Agents & Artifacts at the Knowledge Level

Samuele Burattini[1], Andrei Ciortea[2], Meshua Galassi[1], and Alessandro Ricci[1]

[1] Dipartimento di Informatica - Scienza e Ingegneria,
Alma Mater Studiorum, Università di Bologna, Cesena Campus, Italy
meshua.galassi@studio.unibo.it, {samuele.burattini|a.ricci}@unibo.it
[2] School of Computer Science, University of St.Gallen, Switzerland
andrei.ciortea@unisg.ch

Abstract. In this contribution, we propose an extension of the Knowledge Level as introduced by Newell in the A.I. context and refined by Jennings in agent-based software engineering to include also the environment as a first-class analysis/design dimension. We revisit and refine the Agents & Artifacts (A&A) conceptual model to be at the Knowledge Level by explicitly introducing a semantic layer based on Knowledge Graphs, and we discuss the benefits with some practical examples.

Keywords: Knowledge Level · Agent-Oriented Software Engineering · Agents & Artifacts · Knowledge Graphs · Semantic Web · CArtAgO

1 Introduction

Four decades ago, Allen Newell introduced the *knowledge level* analysis to characterise intelligent agents as knowledge-based systems, abstracting from application-specific details and implementations [14,16]. According to this characterisation, a computational system can be viewed across multiple levels of abstraction — a hierarchy of computer system descriptions[3]. The Knowledge Level is just another level within that same hierarchy: a way to describe the behaviour of (intelligent) systems with wide-ranging capabilities, where capability is defined in terms of having "knowledge" and behaving in light of it. The key feature of the Knowledge Level from a software engineering viewpoint is that it abstracts completely from the internal processing and the internal representation: all that is left is the content of the representation and the goals towards which that content will be used.

The concept has become a keystone in agent-oriented software engineering [10], along with the very similar characterisation introduced, in the same period, by Dennett with the *intentional stance* [6] — effectively setting the level of abstraction that we expect when modelling and designing a software component as an intelligent agent. Two decades ago, the concept was further extended by Jennings in the context of agent-oriented software engineering to also include the social/organisational dimensions [10][4].

[3] Appendix A reports the levels as depicted by Newell in [15].
[4] Appendix B reports a description of the knowledge level and the social level as depicted by Jennings in [10].

After two decades, we further extend this important conceptual framework with a missing element that proved to be, both in the case of humans and in Agent-Oriented Software Engineering (AOSE), an important dimension for analysing and designing systems: the environment. This extension aims at providing a uniform level of abstraction to describe both the goal-oriented behaviour of software agents and the environment they can exploit to achieve such goals by discovering, manipulating and creating resources and services.

Accordingly, the first contribution of this paper is about framing and discussing the role of the environment as a first-class design abstraction at the Knowledge Level. To achieve that, we look at ways in which knowledge about the real world is currently being represented and shared in other kinds of systems, following the evolution of the digital transformation of different domains using the Semantic Web — for instance, in the Web of Things[5]. In Section 2, we discuss this point, using the Agents & Artifacts (A&A) conceptual modelling [17], which was implicitly conceived to be at the Knowledge Level.

The A&A meta-model was conceived informally, without identifying a clear connection at the knowledge level with domains. Accordingly, as a second core contribution of this paper, in Section 3 we discuss a refinement and extension of the A&A meta-model to be fully effective for supporting the Knowledge Level, and in Section 4 we briefly describe a first extension of the CArtAgO framework [19] implementing it. We conclude the paper with a brief road-map for future work in Section 5.

2 Enriching the Knowledge Level with Artifact-Based Environments

As remarked in AOSE literature [22,23,21], the environment can be used as a first-class abstraction when designing and programming agent-based systems. In particular, it can be used for encapsulating and providing functionalities to agents at different levels [22]: a *basic level*, to enable direct access to the deployment context; an *abstraction level*, providing agents with an abstraction level that shields low-level details of the deployment context — as well as other resources in the system; an *interaction-mediation level*, providing agents an interaction-mediation level to support mediated interaction in the environment; a *reflective level*, providing a reflective interface to the functionality supported by the environment, enabling agents to modify the functional behavior of the environment [18].

