, Felix Salfner

Humboldt-Universität zu Berlin, Unter den Linden 6
10099 Berlin, Germany
{malek, ibach, milanovi, richling, salfner}@informatik.hu-berlin.de

Abstract. In an Internet of Things users are confronted with a growing complexity and dynamicity of interaction patterns and new questions arise concerning ownership, trust and liability. To address these questions, a societal model of computing, the KINGDOMS, is introduced. It allows to trace back every entity to exactly one domain managed by a legal authority, a “king”, and derives interaction rules analogously to the societal model. In contrast to physical, biological or other societal models, the KINGDOMS model offers better insight into the future world of services where the issues of adjustability, liability, dependability, security and scalability are of an essence. It also offers a new organizational paradigm for putting humans back in charge.

Keywords: adaptivity, dependability, human-centric computing, interoperability, liability, mobility, service-oriented computing

1 The Challenge

After a period of explosive, unmitigated growth, the world is ready for the next chapter of the IT revolution, namely, the support of ambient, human-centered ubiquitous computing where dynamic change, flexibility, on-demand components and services configurability will be expected. The web with virtually billions of devices attached to it and quintillions of bytes of data will have to transform into an ambient information, knowledge and remote control utility available to unprecedented numbers of novice as well as mature users anywhere and anytime [6]. The dream of adaptive, maintenance-free, self-relying and user-centered systems must become a reality as public dependence on those systems will continue to rise and there will be insufficient human resources to continually support and maintain the computing/communication infrastructure [1]. This article lays out a foundation for this challenge.

To conquer the challenge, the KINGDOMS (Kingdoms of Interoperable Next Generation Dependable Open Mobile Systems) are introduced. The KINGDOMS potentially have a larger population than mankind: the full manifold of software and electronic devices ranging from RFID tags and network-attached temperature sensors over networked video recorders, cameras and desktop computers to full-blown server farms. We call this concept *subdued* or *ambient* computing to emphasize that humans should control the computers they own in a way that is not a burden but an assistance.

The KINGDOMS is a societal model of computing that offers humans to become “kings” and to be again in charge of computers and their impact on lives. Computers have traditionally developed along the lines of human activities, and of organizations, government bodies and recently families.

Starting with requirements, we derive a societal model reflecting a drive towards service-oriented, ambient and human-centric computing. Uniqueness of this model is based on the introduction of concepts to computer systems such as ownership, liability and trust, which are also common in human societies.

2 Requirements

In order to develop and motivate our model of the KINGDOMS we start with an analysis of requirements that are essential for our vision to become reality. We take a human’s perspective because, ultimately, supporting the humans should be the main purpose of a wide-scale service-oriented world.

Controllability. Humans expect a reliable function of computer systems in a way that ensures that they have control and are not overwhelmed or disturbed by machines in an unwanted way. This might be a contradiction in itself because non-disturbance may imply that complexity should be hidden from the user while control may require the opposite. Furthermore, the level of desired complexity may vary among users as well as for different purposes: Some user trust a system only if the user can see what is happening while another user would not use a system that continuously interrupts their work by presenting too many details. Therefore, the model must allow an *adjustable level of abstraction*. This also addresses the issue of privacy as we are dealing with systems that are embedded in the private environment and that may at the same time communicate world wide.

Liability. Following the vision of a service oriented world on a global scale [7], services of different origins will autonomously and automatically cooperate in order to serve a purpose that was defined by a human. With increasing acceptance and spreading of services, the number of such requests and the criticality with respect to the real world will dramatically increase. Therefore, in the case of physical or financial damage to the real world, one of the essential concerns is the issue of *liability*. This is especially true for business-to-consumer and business-to-business interactions. Hence, liability must be one of the central requirements of our model.

Dependability. Due to an increased and broad usage of services, dependence on correctness and reliability of those services increases as well. Additionally, if one is liable for provided services, reliable operation must be ensured (to as high extent as possible). Furthermore, dependability characteristics must be available to potential users of the service allowing them to assess what type of service and what quality of service is guaranteed. Such guarantees should encompass non-functional properties such as *dependability* including *security* with appropriate quality levels.

It is a well-known trend that future computing scenarios will to a large extent consist of mobile devices that have to adapt to the environment they operate in. Since this trend is already a reality in many areas, any viable model of organization of computing environments should include *mobility* and *adaptivity* as key requirements.

