Combining Two Modelling Approaches: GQM and KAOS in an Open Source Project

Nursultan Askarbekuly(✉), Andrey Sadovykh, and Manuel Mazzara

Software Engineering Laboratory, Innopolis University, Innopolis, Tatarstan, Russia
n.askarbekuly@innopolis.university

Abstract. GQM is a software metrics method that produces meaningful and appropriate measurement metrics based on the specific goals of an organisation. KAOS goal modelling is a software engineering approach that allows to identify high level goals of an organisation, refine and expand them into lower-level more concrete goals and assign them to specific system agents. The two approaches can be combined, such that the goal modelling is used to derive specific goals rooted in the organisational context, policies and strategies of an organisation. The GQM, in turn, utilises the derived specific goals to produce appropriate and meaningful metrics that can be used to obtain valuable insights and track the state and progress of the formulated goals. Moreover, both models can be presented within a single diagram, thus providing an expressive and concise overview of the systems goals, actors and measurements. The suggested combination is then applied in the context of an open source project, followed by the discussion on potential benefits and drawbacks of using the two approaches.

Keywords: Goal Question Metric · GQM · KAOS · Goal model · Software metrics · Requirements engineering · Open source software

1 Introduction

Goal, Question, Metric (GQM) is a measurement modelling approach applicable in a wide variety of contexts ranging from assessing software reliability and validation of engineering techniques to evaluating the impact of business strategies [5–7].

It allows engineers to choose appropriate software measurement metrics. The GQM method takes a set of specific goals of an organisation as a basis, and connects them with metrics in order to measure their achievement and progress. Thus having a set of specific goals of an organisation is a prerequisite to applying the GQM model and extracting the metrics.

To produce the specific goals, understanding the higher-level strategic goals and the organisation environment is necessary and requires a substantial amount of effort, such as interviewing the stakeholders and working in close collaboration

Published by Springer Nature Switzerland AG 2020
V. Ivanov et al. (Eds.): OSS 2020, IFIP AICT 582, pp. 106–119, 2020.
https://doi.org/10.1007/978-3-030-47240-5_11

with the organisation's team. However, the GQM in its essence does not reflect the context and high-level goals from which it was derived, and therefore requires other more general modelling approaches to reflect the bigger picture.

KAOS is a goal modelling approach used in requirements engineering [1,2]. It identifies high level system goals and proceeds top down to refine them, thus producing more concrete lower-level goals. The lower-level goals are then assigned to agents, such as software modules, external systems or individuals, responsible for the satisfaction of these goals. The diagrammatic notation used in the approach allows one to express and analyse various complex relations between the system goals and entities, and allows one to capture a system's functional and nonfunctional aspects and constraints.

The correspondence and complementary nature of the two approaches makes combination and interchange of their techniques and tools possible. In particular, the KAOS goal model's comprehensive diagrammatic notation can be used to reflect and facilitate the derivation process of the specific set of goals from the organisational context and higher level strategic goals. The GQM, in turn, can use the derived set of the specific goals to produce the set of appropriate metrics to measure and analyse the state and achievement of those goals. As a result, the two approaches produce a mapping of high-level strategic goals to more concrete lower-level goals, which are then assigned to agents, responsible for their fulfilment, and to appropriate metrics, that allows tracking, analysis and further interpretation.

In the following two sections a brief overview of the two modelling approaches is presented. Then in the fourth section an example of combining the two approaches is examined, and its main advantages are outlined. The fifth section is a case study on applying the combination of two approaches in the context of an open source software project. The authors demonstrate how the combination can be used for an actual project, and then list possible benefits and drawbacks to using the approach for the purposes of documenting requirments for an open source software (OSS) project.

2 Goal, Question, Metric (GQM) Method

One way for an organisation to have some valuable insights regarding itself, its assets and processes is to conduct some appropriate measurements and interpret them correctly. The question of what comprises appropriate measurements is critical. The GQM method helps one to arrive at those measurements, provided there is a set of specific goals for that organisation [2]. To get from each specific goal to a measurement the method defines three levels (Fig. 1), each representing a step on the way from the goal to a metric:

- *Goal:* Conceptual level, where each of the specific goals is represented.
- *Question:* Operational level, at which questions are posed against the goal in order to assess its achievement and identify some of its aspects through which the progress can be observed and measured.