At the Knowledge Level, this accounts for enriching Newell's and Jennings's conceptual framework to include the environment at the same level of abstraction with agents and agent organisations, modelling the open of resources and tools as first-class *artifacts* — as introduced by A&A [17] — that agents and organisations may build, use, and share in order to accomplish their individual and social goals. The concept of artifact and the overall A&A conceptual model were

[5] See the W3C Web of Things: https://www.w3.org/TR/wot-architecture/

mainly inspired by Activity Theory [20,18] and Distributed Cognition [9], which are prominent conceptual frameworks and theories that investigated the role the environment for supporting human activities at large (cognition, reasoning, learning, etc.).

From an engineering point of view, artifacts model those parts of the system that are not effectively described as goal-oriented knowledge-based systems, acting to attain goals, but more as *function-oriented* or *service-oriented* components used by the goal-oriented ones. Like artifacts designed for humans, a key feature of artifacts designed for agents is given by the affordances that they provide to enable their (effective) use by agents, defining the underlying interaction model at the proper level of abstraction. In the A&A model, these affordances are based on observable properties, operations (actions, from the agents' viewpoint), and observable events [19]. Another concept is the artifact *manual*, i.e. a document describing what are the functionalities of an artifact and how to interact with it.

Nevertheless, in order to be exploited by intelligent agents at the Knowledge Level, artifacts should be designed and conceived at the same level. In the next section, as a core contribution of this paper, we discuss how the A&A conceptual model can be further refined and extended for this purpose.

3 Artifact-Based Environments at the Knowledge Level

Using A&A at the Knowledge Level means that the artifact-based environment is meant to be used by intelligent agents to perceive and act upon domain entities — possibly representing assets in the real world — as well as to create and exploit resources and tools that are instrumental for attaining their goals. For this purpose, we envision a further refinement or characterisation of the A&A model in which:

- Artifacts should be *semantically ground to domain entities at the Knowledge Level*: their affordances and their manuals should be described at that same level of abstraction;
- *Relationships among entities* at the domain level should be explicitly represented and reified at the artifact level so that agents can reason about them;
- *Workspaces* – another main concept in A&A - can be used to define boundaries for agent activities, i.e. contexts where one or multiple agents can create and share one or multiple artifacts, as well as logical contexts that share the same domain vocabulary to describe the entities within them.

To support this refinement, we introduce an explicit semantic layer for A&A, not bound to any specific domain but expressive enough to support the design of artifact-based environments eventually involving multiple domains and ontologies. The semantic layer is based on the concept of *knowledge graph* [8]. A Knowledge Graph (KG) is *"a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose*

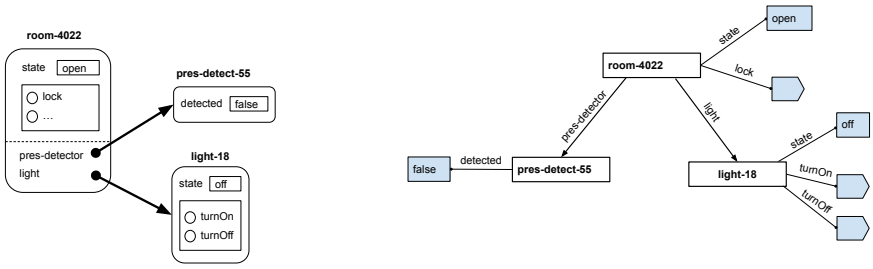

Fig. 1. Smart room scenario. On the left: the three artifacts, representing a room, a presence sensor, and a light. On the right: the corresponding Knowledge Graph.

edges represent relationships between these entities" [8]. An artifact-based environment can then be mapped into a KG where each artifact has a corresponding node in the graph — representing an entity of interest at the domain level. Following the A&A meta-model, artifacts feature observable properties, actions, and observable events. These are represented in the KG by (dynamic) data properties of the corresponding entity, i.e. as a relationship between the entity and a typed value. To capture relations among entities (edges between nodes) we extend the artifact meta-model with the concept of (observable) relationship.

As an example, let's consider a toy smart room scenario in an Internet of Things (IoT) context (see Figure 1). The scenario includes a room, a presence detector, and a light as domain entities. The figure shows the three artifacts modelling this environment (on the left), and the corresponding KG (on the right). Let's consider a very simple intelligent agent, situated in this environment, designed to accomplish an energy-saving goal by turning off the light when no one is in the room, and turning it on if someone enters (and the light is off). In order to accomplish its goal — defined at the Knowledge Level — the agent can exploit the artifact-based environment, whose semantics are defined by the corresponding KG. In particular, the agent may continuously observe the presence detector and turn on/off the light by acting on the lamp. For this purpose, the agent may start observing the presence detector by doing a *focus* on the corresponding artifact. In A&A, by issuing a focus on some artifact Ar, an agent starts perceiving the observable state of Ar and the observable events generated by Ar, including those related to changes about observable properties. Then, as soon as it perceives that someone has been detected, e.g. by perceiving an observable event generated by the artifact representing the presence detector, the agent may turn on the light by acting on the corresponding artifact — if the light was not already on (this state can be perceived by the agent by observing the lamp as well).

