Generating Value Models using Skeletal Design Techniques

Iván S. Razo-Zapata, Ania Chmielowiec, Jaap Gordijn, Maarten van Steen, and Pieter De Leenheer

VU University Amsterdam
De Boelelaan 1081
1081 HV, Amsterdam, The Netherlands
izapata,ania,gordijn,steen,pgm.de.leenheer@few.vu.nl

Abstract. This paper presents a novel approach to automatically generate e^3*value instance models*, based on skeletal design techniques. The approach has three phases. While the first two phases are related to the generation of a *value activity network* based on a given *value skeleton*, the third phase matches the elements of the value network with the capabilities of service providers. The main objective of our approach is to re-use a set of *value skeletons* for covering an industry sector from which more business cases may be generated. Finally, we validated our approach in a realistic case study.

1 Introduction

Enterprises increasingly participate in *networked value constellations* [1]. By doing so, enterprises can jointly satisfy a more complex consumer need than they could ever do on their own. Consider for instance the music industry, our case study domain. If a person listens to music on the radio, apart from the radio station, Intellectual Property Rights (IPR) societies will be needed to collect money for the artists, song writers, producers and other IPR owners. All these actors are part of a value constellation that is needed to listen to a music track on the radio.

In earlier work [2], we proposed the e^3*value* methodology to design value constellations. In brief, e^3*value* is a conceptual modelling approach, which is used to describe a network of enterprises who exchange objects of value with each other.

The contribution of this paper is to lift e^3*value* to the *industry* level, e.g. to describe the music industry, rather than just a single business case, as e^3*value* normally is used for. To do so we present e^3*value skeletons*. One of the most important differences between an e^3*value skeleton* and a normal e^3*value* business case model is that a business case contains named, specific, enterprises who participate in the business case at hand, whereas a skeleton contains only the set of activities to be performed for a business case.

One of the purposes of an e^3*value skeleton* is to easily generate many variations of business cases, which can be presented to stakeholders for selection.

Moreover, since mass configuration of products is playing an important role nowadays [3], our long term ultimate goal is to automatically compose networked value constellations, including the required business processes and IT support in the form of web services. Such IT is then aligned with the business, since both are designed in an integrated way.

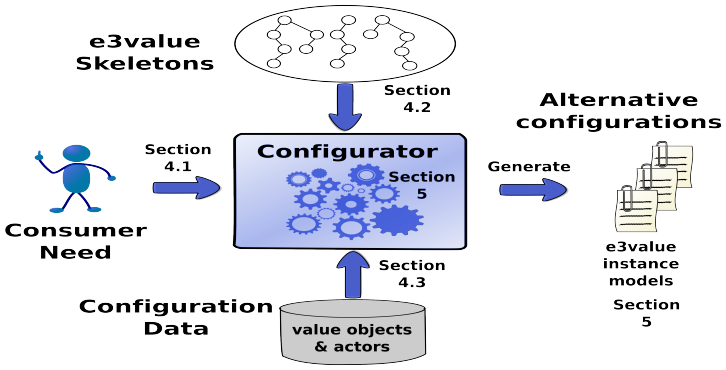


Fig. 1: Generation of value models using skeletal design techniques.

We refer to industry models as *skeletons* and to business case models as *instances*. By using skeletal design techniques (Sect. 2.2), we derive e^3*value instance models* from a *skeleton*. For the music industry (case study description in Sect. 3), we have developed various e^3*value* instance models to study variations of these models. Abstracted versions of these variations are captured in *skeleton* models (Sect. 4.2). By using user input (Sect. 4.1) and configuration data (Sect. 4.3), we generate e^3*value* instance models (Sect. 5). Fig. 1 depicts this general idea. Finally, Sect. 6 presents conclusions and future work.

2 Value models and skeletal design

2.1 Value models

An e^3*value* model depicts a network of enterprises creating, distributing, and consuming objects of economic value [4]. The focus of the model is on **what** kind of objects enterprises must exchange to each other in order to cover consumer needs [1]. Fig. 2 shows (at the bottom) the modelling constructs of e^3*value* and (on top) an educational example.