Scalability. As already stated, there is a wide range of potential participants in such a computing environment ranging from very small scale sensor devices up to large server farms. Their capabilities are very different and it is unrealistic to assume that all of them can participate in an all-to-all communication scheme. This would imply that a service environment has to be simple enough to be exercised on small devices without being too simple to fulfill the needs of big systems. On the other hand, understanding the service language or providing the necessary network connection including all protocols may well exceed the capabilities to be executed by a simple device.

Consider, for example, a simple temperature sensor node consisting of an 8-bit microcontroller, a sensor attached to an AD-converter, sending temperature values to a serial channel. Devices of such kind will become predominant in an Internet of Things, but due to power or other constraints they may never be able to support even simple network protocols such as IP. Instead, such sensors will be connected to some gateway device that is able to read the data collected by the sensor (directly or after further processing) and provide it to a client via some high-level protocol [4]. In these environments, interoperability between systems is only needed at the top level and it is sufficient and also more efficient if interfaces and protocols are tailored to the purpose within such a hierarchy. Therefore, different capabilities of devices have to be considered as *scalability* and support for *hierarchical* structures are a key requirement. These hierarchies do not only apply to communication but as well to command structures, as normally the only purpose of the majority of subordinate devices is to act on command from superior devices.

3 The KINGDOMS – A Model Derived from Society

Analogies are often used as starting points to make a model easy to understand and to benefit from transforming properties from the analogy into the model and to learn from this analogy. Such models usually try to manage system complexity and handle emergent characteristics by design principles derived from nature such as decentralized control and self organization. In the case of computing models, they are frequently derived from analogies to fields such as physics, biology and sociology [2, 7]. Regarding the previously described requirements, the decision, which field is best suited for our purpose seems to be evident. While properties such as an adjustable level of abstraction or hierarchical structures can be found in physical models (e.g., the solar system) or in biological models (organisms consist of sub-organisms at different levels of complexity), the issue of liability is missing – and liability is one of our major concerns. Liability is a societal concept and, therefore, societal models seem to be suitable to serve as an analogy for a computing model subject to the given requirements.

3.1 Looking for an Appropriate Societal Model

From human history and daily life we know several societal structures considering humans and their relations to each other. They scale from small entities such as families or workgroups up to the whole world with over 6 billion people. It is quite obvious that in a societal analogy to modern computing, humans could be equivalent to computing devices. The following paragraphs cover some concepts present in societal models and discuss their applicability to our problem domain. This helps sorting out a lot of models.

Equality of subjects. Many modern societies obey human rights and therefore assume that all humans are equal. Although this is one of the most important and basic issues in modern societies, equality of subjects does not translate well into computing environments. Computing devices are unequal in nearly all measures such as computation power, communication capabilities, physical capabilities, interoperation capabilities, mobility or freedom of choice.

Advancement. Analyzing human societies there are also structures where people are not equal with respect to some aspects such as authority or freedom to make decisions. For example, each viable government, army or company have well defined command structures giving some people the right to do (or decide) more than others. Models for such structures perfectly meet our requirement of supporting hierarchies. However, they introduce an undesirable problem: One of the powerful driving forces inside such structures is the strive for, e.g., career advancement, meaning that a person starts at a certain level and has the possibility to move upwards in hierarchy. This is not appropriate for computing environments as a machine normally serves some purpose that has been defined during construction and it will most likely not move to some other level: A sensor that measures the speed of a car’s engine and that reports to the motor control unit will never become an engine controller (unless it is upgraded). Translating it into the world of societal models, a machine corresponds to a person whose position is defined by birth.

Ownership. Related to predefined positions and rights, is the issue of ownership. It leads back into medieval, ancient or even older times where a human could be owned by another human. Even if it is not true in a direct sense that one machine belongs to another machine, it brings forth an adequate concept for the issue of liability: Each computing device is under the control of a human or a legal entity being liable for that device. The power of disposal results from ownership, rental agreement or other transactions. This approach of including liability into the model fits the social analogy: Everybody is responsible for what his/her belongings and assets are affecting.

From this discussion follows that *feudalism* might be the answer to our search for a societal model as an appropriate model to structure modern computing environments.