The semantic extension based on KG allows to substantially empower the expressiveness of the basic capabilities provided by artifact-based environments.

3.1 Querying and Observing at the Knowledge Level

In A&A, an agent has a primitive action read-obs-property(Ar,P) to retrieve the current value of an observable property P of an artifact Ar. By mapping an artifact-based environment into a KG view it is possible to make more expressive queries, involving graphs of entities (artifacts).

In this paper, we consider RDF[6] as a standardised data model for representing KGs. A KG can be represented as an RDF graph, that is a set of triples `(subject, predicate, object)` where each triple represents a property or relationship of the `subject` entity. For instance, in our case, the `subject` could be an artifact identified by a uniform identifier (e.g., a URI[7] or an IRI[8]). The `predicate` could describe a data property – as triples where the identifier of the property is used as predicate – or a relationship to another artifact – as triples where the identifier of the relationship is used as predicate and the object is the identifier of another artifact. Given an RDF representation of a KG, the graph can then be queried using SPARQL[9].

Accordingly, any artifact-based environment extended at the Knowledge Level can then be described in RDF and queried by agents using SPARQL. For example, in the toy scenario suppose that the room may have multiple lights referred to by the light relationship. An agent can query the environment to find out which lights in the room are on:

```
SELECT ?light
WHERE { "room-4022" :light ?light .
        ?light :state "on" .}
```

Besides querying, continuous observation can also be empowered. In particular, we can introduce and exploit a variant focus-all of the focus primitive action so that by issuing a focus-all on an artifact Ar, an agent may perceive the observable state and future observable events not only of the specific artifact but of all artifacts linked to that artifact, according to the relationships in place. In the toy scenario, for instance, a focus-all on room-4022 would imply to start observing the room, as well as the presence detector and the light.

3.2 Semantic-driven Creation of Artifacts in Workspaces

Framing an artifact-based environment at the Knowledge Level implies that the dynamic construction and extension of the environment should be characterised at that level as well. In particular, the dynamic creation of an artifact, possibly linked to or linked by some other artifacts, should be a possibility provided by the environment — grounded at the domain/semantic level.

For instance, extending the smart room example introduced above, a personal assistant agent could detect that its user has entered a room. Accordingly, it

[6] https://www.w3.org/TR/rdf11-concepts/
[7] https://www.rfc-editor.org/rfc/rfc3986
[8] https://www.rfc-editor.org/rfc/rfc3987
[9] https://www.w3.org/TR/sparql11-query/

might want to make some data about the user available to other agents (e.g. a room manager agent) by creating a UserProfile artifact in the smart room workspace, linked by the room artifact by means of a user relationship, to expose data about his desired light level.

In basic A&A, a primitive action make-artifact is provided for instantiating a new artifact by specifying the template and construction parameters [19]. The action corresponds to an operation provided by a pre-defined *workspace artifact* available in each workspace, providing basic capabilities to work inside that workspace (to create and dispose of artifacts, to focus on artifacts, etc.).

Raising A&A at the Knowledge Level implies to revise this mechanism in order to allow for *driving and constraining artifact creation at the domain/semantic level*. Accordingly, each workspace, as a context of semantically-driven agents' activities, may be initially configured — at workspace creation time — with an artifact representing from the agent point of view the *single entry point* of the context, providing the initial set of actions to extend/develop it, according to the possibility defined for that context at the semantic level. In the smart room scenario, the `SmartRoom` artifact would function as an entry point, providing a notifyNewUser operation, creating a new UserProfile artifact and the relationship user linking to it.

4 Bringing CArtAgO at the Knowledge Level

To start exploring in practice the vision brought by this paper, we developed a semantic layer on top of the existing CArtAgO framework, which is the main reference implementation for the A&A meta-model and part of the JaCaMo [1] platform. For this first integration, we focused on generating a semantic description of the artifacts so that agents could exploit the resulting Knowledge Graph to query the environment. We considered examples with just one workspace with a centralised KG associated to it to start with.