– *Metric:* Quantitative level, at which appropriate metrics is selected to answer the questions posed at the operational level in a quantitative manner.

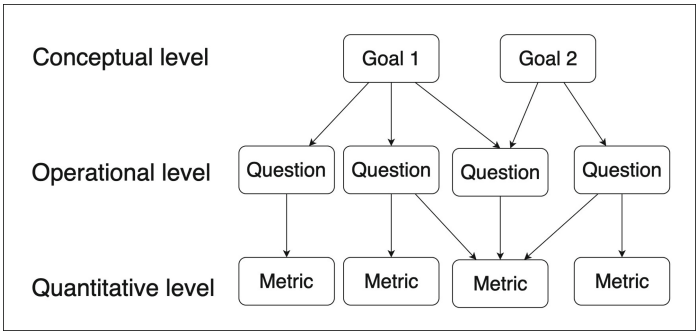

Fig. 1. Levels of the GQM model

Following is an example of the GQM process using a simple example of a web application of a company selling airplane tickets. The process starts with a specific goal. It has to be formulated in a particular way and be associated with an object, which in the context of software development could be an artefact, process or resource. Within the goal a purpose is defined, together with a particular quality issue of the object and a viewpoint from which it is looked at. The example goal is to improve usability of the ticket booking feature from the point of view of a person booking a ticket (end user). The object of the goal is the ticket booking feature. The purpose is to improve, the quality issue is the usability and the viewpoint is that of a person booking a ticket.

Once the specification of the goal is ready, questions can be posed against the goal and metrics can be proposed to answer the questions. Looking at the example goal above, its purpose and quality issue one can deduce following questions:

– What are the completion rates of the feature?, i.e. what portion of users actually accomplishes booking a ticket.
– What is the average user task time?, i.e. how much time does it normally take a user to accomplish booking a ticket.
– How satisfied are the end users with the feature's usability?

Each of the questions can have one or several metrics answering it. The whole GQM can be presented in a tabular format [2], as in the Table 1. Often, there is a set of specific goals and a GQM is produced for each one of them. Evidently, having these set of specific goal is what makes deriving metrics possible. The question arises then of how the set of specific goals for an organisation is actually produced.

Table 1. Exemplary GQM

Goal	Improve the usability of the ticket booking feature from the point of view of a person booking a ticket (end user)
Purpose	Improve
Quality issue	Usability
Object	The ticket booking feature
Viewpoint	End user
Question	Q1: How long does it take a user to complete the booking?
Metrics	M1: Average task time
	M2: Standard deviation from the average
	M3: Percentage of cases exceeding the average maximum limit
Question	Q2: What are the completion rates of the feature?
Metrics	M4: Percentage of successfully completed bookings
Question	Q3: How satisfied are the users with the feature?
Metrics	M5: Average subjective rating by users grading the feature

In fact, the team constructing GQMs needs to accomplish a thorough work of understanding the context of the organisation itself, its mission, strategy, policy and high level goals. The team also needs to understand the local context of the object being measured, which can be a product or some process within a project. Finally, viewpoints of various stakeholders need to be taken into consideration to serve as another important element formulating the goals. The whole process involves interviewing stakeholders, analysing the environment and collaboration between the corporate management, project and GQM team. The derived goals need to be analysed with regards to their relevancy and relation to the high level goals [2].

Evidently, deriving the specific goals is a lengthy and nontrivial process. Therefore presenting only the specific goals and their corresponding GQMs is insufficient to understand the greater context and the relationship between specific goals, the environment and strategic high level goals of the organisation. So the question arises, what techniques and approaches can be used to simplify the process of deriving the set of specific goals? Furthermore, can the context and its relationships with the specific goals be represented in a concise and comprehensive manner?

Yet one more important aspect related to the metrics is related to the actors involved. Does the measurement process involve personnel, devices or software? Who or what is responsible for accomplishing the goal and gathering the measurement data?

The GQM's tabular format above is in fact only a concise output of the work preceding it, while the context and the actors involved with the system can be conveyed through text or by using other modelling approaches [2]. The KAOS

goal modelling approach provides tools and techniques necessary to reflect the strategic context and actors involved. In the following section, KAOS and its tools are introduced, and the benefits of combining it with GQM is elaborated upon based on an example.