The most important e^3*value* constructs are as follows: *Actors*, such as a buyer and seller, are economically independent entities (Fig. 2). Actors transfer *value objects* (money, goods) by means of *value transfers*, which in turn connect *value ports*. For value objects, some actor should be willing to pay, which is shown by

[1] and not in HOW, this is the focus of a business process

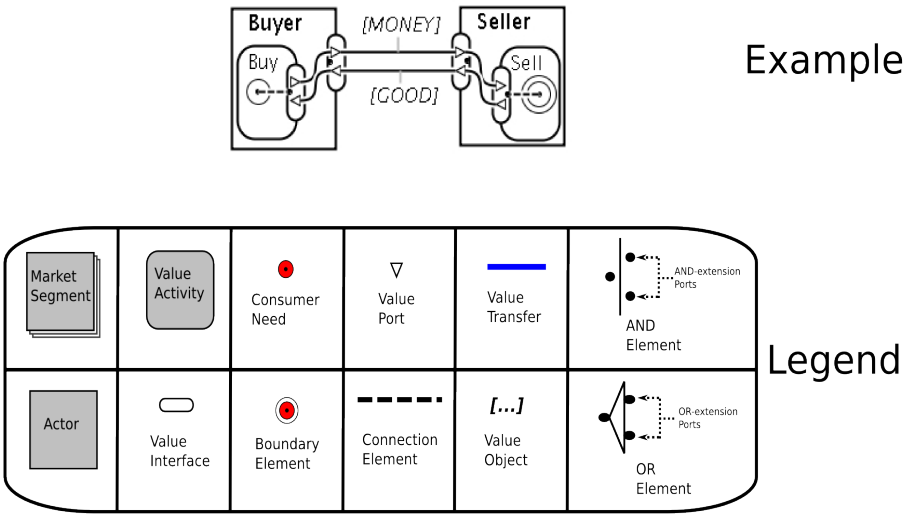


Fig. 2: e^3*value* constructs for value modelling.

a *value interface*. A value interface models the *principle of economic reciprocity*: Only if you pay, you can obtain the goods and vice versa. Besides, actors perform *value activities*, which create something of economic value. Finally, an e^3*value* model contains a dependency path, starting with a consumer need and ending with a boundary element. Along the path are value transfers, value interfaces and connection elements. The dependency path shows how many value transfers are executed as a result of a consumer need. An elaborated formalisation of e^3*value* can be found in [4].

2.2 Skeletal design

Since we want to generate value models by means of value skeletons, it is useful to take into account configuration design theory. According to Motta, a configuration design problem does not exhibit complex spatial requirements and all possible solutions adhere to *a common solution template* [5]. For this reason we think the configuration of services, by using value models, adheres to a generic template. Wielinga and Schreiber describe four main categories of configuration design: *assignment*, *scheduling*, *skeletal design* and *parametric design* [6]. The last two categories are related to our research.

Parametric design is described as the process of assigning values to *design parameters* not only satisfying needs and constraints but also following an opti-

mization criterion [5] [2]. Skeletal design refers to a model-based mapping between components and the assembly [6]. Because our solution is not considering assigning values to a fixed template but a model-based generation of new components, our approach is more related to skeletal design.

To sum up, our approach considers that the target artifact is modeled as a set of *elements* and the solution implies *generating elements* based on *a common solution template* matching the given design requirements and constraints (Sect. 5).

3 Case study: Clearing Intellectual Property Rights

In the music industry, Intellectual Property Rights (IPR) are an important concept. Neighboring rights are a kind of IPR, which have to be paid by users if they earn money by playing music, or in other words, if they make music public. Clearing such neighboring rights involves two steps: collecting fees from IPR users, *i.e.* radio stations, bars, discotheques and others, and distributing these fees to Right Owners, *i.e.* artists, song writers, producers. This process is usually performed by IPR societies and is called *clearing tracks*. One of the IPR societies designed to perform this activity in The Netherlands is SENA [3], our case study partner which is also one of our stakeholders.

Some results for modeling this case study have been already provided [2], however these results only address one of the multiple scenarios that can emerge in the music industry, such as new actors performing less or more activities because of *market liberalization* and new trends in the way of *delivering music* [7, 8]. One such a trend is that SENA, instead of clearing tracks based on play lists of the top 30 radio statios (estimation), clears tracks based on the actual usage (*pay-per-play*).

The e^3*value* model depicted in Fig. 3 presents a pay-per-play solution for background music. In this e^3*value* model, IPR users are the starting point, as they require to broadcast background music which is provided by background music providers (BMP). A BMP can provide background music in different ways, one of those is streaming (Value Transfer A). When providing streams, the BMP must pay to IPR Societies which collect fees related with making a stream available to the public (Value Transfer D-E). Here, the public refers to IPR users.

