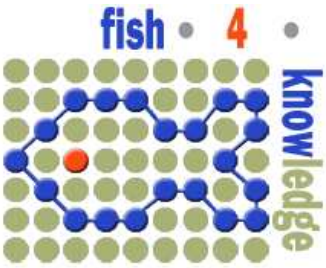

Fish4Knowledge Deliverable D5.1

Component Interface and Integration Plan

Principal Author: B.J. Boom, R. B. Fisher
Contributors: UEDIN
Dissemination: PU

Abstract: The component interface and integration plan is one of the deliverables of the Fish4Knowledge project. The purpose of this deliverable is to describe how the individual components of the partners have to cooperate. This plan gives the grand design of the entire system and the role of the individual components in this system. For the individual components, we have defined the purpose, input, output, possible method of evaluation and possible failure. The development of the individual component will be entirely the responsibility of the different partners. Another tasks of the partners is to maintain the descriptions of their components using the wiki. Issues that are important for the integration of the different components are also described in this plan. The plan also contains milestones which indicate when important steps in the development of the entire system should be finished.

Deliverable due: 3 Month

1 Introduction

The Component Interface and Integration Plan will help the partners in the Fish4Knowledge project to cooperate. The partners in the Fish4Knowledge project are each responsible to create certain output information. In most case, this output is data that other partners depend on. The RDF/XML Datastore Definition together with this plan will help the partners to cooperate in this project. This plan give the partners the freedom to develop their own component(s). We only specify the input and output parameters necessary for the entire system to work properly.

The following components are defined within this project:

- Fish Detection (UNICT)
- Fish Tracking (UNICT)
- Fish Description (UEDIN)
- Fish Recognition (UEDIN)
- Fish Clustering (UEDIN)
- User Interface (CWI)
- Query Engine (CWI)
- Workflow (UEDIN)
- Database (NCHC)

We divide this document into three section, namely component interfaces and component integration. The Section 2, we will discuss the cooperation between components in order to create the entire system. In this section, we firstly explain the grand design. Secondly, we will give a definition for the component. Thirdly, we will discuss the evaluation of the entire system and the separate components. Fourthly, we perform risk management by thinking of possible failures. In the Section 3, we give a detailed description of each component mentioning the previous issues. The Section 4 will discuss the issues that are important to adding the different components together. These issues are the platform on which the component runs, the development environment and the dependencies between components which have to be monitored. We also added a timeline with some milestone, giving an indication when we expect certain deliverables in the development of the system.

2 Component Interfaces

2.1 Grand Design of Interaction

The main idea is communicated by means of the storage facility(s), this means that the data that is processed by the components is available to all partners in the project, but more importantly to the end-user. The idea is that all components write their output to a storage facility. There will be a component (database component) that will collect and store the data, but also allows

us to query and retrieve the data again.

There will be a lot of data, so we probably have to use a distributed storage facility. This data also comes in different formats, like video, image, ontologies, for which we can use different sort of databases (SQL, triple stores), that can deal with the different formats. However, by using the database component, we intend to give the other components a simple interface to the storage facilities without having to worry for instance about storing information in a distributed manner or different interfaces to retrieve different kind of information.

In this section, we will give the overview of the system and how the components interact with each other by means of the storage facilities:
First the videos from the underwater webcams are store in the storage facilities, the Fish Detection component will get the videos out of the storage facilities and will find the fish and label their location in the frames (fish location). The Fish tracking allows us to follow fish in multiple frames and also can contribute in behaviour studies of fish. The Fish Detection/Tracking components will again store the obtained information (for example the fish locations) in the storage facility. The Fish Description component will add certain descriptions to the stored fish (like the kind of tail the fish has, or the colour of the fish). The Fish Recognition component will try to determine the exact species label (or family label) and will store this. Fish Clustering allows us to determine if fish are very similar to each other, but this is especially interesting for finding outliers (not common species or even new species). The Query Engine is able to retrieve all the information previously stored in the storage facilities, and for instance count the number of species X during the month December. The User Interface represents the information to the users, but also gives the user an interface to search through all the information in the storage facilities. The workflow component will check which system resources are available to perform new jobs. For instance, it will keep track of the videos which have not been processed yet by the Fish Detection/Tracking component and the fish which have not been processed yet by the Fish Description/Recognition component. The output will probably be that it starts these processes if there are resources available (like memory and CPUs). Furthermore, it should be able to handle special requests by the user interface, running different settings of computer vision components (fish detection, tracking, description, recognition and clustering).

In Figure 1, we show a schematic representation (UML Component Diagram) of the entire system. The components basically have interfaces and sockets and the information flow is given by the arrows. This schematic gives a rough overview. In Section 3, we define the possible inputs and outputs in more details. Notice that most components connect with the database component, because this will both allow to store the output of the components and provide input to the components.

2.2 Definition of a Component

For the Fish4Knowledge project, each of the partners has to create its own component(s). The exact implementation of the component is up to the partners, however the information flow between the partners has to be defined in order to cooperate with each other. Each of the components has an input and an output, which will be defined for each component separately.