3 Goal Modelling: Keep All Objectives Satisfied (KAOS)

Goal modelling is an approach widely used in requirements engineering, and it allows to capture requirement by analysing a software system's high-level goals and contextual environment.

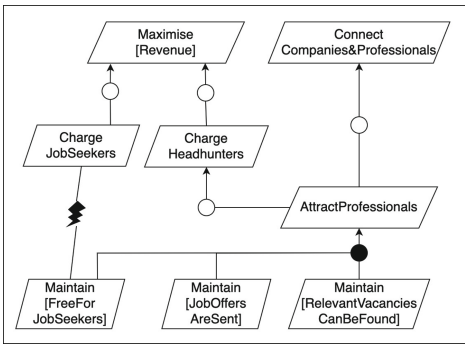

Fig. 2. Goal model for a professional social network

The specific approach being introduced here is KAOS. The abbreviation stands for Knowledge Acquisition in automated specification or Keep All Objectives Satisfied, but within this paper it will be interchangeably referred to it as KAOS and the goal model. The goal model uses expressive and concise diagrammatic notation to convey the flow and refinement of a system's high level strategic goals into lower-level technical goals. Moreover, it can conve various complex relations between the goals at different levels [1].

Figure 2 demonstrates a simple example of applying goal model in the context of building a social network for professionals. This fairly simple diagram contains several aspects that worth noting:

1. The top level element are the high level strategic goals of the system being built. There are two high-level goals: to generate revenue, and to connect companies with professionals. The strategic goals are then broken down into finer-grained goals in a top down fashion, such that the goals on the third level in the diagram are actually describing the technical design objectives of the system.
2. Relationships between goals at different levels can be identified and denoted. The first type is the relationship between a parent goal and its direct children.

Does the system need to achieve all of the child goals in order to satisfy a parent goal, or the child goals are actually alternatives, and fulfilling only one of them is sufficient? The KAOS notation uses something called or- and and-refinement to expressing that, and the refinements are denoted by the blank and solid circles serving as connection between the goals (Fig. 3).
The solid black circle denotes that all child goals have to be fulfilled, and is call complete and-refinement, as in $AttractProfessionals$ goal and its children [1].

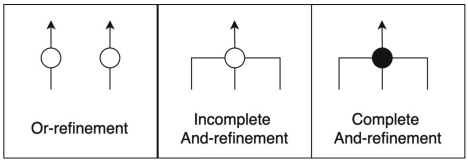

Fig. 3. Refinement types

If there are two or more blank circle coming out of a single goal, like in the case of the $GenerateRevenue$ goal, it denotes that the children connected through the circles are alternative ways of satisfying the goal. It is referred to as or-refinement.
The case in-between, not present in the example diagram, is referred to as incomplete and-refinement, which has the meaning that only some and not all the child goals need to be achieved in order to satisfy the parent goal. It is denoted by a single blank circle pointing an arrow at the parent goal, with several lines coming out of it and connecting the circle with the child goals.

3. Having system goals in one place, allows one to analyse the goals and to identify and denote the conflict between the goals on different levels. A lightning bolt shape connection is used for that, as can be observed in the case of two apparently conflicting goals from the example: $ChargeJobSeekers$ and $Maintain[FreeForJobSeekers]$. Together with identifying conflict, the goal model approach suggest various mechanisms of resolving the conflicts [1].
4. As can observed, the goals on the third level have the prefix Maintain before them. This comes from the notion of behavioural and soft goals used in KAOS. Behavioural goals are preceded by prefixes such as Achieve, Maintain and Avoid, and denote some logical constraints on the behaviour of the system. Achieve means that the goal has to be achieved at some point, and often comes with a conditional close dictating when it should be achieved. Maintain defines that some desirable state needs to be up-kept in the system at all times, while avoid means the opposite, i.e. that some state must never occur. Soft goals, in turn, denote other less stringent conditions on the system, and use prefixes such as maximise, minimise. This technique of defining various types of goals provides an expressive and fairly flexible tool of conveying constraints on the system and its environment through the goal statements.

For some goals, a different refinement strategy can be used. Instead of defining subgoals, an obstacle preventing the goal from being achieved can be defined. Figure 4 represents an example of such a case.