IPR users have to also pay IPR Societies, as they make public the music also to their customers. Although the BMP and IPR users pay to two IPR Societies, it does not mean they pay twice for the same thing. Paying BUMA/Stemra [4] is because of the right that the composers, publishers and lyricists hold (Value Transfer C), whereas paying SENA is related to the rights of the performing

[2] Motta also mentions preferences, however at this stage they are not essential to find an optimal solution.

[3] (Dutch: Stichting ter Exploitatie van Naburige Rechten, English: Foundation for Exploitation of Neighboring Rights)

[4] BUMA is other IPR society, also working in The Netherlands

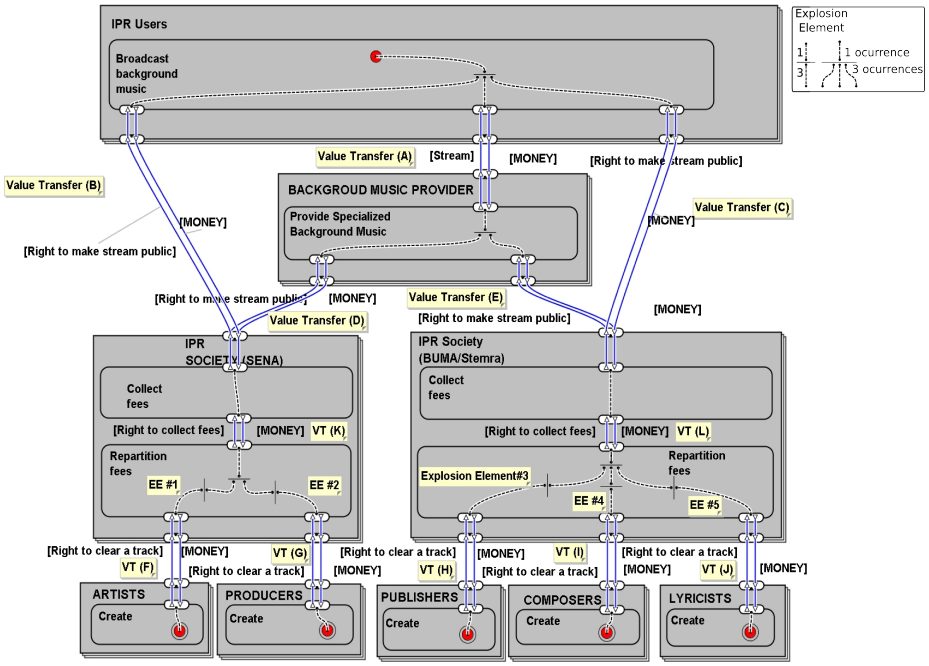


Fig. 3: e^3value model for the stream-based solution.

artists and producers (Value Transfer B). Indeed, Value Transfers A-E are constraints imposed by law, since IPR societies are the responsable entitites for collecting fees on behalf of IPR owners.

Once IPR societies got IPR-user fees, they retain a small percentage of these fees and the next step is to repartition the rest of the fees among Right Owners (Value Transfer K-L). This set of Right Owners is mainly composed by Artists, Producers, Music Publishers, Composers and Lyricists. Whereas SENA only repartitions fees to Artists and Producers (Value Transfer F-G), BUMA/Stemra does the same for Publishers, Composers and Lyricists (Value Transfer H-J).

Explosion Elements Due to the fact that tracks are usually performed by more than one artist, SENA must transfer the collected fees to each artist involved, consequently the value transfer F will occur more than one time per consumer need. In order to allow that SENA contacts more artists, an *explosion element* (EE #1) is added. Since the same holds for all the right owners, there are more explosion elements for each of them (EE #2 - EE #5). An explosion element models that the connected value transfers happen multiple times, rather than just once per dependency path execution.

In this scenario there are only two enterprises clearing tracks (SENA and BUMA). As we described before, market liberalization will require a more flexible scheme to assign value activities to different enterprises, *e.g.* the activi-

ties "Collect Fees" and "Repartition Fees" can be performed by different actors rather than one actor. In order to address this problem we have generated and analyzed variations in which several enterprises can cover different value activities. We have used these variations to build a few *e³value skeletons*, however because of space restrictions just one of these *skeletons* is described in the next section.

4 Input elements for the configuration process

4.1 Consumer needs

For the configuration process, we assume that our consumer need is *playing background music* and that we can cover such a need by *providing a stream*. In addition, since the pay-per-play scenario must be analyzed, we also assume that our stream contains only one music track. So, the need boils down to a streamed track.