The KG is empty at the beginning of the application. When instantiating new artifacts, they automatically add their own semantic description and generate an ontology based on the artifact class implementation. The implementation uses Apache Jena[10] framework and RDF triplestore wrapped in a `SemanticEnvironment` interface. The class `SemanticArtifact` extends the CArtAgO `Artifact` base class, adding to the base behaviour the automatic insertion and update of RDF triples to the KG when needed (e.g. when initialising the artifact, when updating observable properties, etc.).

Listing 1.1 shows how an artifact can be defined with the new API. The lightswitch artifact has a `pressed` observable property and a `controls` relationship with the light artifact it is controlling.

```
1 public class LightSwitchArtifact extends SemanticArtifact {
2
3   void init(boolean isPressed, String idConnection){
```

[10] https://jena.apache.org/

```
    super.init(this, this.getId().getName());
    defineObsProperty("pressed", "boolean", isPressed);
    defineRelationship("controls", idConnection);
  }
  @OPERATION void press() { setPress(true); }
  @OPERATION void release(){ setPress(false); }

  private void setPress(boolean p){
    updateValue(pressProperty, this.press);
  }
}
```

Listing 1.1. An example of how to use the `SemanticArtifact` API to define an artifact.

The corresponding Knowledge Graph will have the definition of the ontology, and of the instances of the artifacts in the environment. In Listing 1.2, an RDF serialisation of the knowledge graph with the instances of two artifacts is shown using Turtle syntax.

```
@prefix : <http://example.org/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

:lamp_0     a owl:NamedIndividual, :Lamp ;
            :stateOn false .

:lightSwitch_0 a owl:NamedIndividual, :LightSwitch ;
            :controls :lamp_0 ;
            :pressed false .
```

Listing 1.2. Knowledge Graph serialisation with a Lamp and a LightSwitch

Agents can then query the generated Knowledge Graph containing all the data about the environment exploiting the semantic layer to discover information about the available artifacts. (Listing 1.3).

```
+!findSwitch
<- query("SELECT ?lamp WHERE {?l rdf:type :Lamp}", R1);
  getValue(0, "l", R1, LampId);
  .concat("SELECT ?d WHERE { ?d :controls :",LampId,".}",Q);
  query(Q, R2);
  getValue(0, "d", R2, SwitchId);
  .println(SwitchId, " controls ", LampId).
```

Listing 1.3. A Jason agent plan performing SPARQL queries on the environment to find a Lamp and then the Switch connected to it.

5 The Road Ahead

In this paper we started crunching a vision extending A&A at the Knowledge Level, doing some first experiments using CArtAgO. The idea has been strongly influenced by existing work in literature about Hypermedia MAS [4,5], in which

A&A and artifact-based environments have been taken as a conceptual model to characterise agents situated on the Web, in a wide perspective including also Semantic Web and Web of Things. Besides Hypermedia MAS, related works include the wide literature in MAS and AOSE about integrating ontologies and Semantic Web technologies in agent/MAS languages and platforms [13,11,12,7,3].

This vision introduces challenges and open issues at different levels, to be tackled in future research efforts. In the following, we discuss three main ones:

Querying and observing graphs of artifacts: A main issue is about the atomicity and consistency of SPARQL queries involving dynamic graphs of artifacts, possibly evolving concurrently. Artifacts in an artifact-based environment may evolve concurrently, for instance, by means of actions performed by different agents. That is: each artifact is guaranteed to evolve atomically, but different artifacts may evolve concurrently. The question then is: what kind of consistency can an agent have by performing a SPARQL query over an evolving graph? In our first exploration, a simple solution is adopted based on workspaces, functioning as a context delimiting consistency. SPARQL queries are guaranteed to be atomic for the graph of artifacts that belong to the same workspace. Nevertheless, in the model proposed in this paper, artifacts in one workspace can link via relationships to artifacts in other workspaces — in a pure Linked Data spirit. This implies handling queries across workspaces.

Working with ontologies: An artifact-based environment at the Knowledge Level could concern entities belonging to different domains, possibly described at the semantic level using different ontologies. For this purpose, the Semantic Web provides a full stack of technologies in addition to RDF, such as RDF Schema (RDFS), the Web Ontology Language (OWL), or the Shapes Constraint Language (SHACL). A main exploration concerns then how to enrich the support for the Knowledge Level as introduced in this paper by considering the full spectrum of Semantic Web technologies. The abundant literature about integrating ontologies in agent/MAS design and programming (e.g. [13,11,12,7,3]) will be an important reference here.