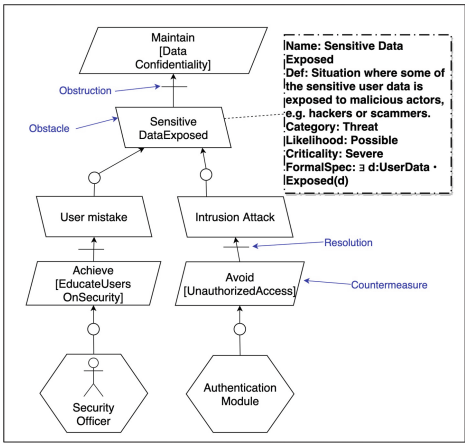

Fig. 4. Goal model obstacles and agents

There are several important aspects in this example too:

1. A goal common to most software systems dealing with personal user data is keeping the user data confidential. An obstacle obstructing that goal, i.e. preventing its satisfaction, would be exposure of some sensitive user data. The obstacle is further refined into two alternative sub-obstacles, and then countermeasures for both of them are found. According to the notation the crossed arrows denote obstruction and resolution relationships. An attentive eye will also notice that obstacles are represented inside a horizontally flipped parallelogram, which signals that obstacles are opposite of goals.
2. There is a table on the right of the *SensitiveDataExposed* obstacle, providing additional details such as the definition, category, likelihood, criticality and even a formal specification expressed using predicate logic. Such tables can be provided for any goals or obstacles, that need an extra level of detail or clarification.
3. Yet, the aspect of biggest interest is the fact that the lowest level goals are attached to hexagons, which represent agents. Agents can be software modules, individuals or devices. Individuals are normally denoted with a human figure, while non-human agents are denoted with titles only. This is one of the key aspects of the goal model, since it allows to attach each goal on the lowest level to an agent, thus connecting the strategic goals with the system or environmental elements responsible for satisfying them. Normally, each low-level goal will have one agent responsible for fulfilling it. In cases when a goal has several agents, the possibility of further refining the goal should be

considered. On the other hand, if an agent is responsible for too many goals, the refinement and better definition of the agent should be considered too.

It is worth noting that the examples in Fig. 2 and Fig. 4 had both business and human related goals (e.g. *GenerateRevenue* and *EducateUsersOnSecurity*), which were included on purpose. The purpose of including such goals is to demonstrate that the goal model approach can be used in various context, not limited to technical requirements specification. In fact goal modelling can be used to reconcile and align technical project decision with the business goals [4]. The goal model diagram in its essence is a mind map, which allows one to visualise the goals and the context, analyse how the elements relate and synchronise between each other, expand the solution, spot conflicts and obstacles.

To summarise, goal modelling allows one to define strategic goals, expand and refine them into lower-level finer-grained goals, and express their breakdown structure and flow. The concise and expressive notation tools such as or-/and-refinements, conflict and obstacles identification allow one to express various relations between goals and subgoals, and identify possible barriers on the way of achieving them. The notion of agents allows to explore and define the actors involved with the system, and to allocate and assign their responsibilities. Altogether, it comprises a powerful tool allowing one to convey the strategic and environmental context, technical and business constraints and actors involved.

4 Combining GQM and KAOS

In this section, an example of combining the two approaches is presented based on the case of the ticket booking company's web application from the second section.

Before building a GQM, the company's high level strategic goals need to be analysed and understood. Increasing revenue is a high level goal shared by many companies. The techniques and tools introduced in the previous section will be used to derive lower level specific goals from the strategic goal, and reflect the thought and derivation process (Fig. 5).

As was mentioned in the previous section, the goal model diagram is normally constructed in a top down manner. Despite the fact that the diagram in the example (Fig. 5) is of relatively small size, there is a number of techniques used to describe various relationship between the goals:

1. The top goal has two subgoals connected by using an or-refinement. The or-refinement signals that the two subgoal are alternatives, i.e. the company can either try to $IncreaseNumberOfBookings$, which is the central function of the web application, or try to $SellBannerAdvertising$ for third parties and extra services.
2. Following the refinement of the $IncreaseNumberOfBookings$ goal, one can see that it has two subgoals itself. Thus in order to have more bookings the company needs to acquire new customers or retain the existing ones. The corresponding goals are $Maintain[CustomerAcquisition]$ and $Maximise[CustomerRetention]$.