In feudalism, mainly monarchs (kings, queens, emperors, sheiks, sultans; from now on referred to as a king) owned all the land including citizens and they were able to do whatever they liked – within their kingdom. However, to the outside world (i.e., other kingdoms) they were responsible for their kingdom and should obey rules (some of them violated them) imposed on them by “international” relationships, contracts and moral authorities like the pope. In order to handle complexity, kings applied a “divide and conquer approach” as they were not able to manage everything on their own: They devolved fiefdoms to a set of lords (magnates, barons, pashas) with the task to administer a region or to defend a coast, etc. The lords applied the same scheme at multiple levels. Individual rights and freedom of choice decreased down the hierarchy ending with bond-slaves who had no rights at all. Adopting the feudal structure we define our model for modern computing environments: the KINGDOMS.

3.2 Laws and Rules of the KINGDOMS

In the KINGDOMS, humans or legal entities (in the traditional judicial sense) are kings of their kingdoms. All devices, that are owned by them in the human world are *citizens* of their kingdom. Within their kingdoms, kings may issue instructions for the devices on their own will but they are, on the other hand, responsible for anything their devices do to the outside world.

Law and order. As it is the case with kingdoms in the real world, laws (rules) are defined inside each monarchy: The kings alone decide what is allowed and what is not permitted. Furthermore, kings can set the level of detail in which information is presented by the user interface. Some kings do not want to be disturbed by technical details while others want to know exactly what happens within their kingdom. The same holds for the level of autonomy that is granted to each device. Some kings may want to grant fine-grained rights to every single device within their kingdom while others simply want their tasks to be performed without dealing with risks and security issues. That is the way the KINGDOMS satisfy the requirement of an adjustable level of abstraction.

The devices of a kingdom have to blindly follow the rules of their king (their owner) and need not to reason about the consequences of their actions. This implies that, while obeying the rules of their king, devices may violate the law applying to their owner (their king). The human beings or legal entities who are kings of their kingdoms are responsible for their kingdoms and are, therefore, also liable for the consequences of actions performed by their technical devices. The kings have to ensure that the rules enforced within their kingdoms are reflecting the laws that are valid for the kings themselves. Additionally, it also implies that humans have the freedom to decide intentionally not to follow the laws applicable to them – and they have to live with the consequences which result from this decision. While this may seem dangerous, this mechanism is a strict enforcement of the requirement that humans have to be in charge: The purpose of computing systems is not to control humans, but vice versa to serve their needs. Another benefit of this approach is that interactions between humans are out of the model’s scope as they should be handled by the real world’s judicial system.

Capabilities of citizens. Within the KINGDOMS, technical devices have at least one out of four capabilities, one mandatory and three optional ones:

- The only mandatory capability is that citizens (computers and other devices of a kingdom) are able to *interact with* at least one other *citizen* of the kingdom including the king.
- The first optional capability includes direct *interaction with human,* i.e., they provide a user interface. Not surprisingly, it is used for task assignment and result reception. Additionally, the interface is used to specify rules and capabilities for the devices of the kingdom.
- The second optional capability is to *interact with the physical world,* i.e., they are sensors or actuators.
- The third optional capability of citizens is to *establish connections to citizens of other kingdoms*, if allowed by the rules of the king.

A general concept within the KINGDOMS is that tasks can either be performed by a device itself, or by distributing work further to other devices following a hierarchical or a peer-to-peer paradigm. To clarify this concept, two examples are provided. First, think of a temperature sensor. Such a device only needs to communicate with an appropriate controller to transmit the temperature data. The communication will be tailored to be small and energy efficient for this particular purpose, e.g., the sensor device has to be capable to communicate only with its controller using a dedicated cable and a proprietary protocol. There is no need for a universal communication protocol, e.g., based on Web Services. The world of Internet of Things should be open and, if possible unrestricted, in this respect. In analogy to a societal model, such dependency is similar to the interactions between a superior and its subordinates. The controller may then provide access to the sensor’s temperature readings by establishing a Web Service that is accessible from within the kingdom or even from the outside world. Such a structure provides scalability and interoperability while devices need only to support interfaces at the complexity level required to efficiently accomplish their tasks (see Fig. 1).