4.2 Value skeletons

The purpose of a skeleton is to abstract the set of relationships that are involved in an *e³value* model. Later on, the instances of a skeleton will reflect a specific business case. In our approach, an *e³value skeleton* implies three important features, which are described and explained by presenting our *e³value skeleton*(Fig. 5).

No actors, No market segments *e³value skeletons* focus on the definition of value activities and not on actors or market segments. In fact, assigning performing actors to value activities is an important part of configuring an *e³value* instance model (see Sect. 5.4).

No Selection The main idea behind skeletons is that there should be an automatic process to instantiate specific business models by filling the elements in such skeleton. Therefore, this process must skip selection procedures like choosing among different dependency paths or different actors, as is normally the case in *e³value* models. In *e³value* words it means that we do not use OR-forks in dependency paths nor market segments.

On the use of Explosion Elements Consider the value transfers F-J in Fig. 3, these value transfers involve the repartition of fees among right owners and look very similar. By using explosion elements, we can abstract these transfers as shown in Fig. 4.a. While generating an instance model, this abstraction can be expanded to cover as many value activities as needed. In this example, we assume that fees must be repartitioned among three right owners, hence three

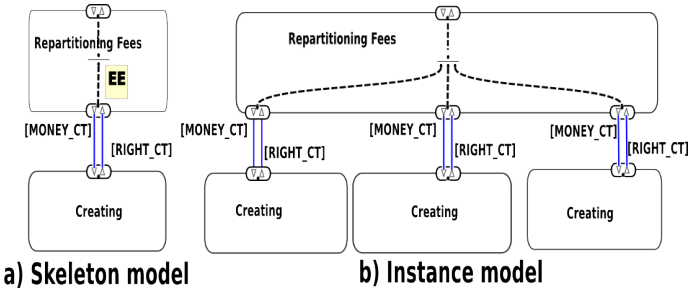


Fig. 4: Using an *explosion element*.

AND-extension ports are needed, which later allow connection with three other value activities (Fig. 4.b).

In the same way, our e^3*value skeleton* (Fig. 5) depicts abstracted versions of the value transfers B-E, K and L (Fig. 3), which are later instantiated into new transfers. How the instantiation process of the skeleton works is explained in Sect. 5.2. Finally, to facilitate reading, our skeleton also depicts short names for some value objects. In this way, RIGHT_MP refer to the Right for **M**aking **P**ublic a track, RIGHT_CL is the Right to **C**o**L**lect fees, and RIGHT_CT is the Right to **C**lear a **T**rack. The same holds for MONEY_MP, MONEY_CL and MONEY_CL.

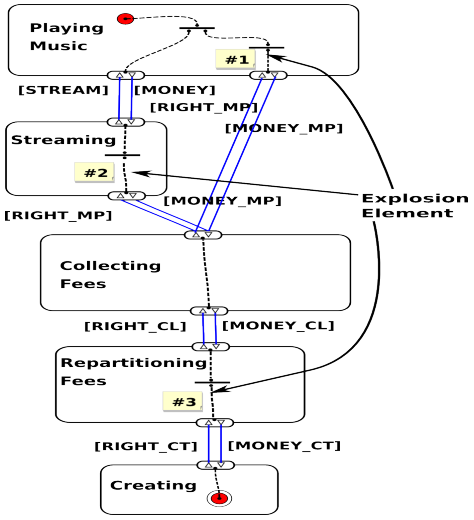


Fig. 5: e^3*value skeleton.*

4.3 Configuration data

The information about the environment in which the configuration process takes place is the last element to be specified before starting the configuration process. This environment, called *configuration data*, is composed of eight elements. The main relationships among these elements are depicted in Fig. 6.

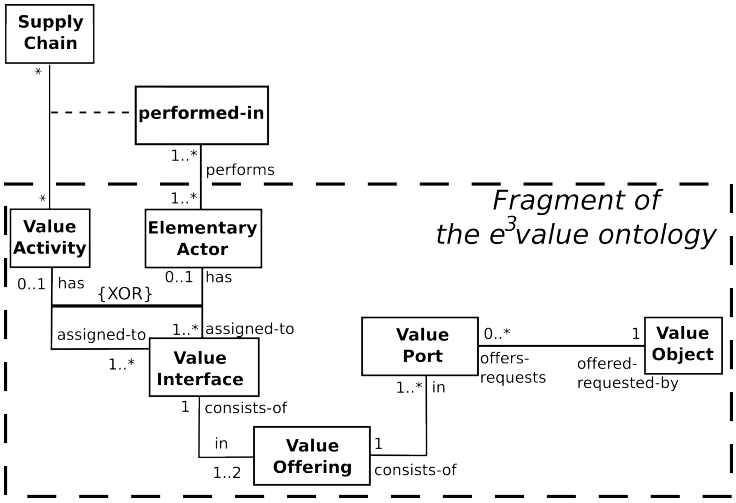


Fig. 6: Configuration data - class diagram.