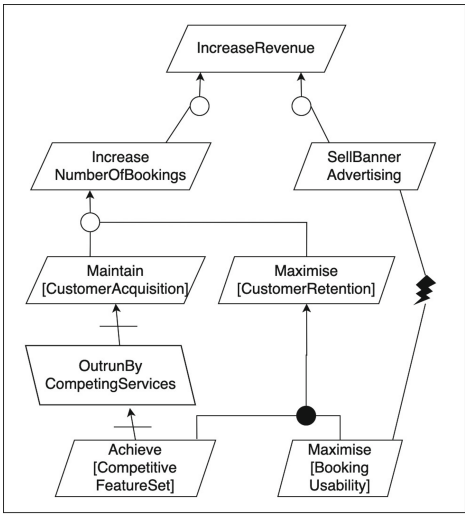

Fig. 5. From strategic to specific goals

3. At the bottom level, there are two goals: $Achieve[CompetitiveFeatureSet]$ and $Maximise[BookingUsability]$. Both of them need to be fulfilled to satisfy the parent goal of retaining customers. The relationship between the parent goal and two subgoals is conveyed through the and-refinement, denoted by the solid black dot connecting the two levels.
4. Two more aspects that should be noted are the following: the of maintaining the CompetitiveFeatureSet is the countermeasure to the obstacle of losing customers to competing services, and the conflicts is identified between the goals BookingUsability and SellBannerAdvertising.

The diagram in the Fig. 5 describes the process of arriving at the specific goal of maximising the booking feature's usability, the GQM of which has been constructed in the second section of this paper, and has another specific goal next to it, for which a GQM can be constructed in a similar manner. The most important thing to note, is that the goal model diagram provides a concise and comprehensive way of visually conveying the context and derivation process of the low-level specific goals used for constructing GQMs.

Interestingly, GQMs can also be constructed for higher level goals, in particular for $Maximise[CustomerRetention]$, $IncreaseNumberOfBookings$ and $Achieve[CustomerAcquisition]$. For example, the customer retention goal calls for a question of "Do the users come back to make repeat bookings through the service?", for which a simple metric can be provided as an answer, namely "Percentage of returning users making repeat bookings". The fact that the GQM can be used to measure the achievement and state of goals at various levels, implies that the GQM metrics can enhance the goal model approach, especially when it comes to soft goals. The very formulation of soft goals using verbs such as

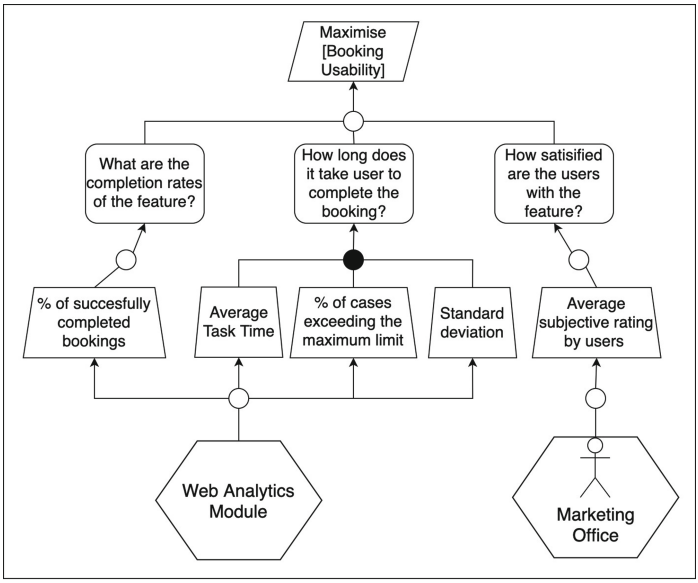

Fig. 6. Integrating GQM into the Goal model

Maximise, Minimise and Optimise calls for metrics to be used in order to track the state of the system, and declare concrete satisfaction criteria for these goals.

Another important point is that GQM can be integrated into the diagrammatic notation of the Goal model. Figure 6 exemplifies the case of the GQM constructed in the Sect. 2.

The notation introduced is a rounded-rectangle for questions and a trapezoid for metrics. This way the GQM can be integrated into the diagrammatic format of the goal model. The advantage of it is two-fold. Firstly, the exact placement of the GQM within the greater context can be observed. Secondly, and more importantly, the agents responsible for gathering particular metrics can be identified and denoted, as in the case of ascribing the user satisfaction metrics to the Marketing Office, and the rest of the metrics to the Web Analytics module.