Multi-Agent Oriented Programming at the Knowledge Level: In platforms like JaCaMo [2,1], the agent, environment, and organisation dimensions are integrated into a coherent and synergistic model. A main issue then is to preserve a coherent view about the Knowledge Level across the different dimensions. In particular, in JaCaMo the A&A conceptual model – implemented by CArtAgO – is integrated with the BDI model/architecture adopted for designing and programming agents in Jason. Accordingly, the A&A/CArtAgO extension is going to impact the way in which the knowledge about the environment is represented on the agent side, in terms of beliefs about artifacts' observable state and events, as well as the actions that can be performed on artifacts. Existing work around AgentSpeak-DL [13] –s integrating Description Logics for knowledge representation in AgentSpeak(L) — and JASDL [11] — combining BDI and Jason with Semantic Web Technologies – will be an important reference to consider for tackling this point.

References

1. Boissier, O., Bordini, R., Hubner, J., Ricci, A.: Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo. Intelligent Robotics and Autonomous Agents series, MIT Press (2020), https://books.google.it/books?id=GM_tDwAAQBAJ
2. Boissier, O., Bordini, R.H., Hübner, J.F., Ricci, A., Santi, A.: Multi-agent oriented programming with jacamo. Science of Computer Programming **78**(6), 747–761 (2013)
3. Chella, A., Lanza, F., Seidita, V.: Representing and developing knowledge using jason, cartago and owl. In: Workshop From Objects to Agents (2018)
4. Ciortea, A., Boissier, O., Ricci, A.: Engineering world-wide multi-agent systems with hypermedia. In: Engineering Multi-Agent Systems: 6th International Workshop, EMAS 2018, Stockholm, Sweden, July 14-15, 2018, Revised Selected Papers 6. pp. 285–301. Springer (2019)
5. Ciortea, A., Mayer, S., Gandon, F., Boissier, O., Ricci, A., Zimmermann, A.: A decade in hindsight: the missing bridge between multi-agent systems and the world wide web. In: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (2019)
6. Dennett, D.C.: The intentional stance. MIT press (1987)
7. Freitas, A., Schmidt, D., Panisson, A.R., Meneguzzi, F., Vieira, R., Bordini, R.H.: Knowledge-level integration for jacamo. In: Fifth International Workshop on Collaborative Agents – Research & Development, CARE for Intelligent Mobile Services (CARE) (2014)
8. Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., Melo, G.d., Gutierrez, C., Kirrane, S., Gayo, J.E.L., Navigli, R., Neumaier, S., et al.: Knowledge graphs. ACM Computing Surveys (CSUR) **54**(4), 1–37 (2021)
9. Hutchins, E.: Distributed cognition. International Encyclopedia of the Social and Behavioral Sciences. Elsevier Science **138**, 1–10 (2000)
10. Jennings, N.R.: On agent-based software engineering. Artif. Intell. **117**(2), 277–296 (mar 2000). https://doi.org/10.1016/S0004-3702(99)00107-1, https://doi.org/10.1016/S0004-3702(99)00107-1
11. Klapiscak, T., Bordini, R.H.: Jasdl: A practical programming approach combining agent and semantic web technologies. In: Declarative Agent Languages and Technologies VI: 6th International Workshop, DALT 2008, Estoril, Portugal, May 12, 2008, Revised Selected and Invited Papers 6. pp. 91–110. Springer (2009)
12. Mascardi, V., Ancona, D., Barbieri, M., Bordini, R.H., Ricci, A.: Cool-agentspeak: Endowing agentspeak-dl agents with plan exchange and ontology services. Web Intelligence and Agent Systems: An International Journal **12**(1), 83–107 (2014)
13. Moreira, A.F., Vieira, R., Bordini, R.H., Hübner, J.F.: Agent-oriented programming with underlying ontological reasoning. In: DALT. vol. 3904, pp. 155–170. Springer (2005)
14. Newell, A.: The knowledge level. Artif. Intell. **18**(1), 87–127 (jan 1982). https://doi.org/10.1016/0004-3702(82)90012-1, https://doi.org/10.1016/0004-3702(82)90012-1
15. Newell, A.: Unified Theories of Cognition. Harvard University Press, USA (1990)
16. Newell, A.: Reflections on the knowledge level. Artificial Intelligence **59**(1-2), 31–38 (1993)
17. Omicini, A., Ricci, A., Viroli, M.: Artifacts in the A&A meta-model for multi-agent systems. Autonomous Agents and Multi-Agent Systems **17**(3), 432–456