Supply Chains The clearing process deals with five types of suppliers: *artists*, *producers*, *composers*, *lyricists* and *publishers*. Most of the time suppliers bring about different dependency paths in which different value activities and value objects are involved. We refer to these paths as *supply chains*. As can be observed in Fig. 6, a supply chain **consists of** several activities, *e.g.* in our case study, activities like Streaming, Collecting Fees, Repartitioning Fees and Creating are needed to provide a track, *i.e.* supply a service. Actually, these are the same activities that appear in the skeleton.

Elementary Actors The elementary actors represent the set of service providers willing to participate in a business case. In this sense, an elementary actor **has** value interfaces to interact with other elementary actors.

Value Activities In the same way, value activities represent the set of activities that must be performed to cover a consumer need. Therefore value activities **has** value interfaces for transfering value objects. As defined in Fig. 6 a value activity has a special relationship in which supply chains and elementary actors are involved. In fact, this relationship yields an aggregation class, the same is explained as follows.

Performed-in By using this class we want to express that elementary actors can only **perform** value activities related to specific supply chains. As an

example, SENA only take care of activities like Collecting and Repartitioning when they deal with artists and producers. BUMA does the same for the rest or right owners.

Value Interfaces A value interface is **assigned-to** either an elementary actor or a value activity but not both at the same time, which is depicted by using the XOR constraint. A value interface **consists-of** one or two value offerings, what the actor wants and expects.

Value Offerings A value offering represents what an actor offers through value interfaces. Therefore a value offering is **in** a value interface and **consists-of** values ports.

Value Ports By using value ports an actor either **offers** or **requests** value objects. In addition a value port is only **in** one value offering.

Value Objects Value objects represent the services or products that actors exchange to each other. For instance, according to our case study (Fig. 3), actors exchange value objects like RIGHT_MP, RIGHT_CL and RIGHT_CT.

5 Configuration process

This process involves three phases. While the first two phases are related to the generation of a value activity network based on a given value skeleton, the third phase matches the elements of the value network with a set of service providers.

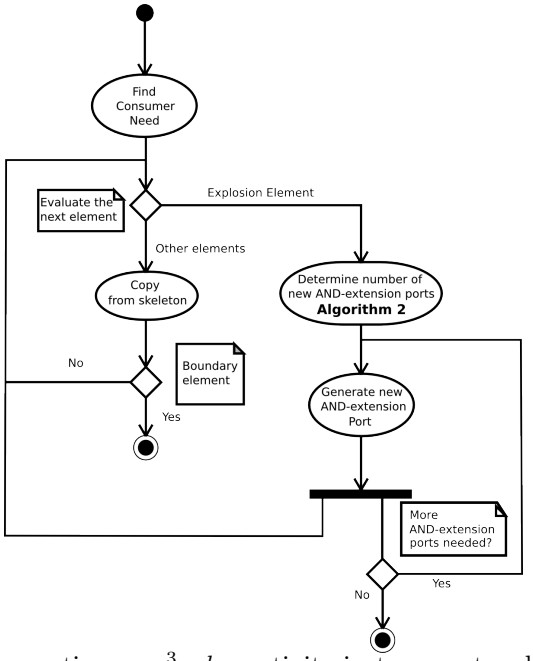
5.1 Selecting an *e³value skeleton*

The first step to build an *e³value* instance model is related to selecting an *e³value skeleton* from which the generation process can start. In this way, the *e³value skeletons* must be chosen according to a consumer need, *playing background music* (see Sect. 4.1). At this point we manually select an *e³value skeleton*. For a guideline how to perform a semi-automatic process see [9].

5.2 Generating an *e³value* activity instance network

Once a consumer need is defined and an *e³value skeleton* is chosen (in this case Fig. 5) , the generation process traverses this skeleton and produces an *e³value* activity instance network. Algorithm 1 performs this process generating this *e³value* activity instance network while traversing the skeleton based on a depth-first traversal method, where the consumer need and the boundary element are respectively the root and the leaf of the tree.