5 Applying the Technique to an Open Source Project

The combination of the two modelling approaches has been tried and tested by the authors on an open source software project called KinApp [8]. KinApp is a mobile application that helps users to keep in touch with their family and friends while being away.

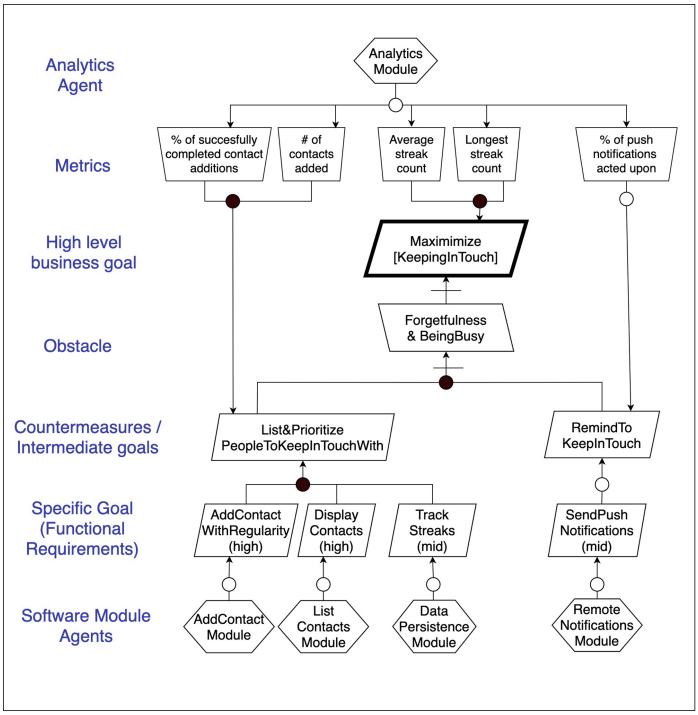

Fig. 7. KinApp's combined GQM and KAOS models

During user research stage preceding requirements elicitation, the authors were able to establish that there is a segment of potential users who would like to keep in touch with their relatives, but for various reasons are not able to do so. The most common reason named was forgetfulness due to being busy and having other priorities to take care of. Based on these findings, the authors developed and implemented a concept for the KinApp mobile application. It is a simple reminder app where users can add people they want to keep in touch with, and configure the desired frequency of contacting for each of them. The application will send reminders, and keep track of streaks (unbroken sequence of successfully contacting the family member or friend on a regular basis).

KAOS approach is used to produce specific goals, corresponding to functional requirements within the app. GQM in turn provided a way of measuring the effectiveness of the tool in achieving the high and intermediate level goals by providing measurement metrics.

Figure 7 demonstrates the process of applying KAOS and GQM to the high level goal of $Maximize[KeepingInTouch]$, right in the middle of the graph. As has been established during user research, the most common preventer is forgetfulness due to having other responsibilities. This can be represented as the $Forgetfulness\&BeingBusy$ obstacle within the goal model. The two

countermeasures are 1) having a prioritized list of people to contact and 2) to remind users to keep in touch. We introduce them as the two intermediate goals. The prioritized list represents the main functionality of the app. It will require several essential features:

- Adding people to list
- Configuring regularity of contact, i.e. how often does the user want to contact each person
- Displaying the list of added people

This translates into two specific goals with the highest implementation priority: *AddContactWithRegularity* and *DisplayContacts*. Another intermediate goal *RemindToKeepInTouch* translates into a specific functional requirement of sending push notifications to the user as reminders that sufficient time has passed since the last contact with a particular person.

Table 2. GQM for $Maximize[KeepingInTouch]$

Goal	Maximize keeping in touch with family and friends from the point of view of a person affected by forgetfulness and being busy (end user)
Question	Q1: Is the person able to follow up with the configured regularity for each added contact?
	Q2: What is the longest streak (an unbroken sequence of successfully contacting an added person in accordance with the regularity)?
	Q3: What is the average streak for all added contacts?
Metrics	M1: Longest streak count
	M2: Average streak count

GQM is then applied to make the above goals measurable. Table 2 demonstrates the process of arriving to metrics for $Maximize[KeepingInTouch]$.