As second example, consider a desktop PC that provides several capabilities: It has a user interface to interact with a human, it is capable to perform certain jobs by itself (e.g., playing music), it can communicate with other devices of the kingdom, e.g., a printer and it can also establish connections to other kingdoms, e.g., connect to a web server owned by another company in order to display an HTML page.

Fulfilling the king’s tasks. As stated above, a king issues orders that should be completed by the citizens of the kingdom. The order is issued to one of the interface-providing devices. There are two ways to process the order: it can be performed internally, within the kingdom, or a device can request services that are provided by other kingdoms using “international” contracts. All task processing by devices has to obey the rules and preferences that have been issued by the king. For example, a king can establish the rule that all international relations have to pass through one dedicated device of the kingdom – a “firewall”.

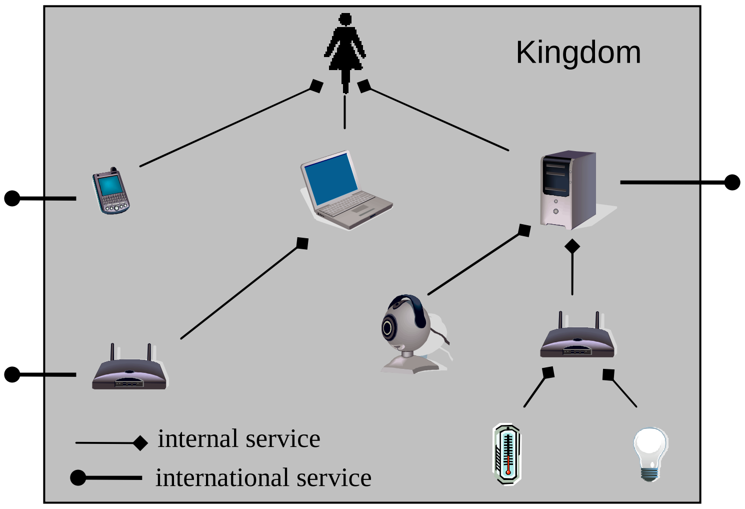

Fig. 1. Internal and international services within a hierarchically structured kingdom

International relations. The KINGDOMS model allows that a human configures its kingdom in a way such that its devices are not permitted to contact any other kingdom. However, many tasks can only be performed using services outside the kingdom. The usage of the internet requires international relations. Obviously, there are four types of international relations: Human-to-human, human-to-machine, machine-to-human and machine-to-machine. The first one relates to direct interaction of kings and is covered in a human society. Human-to-machine interaction addresses the scenario of a human using a device belonging to someone else, e.g., using the desktop PC of a friend. Therefore, it must be assumed that the device has a user interface and there is a human-to-human contract by which usage of devices is regulated. The next is machine-to-human interface where machine can convey (e.g., display) computation results or measurements to a human, a king of another kingdom and it is covered by a contractual agreement between kingdoms. The last type of international relations addresses machine-to-machine cooperation. In such a cooperation, the machines have to be set up automatically upon request, conforming to the rules of both kingdoms and acknowledging their requirements. For example, the human requester has to allow the devices of his/her kingdom to establish an international interaction to the service provider and if there are constraints regarding the service level, it has to be acknowledged that the requested level can be delivered by the service provider. Additionally, the service providing device must check with the rules of its king whether the service requester is allowed to use the service and how, e.g., payment is handled. Establishing international relations may as well include third-party services such as authentication, trust centers, reputation systems, ontology services, translation services, etc.

The key issue of international cooperation is to establish a *person-to-person chain of liability*: Starting from one user requesting a service, a chain of liability is established via devices of the requester's kingdom, via international contracts to the "king" of the service providing devices (see Fig. 2).

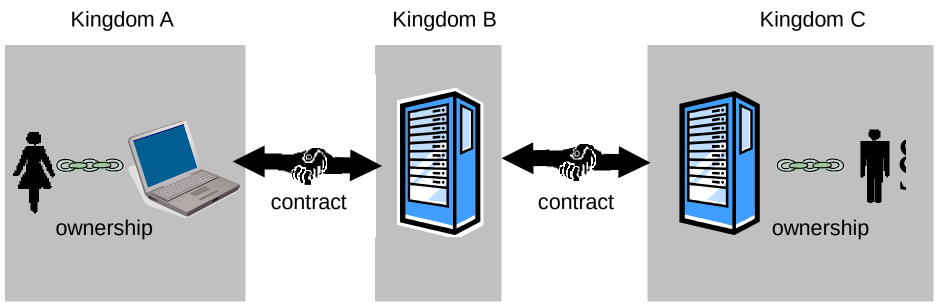

Fig. 2. Person-to-person liability chain

The KINGDOMS build on service oriented architectures to establish international relations. As in the real world, such interactions are subject to *contracts* describing the conditions of service usage. It is a matter of liability and of dependability that contracts have to include non-functional aspects such as response times or payment modalities.