Algorithm 1 considers each element in the skeleton. If the considered element is not an explosion element, the element is simply copied into the value activity instance network. If however the element is an explosion element, Algorithm 2 is executed, which evaluates the explosion elements and determines the number of new AND-extension ports to be generated.



Algorithm 1 Generating an e^3value activity instance network.

Explosion Parameters Each explosion element has a set of parameters. These parameters are *e3_explosion*, *ce_related_actor* and *ce_refers_to_vo*. How these parameters work is explained in the Algorithm 2, and their effect is illustrated in Fig. 7.

The parameter called *e3_explosion* controls whether the explosion element will be evaluated. As can be observed in Fig. 7, there exist two ways to evaluate the explosion element:

1. the explosion element will result in a number of parallel supply chains (See Fig. 7a/b, *i.e.* $ce\_related\_actor == FALSE$),
2. the explosion element will result in a number of actors (Fig. 7c/d, *i.e.* $ce\_related\_actor == TRUE$).

5.3 Running example

We illustrate the case when a track, DE_WAARHEID, is cleared on behalf of three artists and two producers, i.e. two supply chains: one for artists and the other one for producers. For readability we omit the composers, lyricists and publishers. Therefore, Algorithm 1 travers the skeleton (Fig. 5). Once Algorithm 1 finds an explosion element, Algorithm 2 is applied to determine the number of new AND-extension ports to be generated.

For instance, assume that our algorithm walks down following the value transfer related to STREAM/MONEY and finds the explosion element #2 (See

Algorithm 2 Determine the number of new AND-extension ports

if $e3\_explosion == TRUE$ **then**
  if $ce\_related\_actor == FALSE$ **then**
    $value\_object = ce\_refers\_to\_vo$
    $New\_AND\_E\_ports$ = Number of supply chains in which $value\_object$ is involved
  else
    $value\_object = current\_track$
    GET current $supply\_chain$
    GET $next\_value\_activity$ according to the skeleton
    EA = set of actors performing $next\_value\_activity$ in $supply\_chain$ and offering $value\_object$
    $New\_AND\_E\_ports$ = Number of actors in EA.
  end if
else
  $New\_AND\_E\_ports = 1$
end if

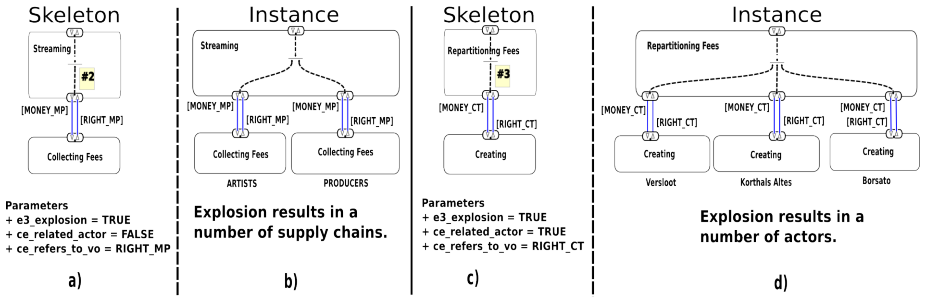


Fig. 7: Explosion parameters. a) Two activities and one explosion element. b) Instantiation of new elements based on *supply chains*. (Generation of supply chains) c) Two activities and one explosion element. d) Instantiation of new elements based on the number of elementary *actors offering a specific value object* (Splitting a supply chain).

Fig. 5). The evaluation of this explosion element takes place *i.e.* Fig. 7 .a and 7.b. This means that we have to count the number of supply chains, and we have to generate the appropriate number of AND-extension ports for that. Since our supply chains are artists and producers, and the RIGHT_MP is involved in the artists and producers supply chains, two new AND-extension ports must be generated, as can be seen in Fig. 8, element #2.

We continue to traverse the skeleton (for the artists supply chain) and find explosion element #3. This time, the evaluation is performed as in Fig. 7.c and 7.d. The number of new AND-extension ports depends on the number of elementary actors performing the Creating value activity in the artists supply chain and for the specific track, i.e. three artists. The process of copying elements from the skeleton continues till reaching a boundary element. Fig. 8 depicts

the final e^3*value* activity network, the numbering indicates the order in which element are instantiated.

