An Automatic Layout Approach for iStar Models

Xiaolin Du, Tong Li, and Dan Wang

Beijing University of Technology, Beijing, China
{du_xiaolin,litong,wangdan}@bjut.edu.cn

Abstract. The comprehensive syntax of iStar modeling language allows requirements analysts to clearly capture stakeholder's needs, as well as dependencies among stakeholders. However, such iStar models cannot be automatically laid out using typical layout algorithms, such as hierarchical layout and circular layout. Thus, constructing and adjusting iStar models are laborious tasks, especially when dealing with large-scale models. In this paper, we propose a tentative approach to automatically lay out iStar models using a force-based layout algorithm. In particular, our approach has been designed by taking into account the syntax of iStar models in order to ensure both neatness and understandability of resulting models.

1 Introduction

iStar modeling framework has been used as an efficient approach to capture and analyze stakeholder's needs in the early phase of requirements engineering. In particular, the comprehensive syntax of iStar modeling language allows requirements analysts to clearly capture stakeholders' requirements, their rationales, as well as dependencies among stakeholders.

However, to the best of our knowledge, there is no algorithm to automatically lay out iStar models. Typical layout algorithms, e.g., hierarchical and circular layout algorithms, cannot serve this purpose, because they do not fit iStar syntax. As a result, analysts have to spend a significant amount of time dealing with layout issues when creating or modifying iStar models. Such a problem is exacerbated by the growth of model scale, resulting in a scalability problem [6].

Challenges of automatic layout of iStar models inherit from their comprehensive syntax. In particular, there are two views of presenting iStar models, each of which has its particular requirements for layout. Firstly, the SD (Strategic Dependency) view exclusively presents actors and their dependency relations, the layout of which should avoid crossed links while maximizing model readability. Secondly, the SR (Strategic Rationale) view includes internal details of actors, e.g., goals and tasks, which are supposed to be presented as a tree-like structure.

Using force-based layout algorithms is a classic graph drawing method, which simulates a physical system with edges acting as springs and nodes as repelling objects [12]. As is shown in Fig. 1, when running such an algorithm, nodes

are pushed away from each other due to their repelling forces while edges hold the nodes together. By iteratively running the algorithm, a graph will eventually come to an equilibrium in which nodes do not change positions from one iteration to the next.

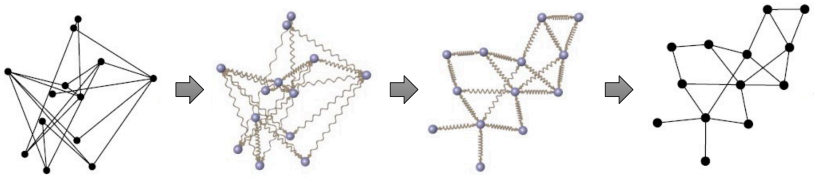

Fig. 1: The spring analogy [12]

In this paper, we present our ongoing research of automatic layout of iStar models. In particular, we propose a forced-based algorithm to automatically lay out elements in the SD view, which has been customized based on the syntax of iStar models in order to produce meaningful layouts. The automatic layout of SR view is not covered in this paper. We will discuss our initial ideas towards this issue and left its in-depth investigation for future research. It is worth noting that terminology used in this paper is aligned with iStar 2.0 modeling language.

2 A Forced-Based Layout Algorithm for iStar Models

2.1 Layout of SD View

When it comes to layout of the SD view, a primary goal is to calculate a reasonable position for each actor. To this end, we need to first transform an iStar SD model into an undirected relational graph, which is then used as input of our force-based layout algorithm. Specifically, each actor in the SD model is turned into a node, and relationships among actors (e.g., *dependency* and *part-of*) are abstracted as edges in the relational graph. It is worth noting that the direction of an edge does not affect the outcome of our layout algorithm, and thus is not considered in this paper.

Dependency relationships among actors are essential in the SD view, reflecting to which extent two actors interact with each other. In order to produce meaningful layout, our approach takes into account such dependency relationships. Specifically, the more dependency relationships exist between two actors, the closer they are. As a result, when abstracting edges among actors, we also calculate their corresponding weights and store them in an edge weight matrix EW, The element in the matrix EW_{ij} is the number of dependency relationships between actors. For example, $EW_{ij} = 3$ means that there are three dependency relationships between actor i and actor j. Fig. 2 shows an example of the transformation from an iStar SD model to a relational graph.