Another aspect of international relations is interoperability. As already stated, we do not request that all devices can interoperate with all others. This means that we do not demand a single definition of a service architecture including a limited set of languages, transport protocols, etc. Instead, we explicitly allow heterogeneity – as long as there is a translator service. This approach also solves the issue of legacy system integration and allows co-existence of different architectures within the KINGDOMS. It is our belief that this is the most promising approach to establish architectures such as the KINGDOMS on a global scale. More homogeneous approaches, forcing everybody to use the same standard, haven't yet achieved global interoperability.

In summary, we see that from the societal model of feudalism, a straightforward solution emerges that has the potential to meet today's challenges of mobile, adaptive, dependable and ubiquitous computing translating into human-centric and ambient computing. The KINGDOMS master complexity by structuring along the borders of ownership and by enforcing contract-based service-oriented architectures to couple devices of different owners, which establishes a network of interconnected person-to-person liability dependencies (see Fig. 3).

4 Key Issues for the KINGDOMS to Become Reality

In order to realize the world of the KINGDOMS, several issues have to be addressed that are of key importance. Some issues can be solved by technologies or concepts that are already available, some issues have been addressed in research but are not yet available for implementation on a global scale and other issues are still open – they need to be tackled in the future.

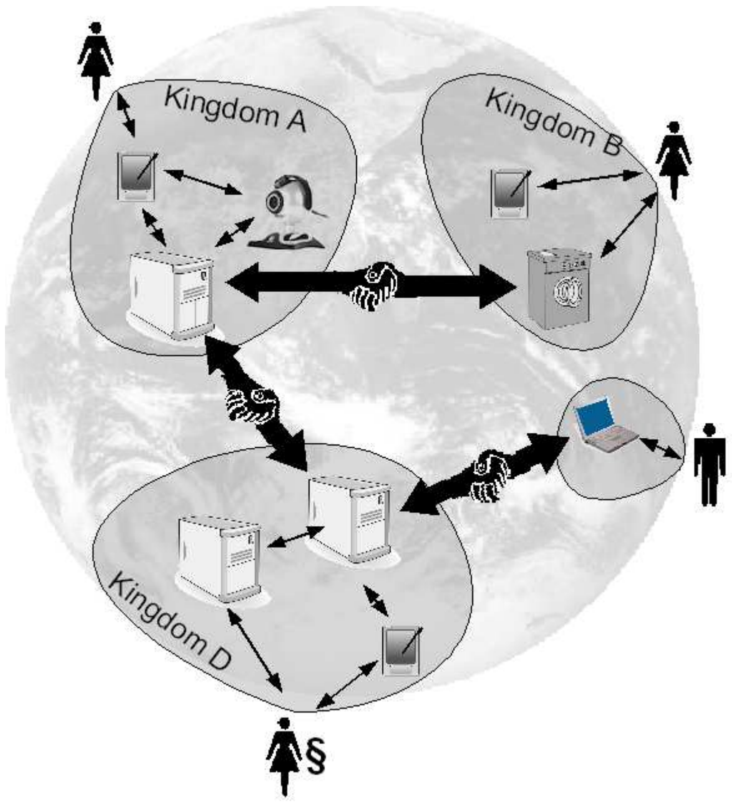

Fig. 3. The world of KINGDOMS with multiple interactions and liability dependencies

4.1 Citizenship

Since the KINGDOMS are derived from a societal model, one of the major issues is a precise definition of citizenship. The KINGDOMS build on *ownership*: each device belongs exactly to only one human or legal entity in terms of accepted law. Additionally, as the aim of the KINGDOMS is to organize technical devices such that they are able to perform tasks issued by humans, another characteristic of KINGDOMS citizens is a *direct or indirect interaction with the king*, which is a direct implementation for the requirement of *scalability* and *support for hierarchical structures*. There are no additional requirements as long as international interactions are not concerned. However, if some device is aimed to set up connections to other kingdoms, it has to possess additional features: it has to provide some *communication technology* that enables interaction to devices of other kingdoms plus it has to *ensure the chain of liability*. The main issues that are associated with liability insurance are *identity management* and *contract acknowledgement*.