Other metrics are similar to the ones demonstrated in the previous sections, and for the sake of keeping the example concise will not be delved into. Noteworthy, the introduced streak metrics necessitate a new functional requirement of having to track streaks for users. The corresponding specific goal is placed under the intermediate *ListAndPrioritize* goal, due to being closely related to the functionality within the goal. This points to the fact, that GQM and KAOS can complement each other and help to reveal implicit requirements during an early stage of a project's timeline.

Importantly, the resulting diagram also demonstrates the reasoning behind each of the specific goals, and serves as a concise and readable way to present the requirements for the project to both end-users and developers willing to join the effort. This can be especially valuable in the context of an open source project, as readability of requirements is one of the important characteristics for the success and adoption of a project by the open source community [9]. Moreover,

the presence of high-level goals can allow the community to suggest alternative ways of achieving the goal, and could facilitate ideation process.

On the other hand, a possible drawback is that maintaining the diagrams relevant can become difficult as the project evolves, which is a common reason the open source projects tend to keep a minimally possible amount of informal documentation [9].

6 Conclusion

In summary, the GQM and KAOS goal model can be used together to derive and represent a system's high- and low-level goals, actors, relationship between them and appropriate metrics to measure various important aspect of the system.

The goal model can facilitate and reflect the extensive process of mapping the strategic goals and concrete specific goals of the system, which are a prerequisite to implementing GQM.

The GQM, in turn, provides a quantitative perspective on this mapping in the form of appropriate metrics rooted in the system's strategic context, and can provide quantification for goals at different abstraction levels. The complementary union of the two models enables a more comprehensive and encompassing approach to defining, analysing and refining software systems.

The combination of two approaches has been tried in the context of an open source mobile application, and allowed the authors to arrive at a set of specific functional requirements and metrics to measure the effectiveness of these functionality in achieving the high level business goal.

Moreover, the inter-complementary effect of both approaches was demonstrated, such that the KAOS serves specific goals to GQM, which simplify producing metrics. Whereas GQM facilitates discovery of implicit low-level goals through the questioning process of making high-level goals measurable.

Potential benefits and drawbacks of applying the combination in the context of an open source project were suggested. The benefits include increased readability and transparency of requirements to the open source developers community and the ability to see the origin and intent of each particular functionality. The latter can allow to suggest alternative or complementary ways of achieving high goals. A possible drawback is the difficulty of maintaining the diagrams and prose up-to-date as the project progresses and evolves. Future studies of applying the two models and those similar to them could allow to further analyze and identify the advantages and disadvantages of using the suggested combination.

References

1. Van Lamsweerde, A.: Requirements Engineering, pp. 287–349. Wiley, Chichester (2013)
2. Basili, V.R., Caldiera, G., Rombach, H.D.: The Goal Question Metric Approach. In: Encyclopaedia of Software Engineering, Wiley (1994)

3. Ullah, A., Lai, R.: Modeling business goal for business/IT alignment using requirements engineering. J. Comput. Inf. Syst. **51**, 21–28 (2011)
4. Ellis-Braithwaite, R., Lock, R., Dawson, R., Haque, B.: Towards an approach for analysing the strategic alignment of software requirements using quantified goal graphs. Int. J. Adv. Softw. 119–130 (2013)
5. Ivanov, V., Reznik, A., Succi, G.: Comparing the reliability of software systems: a case study on mobile operating systems, Innopolis University (2017)
6. Ivanov, V., Succi, G., Pischulin, V., Yi, J., Rogers, A., Zorin, V.: Design and validation of precooked developer dashboards (2018)
7. Basili, V.R., et al.: Determining the impact of business strategies using principles from goal-oriented measurement, Internationale Tagung Wirtschaftsinformatik, Books OCG, Vienna, Österreichische Computer Gesellschaft, Austria (2009)
8. KinApp, Github Repository, KinApp Mobile Open Source Application. https://github.com/nurlingo/adabi, 12 Mar 2020
9. Scacchi, W.: Understanding the requirements for developing open source software systems. IEE Proc. Softw. **149**(1), 24–39 (2002)
10. Scacchi, W.: Understanding requirements for open source software. In: Lyytinen, K., Loucopoulos, P., Mylopoulos, J., Robinson, B. (eds.) Design Requirements Engineering: A Ten-Year Perspective. LNBIP, vol. 14, pp. 467–494. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-540-92966-6_27