Let $G(V_G, E_G)$ denotes a relational graph, V_G represents the node set of G, and E_G represents the edge set of G. We use FR (Fruchterman and Reingold)

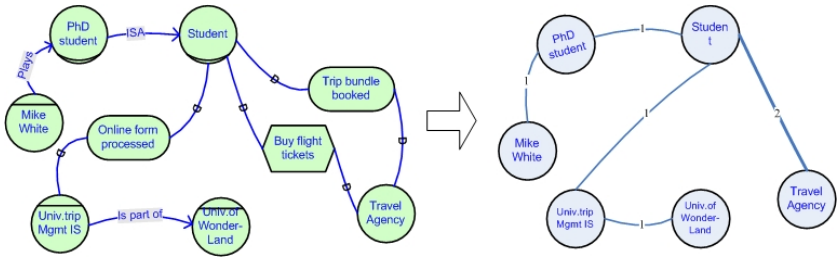

Fig. 2: An example of graph transformation

layout algorithm [7] to calculate the positions. Each iteration of the FR layout algorithm consists of three steps: 1) calculate the effect of attractive forces on each vertex; 2) calculate the effect of repulsive forces; 3) limit the total displacement. Note that forces are modeled based on the optimal distance between vertices, which is calculated as below:

$$k = C * \sqrt{\frac{area}{number\ of\ nodes}} \tag{1}$$

As shown in Eq. 1, k is calculated as the radius of the empty area around a node, where the constant C is found experimentally. In such a way, all nodes are able to be uniformly distributed in a canvas. Subsequently, given d the distance between the two nodes, attractive forces f_a and repulsive forces f_r are defined as below:

$$f_a(d) = \frac{d^2}{k} \tag{2}$$

$$f_r(d) = -\frac{k^2}{d} \tag{3}$$

Our layout approach takes into account the weight of each edge. Specifically, an edge with bigger weight indicates that its endpoints will be arranged closer. Thus, we improve the attractive force formula as

$$f_a(d) = EW * \frac{d^2}{k} \tag{4}$$

For edges that are derived from dependency relations, the corresponding dependum will be modeled in the middle of edges. As such, two nodes should not be placed too close to each other. To this end, a threshold τ is defined to control the minimal distance between connected nodes.

As summarized in Algorithm 1, our approach first transforms a textually represented iStar model into a relational graph, while forming an edge weight matrix. After that an improved FR algorithm is iteratively applied to calculate the position of each node, based on which a graphical iStar model can be constructed.

2.2 Layout of SR View

As for the layout of SR view, which is left for future work, we discuss here our initial thoughts towards this issue. In general, this layout task can be divided

Algorithm 1 A customized force-based layout algorithm

Input: a textual specification of a iStar model
Output: a graphical representation of the iStar model
1: Transform the iStar model to a relational graph G;
2: Calculate an edge weight matrix EW;
3: Apply the improved FR algorithm;
4: { Calculate k
5: **For** $i = 1$ to iterations **Do**
6: Calculate repulsive forces;
7: Calculate attractive forces;
8: Adjust the total displacement;
9: if the distance between two nodes is less than τ
10: adjust moving direction;
11: }
12: Construct the graphical iStar model

into two sub-tasks: laying out all the intentional elements inside actor boundaries and positioning each actor boundary as a whole in the entire picture. Since most intentional elements are supposed to be organized in a tree-like structure, an intuitive solution is using a hierarchical layout algorithm. However, for quality requirements and corresponding contribution links, which are laid out in a different manner, we need to design particular algorithms to deal with their layout. Once elements inside actor boundaries can be well arranged, we can then treat each of them as a single element and layout them together with actor nodes. To accommodate this, we will need to further adjust the force-based algorithm proposed in this paper.

2.3 Evaluation Plans

Once we extend our proposal to cover the layout issue of SR view, we plan to implement the entire layout approach as an independent library, which is able to be integrated into various iStar modeling tools [1]. As a starting point, we plan to first implement the layout approach on top of our previous goal modeling tool MUSER [2], serving as a prototype tool.

With the help of the prototype tool, we plan to evaluate the effectiveness of our layout approach through a series of studies. Specifically, we are interested in investigating the following research questions:

- **Can our approach create understandable layout of i models?** We plan to carry out several user studies, in which we will ask iStar experts to assess to which extent the resulting layout of our proposal can be understood.
- **Is our approach scalable to large-scale models?** We will evaluate the scalability issue of our approach by applying it to a set of large-scale models.

[1] `http://istar.rwth-aachen.de/tiki-index.php?page=i%2A+Tools`
[2] `http://disi.unitn.it/~li/MUSER/Intro.html`

– **Can our approach lead to faster modeling practices?** We plan to design a controlled experiment to compare the performance of two groups, one of which is assisted by our approach. In particular, we aim to collect feedback from participants via focus group interview, based on which we can improve our approach to better support iStar modeling.