Communication Technology. We do intentionally not restrict devices of the KINGDOMS to use prespecified communication techniques as this would inhibit their wide-spread use. Communication techniques such as Web Services, CORBA, etc. are only means to collaboratively solve goals specified by humans. Especially, regarding the requirement of *mobility* and *adaptivity*, a restriction of communication techniques could be questionable in this case. The key issue is insurance of the chain of liability, which specifically consists of identity management and contract acknowledgement.

Identity Management. In order to establish the chain of liability over the borders of several kingdoms, each link in the chain needs to provide an identity. Identity in this context constitutes a *mapping from an ID to a human or legal entity*. Again, the architecture of the KINGDOMS does not install one dedicated way to handle or verify this mapping. Instead, existing trust-centers, authentication services or even reputation systems can be used to “verify” the mapping. From this follows that different levels of trust in the validity of identities are acceptable in the KINGDOMS as the required level depends on the purpose of cooperation. However, the maxim of liability always holds: both kings (being legal entities in the human judicial sense) of any international cooperation are always liable. Therefore, the requested level of trust increases with the criticality or monetary value of the transactions. Nevertheless, open issues regarding management of identities remain. For example, it is not yet clear how the mapping to a legal entity can be represented. Furthermore, a kingdom might use several identities pointing to the same legal entity depending on the type of cooperation and the technology that is used. Identity management also has to address the requirements of mobility and adaptivity.

Contract Acknowledgement. The second issue that is of key importance when establishing the chain of liability is that both partners of any international relation have to acknowledge what exactly has to be delivered and what are the requirements / constraints of delivery. In service-oriented computing such an acknowledgement is called *contract*. Some work has been published proposing ways to specify such contracts in a formal and machine readable way. See, e.g., [3, 5]. Especially, regarding the requirement of *dependability* it is not a trivial task to represent non-functional properties such as real-time constraints or service reliability. This is also closely related to the issue of service semantic representation.

Another open issue is how contract violations can be handled. Of course, the chain of liability ensures that the deceived partner of a cooperation can bring the other partner to trial. The question is how to avoid future fraud. One approach is the wide-spread use of reputation systems another method (at least for service providers) is to minimize the risk of contract violation by consequently demanding dependability (including security).

4.2 Managing Complexity

One of the driving forces for the development of a new paradigm for computing is the rapidly increasing complexity of computer systems that have already exceeded the level that can be handled by majority of human beings. Although the KINGDOMS provide a general structure to manage complexity, some issues still have to be addressed.

Specifying goals. Devices of a kingdom should serve the goals that have been issued by their king. However, it is not clear how these goals can be specified in a manner that can be processed automatically and correctly by the devices, including payment issues and non-functional properties.

Fulfilling the goals. In future, there might be millions of kingdoms comprising billions of devices offering trillions of services. How may a device manage to fulfill the goal issued by its king? In service oriented architectures, there are two techniques that are closely coupled. *Service composition* tries to assemble available services as building blocks in order to fulfill some formally specified complex goal, while *service decomposition* tries to break down the complex goal into parts that can be delivered by available services. However, in order to compose or decompose, the manifold of available services has to be explored. This issue is known as *service discovery*, which should be mainly handled by service directories but an efficient methodservice discovery on global scale still needs to be developed.

Defining semantics. People quite frequently tend to use different terms for the same item or concept. The KINGDOMS model does not provide specific means to avoid such confusion. However, the architecture of the KINGDOMS allows and encourages establishing services that solve these misunderstandings. One way to do it is to use *ontology services* that map terms and concepts. This is equally important for semantic descriptions of services as well as for the specification of non-functional properties.

On the level of communication the same problem appears: There will be the need for cooperation between devices that do not share a single communication protocol. In order to address this problem, *translation* or *gateway services* need to be established.