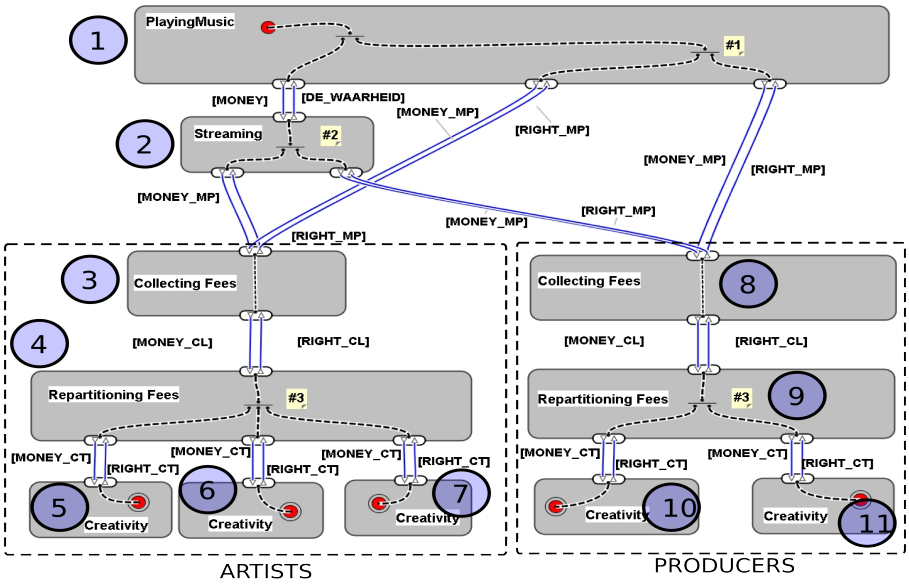


Fig. 8: e^3*value* activity network.

5.4 Assigning value activities to actors

After generating the e^3*value* activity network the next phase assigns activities to service providers. In a naive-sequential way, Algorithm 3 describes this process.

Algorithm 3 Assign activities to actors

```
for each value activity to be assigned do
  assigned = FALSE /* Boolean Flag */
  for each elementary actor && assigned == FALSE do
    if the elementary actor can perform this value activity in the specific supply
    chain and provide the needed value object then
      assign activity to elementary actor
      assigned = TRUE
    end if
  end for
end for
```

As can be observed in Algorithm 3, there is a search to determine whether an elementary actor can perform a value activity in a specific supply chain (by evaluating the performed-in association in Fig. 6). For instance, when assigning the value activity Collecting Fees (Fig. 8, number 8), the algorithm will search for an elementary actor that can perform this activity in the supply chain related to producers. As we already know, SENA is able to perform such as activity in that specific supply chain, therefore the algorithm assigns this activity to SENA, as shown in Fig. 9.

5.5 Running example

The model in Fig. 9 represents the way in which a set of enterprises work together to cover a consumer need which is composed of two things: DE WAARHEID and the needed rights to make this track public.

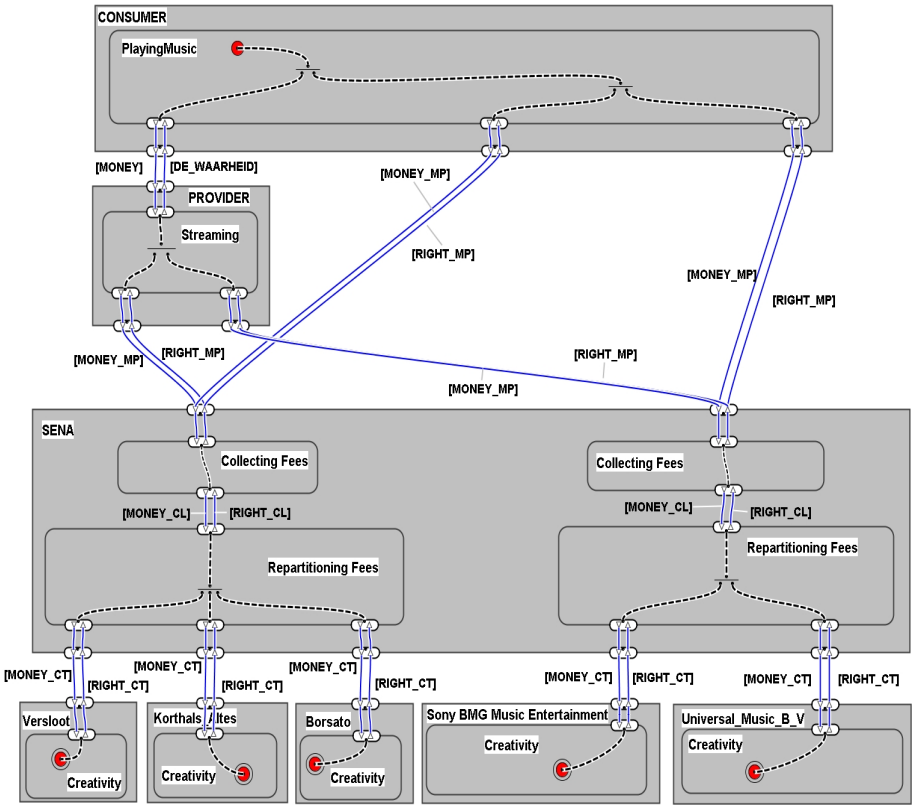


Fig. 9: Instance value model based on skeleton in Fig. 5.