3 Related Work

We have been aware of a number of proposals that are relevant to ours. OpenOME and jUCMNav provide some basic graph layout functions to automatically arrange the goal elements, which mainly focus on laying out intentional elements inside actors [2,1]. These approaches can well complement our current proposal, rendering a comprehensive layout approach. Moreover, Gregorics et al. proposed a textual layout description language, which allows users to define the arrangement of important diagram elements [10]. Ghazi et al. propose FlexiView to improve the visual representation of requirements artifacts, which also leverages magnetic-spring algorithms but in a different manner [8]. Specifically, their approach partitions requirements engineers' workspace based on the magnetic-spring model in order to present different types of requirements artifacts at the same time, minimizing scrolling distance.

Different from most iStar modeling tools, Grau et al. propose J-PRiM which does not show the i* elements in a graphical way but in a tree-form hierarchy [9]. In such a way, as models grow, the scalability issue can be relieved, and modelers are able to find particular elements in a faster way. Note that such a textual modeling approach and our proposal can complement each other. Specifically, management of iStar models can be done in the textual view, while our approach is responsible for visualizing the textual models when necessary.

Graph visualization helps users to gain insight into data by turning the data elements and their internal relationships into graphs [4]. Graph layout problems are a particular class of combinatorial optimization problems whose goal is to find a linear layout of an input graph in such a way that a certain objective function is optimized [5]. Force-based layout is a very popular graph visualization method. It is first proposed by Eades in 1984, which has then been revisited and improved by many researchers in different ways (e.g., [11]). Force directed model can be expressed either in terms of forces acting on the physical objects, or in terms of a potential energy reflecting the internal stress of the system. Algorithms to simulate a system's relaxation typically try to move the objects iteratively into stable states in which all forces cancel each other, or to minimize the energy directly [3].

4 Conclusions and Future Work

In this paper, we present a tentative approach to automatically lay out iStar models using a customized force-based layout algorithm. As ongoing research, our proposal mainly focuses on the layout of iStar SD view thus far. In addition,

we have discussed possible ways of extending our approach in order to lay out elements in the SR view.

Our approach can be applied to textual representations of iStar models, the benefits of which are twofold. Firstly, modelers are able to exclusively focus on the content of models to be built without being bothered by layout issues. As such, our approach can save much effort on the creation of iStar models, especially when modeling large-scale scenarios. Secondly, there are increasing demands of improving usability of iStar modeling tools in order to better deal with scalability issues [13]. With the help of our approach, tool developers can also be released from never ending improvement of graphical interfaces. Moreover, by enhancing iStar tools with the automatic layout feature, they are able to better support goal model reasoning tasks that incrementally add new elements, e.g., goal refinements.

As for future work, we plan to first extend our proposal is to deal with automatic layout of the SR view. After that we will implement our layout approach in a particular iStar modeling tool, based on which we will evaluate the usefulness of our approach through a series of empirical studies.

Acknowledgments. Tong Li acknowledges the support of BJUT Startup Funding No.007000514116022. Xiaolin Du acknowledges the support of Beijing Postdoctoral Research Foundation No.Q6007012201601.

References

1. jUCMNav. `http://jucmnav.softwareengineering.ca/jucmnav`.
2. OpenOME. `http://istar.rwth-aachen.de/tiki-index.php?page=OpenOME`.
3. U. Brandes. Drawing on physical analogies. *Drawing graphs*.
4. W. Cui and H. Qu. *A survey on graph visualization*. PhD thesis, Hong Kong University of Science and Technology, Hong Kong, 2007.
5. J. Diaz, J. Petit, and M. Serna. A survey of graph layout problems. *ACM Computing Surveys*, 34(3):313–356, 2002.
6. H. Estrada, A. M. Rebollar, O. Pastor, and J. Mylopoulos. An empirical evaluation of the i* framework in a model-based software generation environment. In *Advanced Information Systems Engineering*, pages 513–527. Springer, 2006.
7. T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software - Practice and Experience*, 21(11):1129–1164, 1991.
8. P. Ghazi, N. Seyff, and M. Glinz. Flexiview: a magnet-based approach for visualizing requirements artifacts. In *REFSQ2015*, pages 262–269. Springer, 2015.
9. G. Grau, X. Franch, and S. Avila. J-prim: A java tool for a process reengineering i* methodology. In *Requirements Engineering, 14th IEEE International Conference*, pages 359–360. IEEE, 2006.
10. B. Gregorics, T. Gregorics, G. F. Kovacs, A. Dobreff, and G. Devai. Textual diagram layout language and visualization algorithm. pages 196–205, 2015.
11. T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989.
12. M. S. ld. Social network visualization. Master's thesis, School of Computer Science and Engineering, Royal Institute of Technology, Sweden, 2008.
13. T. Li, A. M. Grubb, and J. Horkoff. Understanding challenges and tradeoffs in istar tool development. In *9th iStar Workshop*, pages 49–54, 2016.