Specifying rules and permissions. Last but not least, the requirements for our vision on future computing scenarios include *total control* and *adjustable levels of complexity* for each king. It is not an exaggeration to assume that there will be a need for individuals to control hundreds of devices as companies already manage thousands of servers and devices, today. New concepts have to be developed on how the manifold of different devices can be managed efficiently. For example, new interfaces are needed that allow for an efficient specification of rules and permissions. This topic also includes issues of human-centric user interfaces and interaction.

5 Examples

To illustrate the idea of KINGDOMS we present two examples:

5.1 Example 1: Webcam-supported Weather Service

Cheap but powerful hardware together with decreasing costs for networking leads to increased usage of networking to connect various devices, e.g., network attached cameras, sensors and actuators, household appliances or infotainment system in cars to name a few. Theoretically, it would be possible to use such networked devices to establish a lot of useful services at very small additional cost. For example, a weather service allowing users to figure out how the weather is in a city can easily be composed by accessing available networked cameras and temperature sensors in that area. Some image processing may make it even possible to include wind and rain. But, on the other side, this kind of basic services (cameras and temperature sensors) can easily be abused to, e.g., illegally monitor people. Furthermore, as such a service is usually based on data delivered by devices belonging to several third parties, it is difficult to guarantee availability or quality of service.

KINGDOMS offer a simple and effective solution: Each camera and each temperature sensor is part of a kingdom (of the corresponding owner of the physical device). Therefore, this owner can establish rules for interaction with other devices inside and outside the kingdom (outside communication may use other devices if direct interaction is forbidden). For example, he or she restricts the camera device in a way that it delivers the complete images only to a small set of trusted partners while all others will only get pictures that do not include humans (or are manipulated in a way that it is not possible to track humans). Furthermore, he or she can define prices for the offered service: For example, a picture that was taken one hour ago is for free while a picture that is not older than one minute costs one Micro-Euro.

By using these weather data and picture services a service provider can implement a picture-enabled weather service that is served by one of his devices (having the right to collect data using international contracts and to answer requests by others). This device finds appropriate data sources (i.e., temperature data, pictures, etc.) by service discovery and negotiates contracts with service providers. These contracts are based on the rules of the owner and the capabilities of the service provider. They can also define quality of service agreements which reach from “no guarantee at all” to “guaranteed delivery together with punishment in case of failure”. In order to allow automatic negotiation, the weather service provider needs to define that define both the cost of the service that is provided as well as the amount of money that is needed to deliver the service. In the case of the weather service this could be something like “average weather from the last two hours” for free to “accurate weather in the city based on data that is no more than one minute old” for 50 Micro-Euro. In order to ensure a reliable service, it has to be assured that an appropriate number of data services is used. Furthermore, in order to prevent the weather service from being in deficit, it has to be requested that cooperation with data providers is achieved at maximum costs, e.g., one Micro-Euro per request.

All partners involved in these interactions are interested in fulfilling the requested quality of service in order to avoid penalty (which is paying a fine or loosing reputation in some reputation system provided by another party). Therefore, they set up their services in a dependable way themselves, or by composing other dependable services such that they do not violate dependability constraints (composability with respect to dependability, see [8]). Such a compositional approach is again based on the chain of liability. In case of failure, the party violating its contract by not delivering one of the basic services can be penalized, and further on along the chain of liability.

5.2 Example 2: Business Process Interactions

Services offer a good mean to perform interactions with other partners in an ad-hoc manner without contracts that are established a-priori by persons. Ad-hoc contracts work well for private purposes such as buying a book or even booking a trip together with an airplane reservation and a car rental. However, there are no guarantees. Therefore, business service interactions that include the possibility of high (physical or financial) damage are not handled that way. Instead, contracts are coarse-grained defining a relation between the partners for a long period of time and a lot of individual service invocations. While it delivers some guarantees it is inefficient and inflexible. On the other hand, today’s solutions such as Web Services offer flexibility but no guarantees.