As can be observed, the set of needed rights contains the rights from artists and producers. Consequently, the consumer must contact to enterprise(s) taking care of those rights, *i.e.* SENA. In the same way, since the stream provider is making public DE WAARHEID, it also needs to get the same rights. In order to get the rights from the set of right owners, SENA should also perform an activity related to repartitioning fees.

Furthermore, the activities related to repartitioning the fees contact the right owners associated to DE WAARHEID. The supply chain related to artists ends with three artists, whereas the supply chain of producers involves two right owners.

Finally, by applying the previous configuration approach, Sections 4 and 5, we have generated different e^3value instance models for other tracks. In this way, we test that by just changing our configuration data *i.e.* actors, their value objects and the activities they perform, we can generate instance models re-using the same skeleton and applying our configuration process. Nevertheless, due to lack of space these instance models are not presented [5].

6 Conclusions and future work

This paper has presented an approach to automatically generate e^3value models, based on skeletal design techniques. One of the objectives of our approach is to re-use a set of value skeletons for covering an industry sector from which more business cases may be generated. Consequently, in order to explain the intended use of this approach we have also presented a case study.

By following the idea of skeletal design we have exemplified how re-use of knowledge can help to solve the problem of massive service configuration. Our e^3value *skeletons* are useful to span (part of) an industry sector, helping to either automate current tasks or forecast/explore new scenarios.

Future work

Our next steps will address the problems related to elicitation of consumer needs, selection of skeletons and assignment of value activities to service providers, i.e. to add more flexibility to Algorithm 3. Consequently, while the work of Sybren de Kinderen [9] represents a good base-line to deal with the first issue, more research must be done to evaluate whether hierarchical configuration or skeletal planning could be implemented to cover our second issue.

Finally, even though our approach solve our configuration problem, we are aware of the limitations of this tool, mainly because some reasoning steps seems to be case specific. Therefore, more validation must be done and another case study will be explored, related to the energy industry where more dynamic behavior can be found.

Acknowledgements. This paper has been partially funded by the NWO/-Jacquard project VALUE-IT no 630.001.205

[5] The reader can consult some instances at: http://www.few.vu.nl/∼izapata/Generating-VM-SDT.php

References

1. J. Gordijn, S. de Kinderen, V. Pijpers, and H. Akkermans, "e-services in a networked world: From semantics to pragmatics," *Lecture Notes in Computer Science*, vol. 5468/2009, pp. 44–57, 2009.
2. J. Gordijn, E. Yu, and B. var der Raadt, "e-service design using i * and e3value modeling," *IEEE Software*, vol. 0740-7459/06, pp. 26–33, 2006.
3. D. Yang, M. Dong, and R. Miao, "Development of a product configuration system with an ontology-based approach," *Computer-Aided Design*, vol. 40, pp. 863–878, 2008.
4. J. Gordijn and J. Akkermans, "e3-value: Design and evaluation of e-business models," *IEEE Intelligent Systems*, p. 1117, 2001.
5. E. Motta, *Reusable Components for Knowledge Modelling: Case Studies in Parametric Design Problem Solving*. IOS Press, 1999.
6. B. Wielinga and G. Schreiber, "Configuration design problem solving," *IEEE Expert. Special issue on AI in Design.*, vol. 12, pp. 49–56, 1997.
7. G. Premkumar, "Alternate distribution strategies for digital music," *Communications of the ACM*, vol. 46, pp. 89–95, 2003.
8. P. Swatman, C. Krueger, and K. van der Beek, "The changing digital content landscape: An evaluation of e-business model development in european online news and music," *Internet Research*, vol. 16, pp. 53–80, 2006.
9. S. de Kinderen, J. Gordijn, and H. Akkermans, "Eliciting multi-supplier customer needs-driven it-services bundles using compensatory and non-compensatory decision models," *1th International Conference on Enterprise Information Systems*, pp. 131–136, 2009.