(dec 2008). https://doi.org/10.1007/s10458-008-9053-x, https://doi.org/10.1007/s10458-008-9053-x
18. Ricci, A., Omicini, A., Denti, E.: Activity theory as a framework for mas coordination. In: Engineering Societies in the Agents World III: Third International Workshop, ESAW 2002 Madrid, Spain, September 16–17, 2002 Revised Papers 3. pp. 96–110. Springer (2003)
19. Ricci, A., Piunti, M., Viroli, M.: Environment programming in multi-agent systems: An artifact-based perspective. Autonomous Agents and Multi-Agent Systems **23**(2), 158–192 (sep 2011). https://doi.org/10.1007/s10458-010-9140-7, https://doi.org/10.1007/s10458-010-9140-7
20. Vygotsky, L.S., Cole, M.: Mind in society: Development of higher psychological processes. Harvard university press (1978)
21. Weyns, D., Michel, F.: Agent environments for multi-agent systems–a research roadmap. In: Agent Environments for Multi-Agent Systems IV: 4th International Workshop, E4MAS 2014-10 Years Later, Paris, France, May 6, 2014, Revised Selected and Invited Papers. pp. 3–21. Springer (2015)
22. Weyns, D., Omicini, A., Odell, J.: Environment as a first class abstraction in multiagent systems. Autonomous Agents and Multi-Agent Systems **14**(1), 5–30 (feb 2007). https://doi.org/10.1007/s10458-006-0012-0, https://doi.org/10.1007/s10458-006-0012-0
23. Weyns, D., Van Dyke Parunak, H., Michel, F., Holvoet, T., Ferber, J.: Environments for multiagent systems state-of-the-art and research challenges. In: Environments for Multi-Agent Systems: First International Workshop, E4MAS 2004, New York, NY, July 19, 2004, Revised Selected Papers 1. pp. 1–47. Springer (2005)

A The Hierarchy of Computer Systems

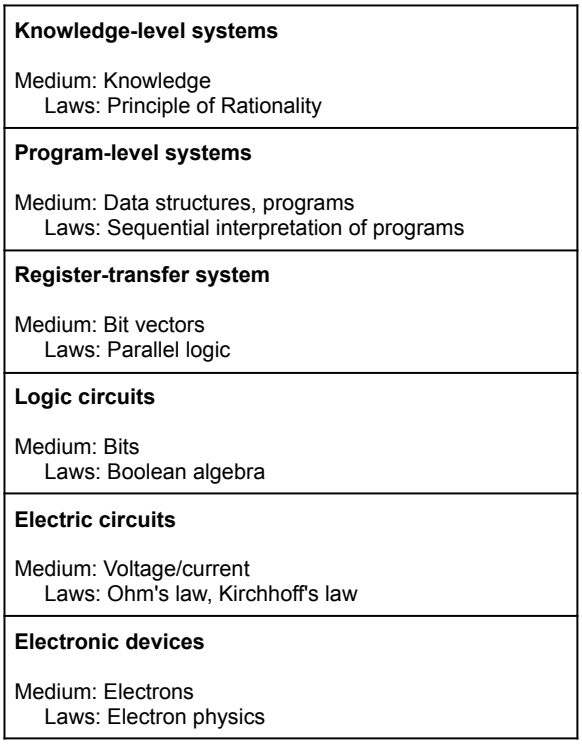

Fig. 2. The hierarchy of computer systems, as reported in [15] (pag. 47)

B Knowledge Level & Social Level

Dimension	**Description**	**Knowledge level**	**Social level**
System	Entity to be described	(asocial) Agent	Agent organisation
Components	The system's primitive elements	Goals, Actions	Agents, Interaction channels, Dependencies, Organisational relationships
Compositional law	How the components are assembled	Various	Roles, Organisation's rules
Behaviour law	How the system's behaviour depends upon its composition and components	Principle of rationality	Principle of organisational rationality
Medium	The elements to be processed to obtain the desired behaviour	Knowledge	Organisation and social obligations, Means of influencing others, Means of changing organisational structures

Table 1. Summary of the knowledge and social levels as reported in [10]