KINGDOMS provide a fine-grained solution: Each company is king of all its devices and defines rules for interactions. These rules include preconditions for any kind of international contracts such as “partner has to have a reputation higher than X in reputation service Y” or “in case of failure partner has to pay twice the damage” or something similar. As every interaction belongs to a chain of liability that is established by the contracts, it is always possible to find the person or legal entity a service provider belongs to (fundamental principle of ownership in KINGDOMS). Reputation systems and authorization centers help to avoid services offered by cheaters similar to the way it is done in human interactions (a company that is successful on the market since 20 years will not take the risk of cheating while a company founded yesterday has less to lose). Like in the real world, selection is – among various factors – done according to the risk.

Therefore, it is possible to negotiate service contracts for nearly any interaction in a fine-grained manner without the need of human interaction and without sacrificing safety of an established contract. Furthermore, this eliminates the dependency on predefined partners because contract negotiation is possible for each new service invocation which allows changing a partner from one request to the next. These contracts are based on specification of quality of service properties such as adaptivity and dependability that have to be fulfilled by service partners. The flexibility offered by this possibility also allows competition on each single service request and not only on a coarse-grained scale (company X decides to cooperate with partner Y for the next two years).

However, the reality in the recent years showed that this kind of interaction is exaggerated in science. In fact, there is only little demand to operate this way, as most companies want to retain control. This kind of control is supported in the KINGDOMS model by appropriate rules for interaction: citizens have to present a king with a pre-processed information, but the decision itself lies with the king only.

6 Summary and Conclusions

We presented a vision of KINGDOMS paradigm, oriented towards personal as well as business (enterprise) use. We have structured the two examples such that they reflect both use-case domains. Although the reasoning used throughout the paper reflects personal use of small-scale, portable or embedded devices in the Internet-of-Things context, the same principles, namely managing complexity through societal organization of software and hardware elements in service of a human user, can be applied to the enterprise computing problems, as described in the second example.

Therefore, the proposed vision can be extended to scenarios where citizens are not only devices or pieces of software, but complex compositions and interactions thereof, forming a service ecosystem where citizens are being born, live, cooperate and eventually die (or are retired). The process need not be static as, apart from the simplest devices, all citizens undergo many changes in their lives (upgrades, patches, versioning), thus making their contracts dynamic and their position within a KINGDOMS ecosystem susceptible to change. The KINGDOMS infrastructure should manage these changes in a way that is seamless and continuous, and if required, transparent to the end user.

Finally, the openness of KINGDOMS paradigm should pave the road to solving one of the biggest issues in today’s enterprise computing: that of seamless interoperability between heterogeneous distributed systems. A potential way to marry the two seemingly conflicting requirements (openness and non-restrictiveness with numerous and disparate concrete technologies) may be the model-driven approach, where properties of KINGDOMS citizens are described in a platform-agnostic way using platform-independent models which are bound to platform-specific objects at runtime using platform- and application-specific connectors. The king and other citizens need not be aware of the platform-specific issues.

The quest for faster, cheaper, bigger and smaller computers continues at incredible and unstoppable pace while the very actor, a human, is left behind, helplessly overwhelmed by complexity and overrun by frequently unwanted e-mails today and services tomorrow. It is high time for reassessment of such activities and finding the way to put a human back in charge. The concepts of ownership, liability, economy and trust are of an essence and so it is with the KINGDOMS.

References

1. P. Horn, *Autonomic computing: IBM's perspective on the state of information technology*, Oct. 2001.
2. L. Kleinrock, *Nomadic computing – an opportunity*, ACM SIGCOMM Comput. Commun. Rev., Vol 25(1), 1995.
3. B. Meyer, *Contracts for Components*, Software Development Magazine, July 2000.
4. N. Milanovic, J. Richling, M. Malek, *Lightweight Services for Embedded Systems*, Proceedings of the 2nd IEEE Workshop on Software Technologies for Embedded and Ubiquitous Computing Systems (WSTFEUS 2004), Vienna, Austria, 2004.
5. N. Milanovic, *Contract-based Web Services Composition*, Dissertation at Humboldt University Berlin, 2006.
6. M. Malek, *CobWeb: Challenge of Billions in the Web*, Proceedings of Future Directions in Distributed Computing, Bertinoro, Italy, June 3-7, 2002.
7. M. Malek, *Introduction to NOMADS*, Computing Frontiers, ACM Conference Proceedings, 2004, Ischia, Italy, April 2004.
8. J. Richling, *Komponierbarkeit eingebetteter Echtzeitsysteme*, Cuvillier-Verlag, Göttingen, 2006.