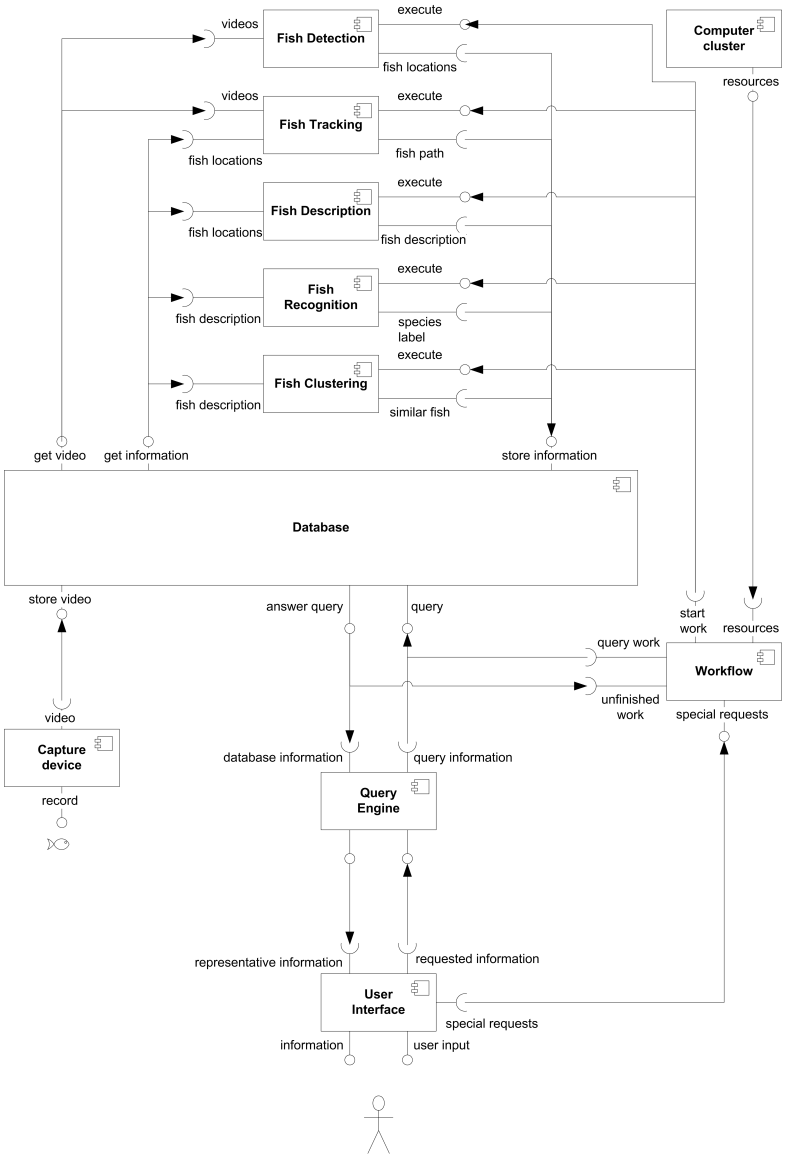

Figure 1: UML Component Diagram, showing the input and output relations of the different components

In order to make a distinction between the importance of certain inputs and outputs, we will label the output as: minor, feature, suggested, and necessary. The focus in the first stage should be on the necessary and suggested outputs. If those are finished other outputs can be added as new features. The users of the system (biologists) can also ask for different features. The separate component are discussed in appendix 1.

Because the Fish4Knowledge project is under development, we suspect many changes in the different components. Because we do not want to lose information, every component gets a unique identifier (by means of a lookup table, purpose and version can be retrieved). The component has to use this identifier while storing information. This means that we can also track information from old components and versions. We can easily add new components for the detection and recognition, or add newer improved version of the component. Notice that this offers a lot of flexibility but also has a drawback. The drawback with this approach is that we store all the information, also older versions which are created by components that still contain some bugs, but it can happen that a marine biologist used these components to get retrieve a certain set of data, so removing information is probably more undesirable. In this case, the replacement of some description is basically inserting newer description with a newer version number of the same component. (Note: we do not show all stored information to marine biologists especially in the early development stage, there will be a test and live environment which allows developers to first test their component, see Section 5.1).

2.2.1 Link to RDF/XML Datastore Definition

The exact Datastore Definition in the databases will be maintained by CWI. They will also state which partner is responsible for creating the information. We propose to use the same priority checks as are used in this document (minor, feature, suggested, and necessary). This document already provides a good basis for these definitions and the CWI can fill in the details. The query interfaces of both the fish detection/tracking and fish recognition/description component can be relatively simple (SQL interface), while CWI will probably develop their own query engines.

2.3 Evaluation

The performance of the entire system and the separate components is very important. In order to monitor this performance, we need to be able to evaluate the system and the separate components. Firstly, we will discussed the evaluation of the entire system and secondly discussed the evaluation of the different components.

Entire System: The evaluation of the entire system will be performed by the marine biologists. There are two major evaluations planned in this project. The first evaluation of the entire system will be perform around the 23 month of the project. A similar second evaluation is performed at the 34 month of the project (last months). During this evaluations, the system will be tested by scientific users volunteers that will participate in an interactive session for a single day. In the second evaluation session, we will takes into account the comments of these users from the first session. In order to make the session with the scientific users volunteers successful, an evaluation plan has to be developed. In this evaluation plan, it is important that

we will highlight the different sides of the project, making the scientific users volunteers aware that there is more than the interface. This allows us to receive also comments on aspects like the computation/query time, computer vision aspects. The University of Edinburgh is responsible for the evaluation session, but CWI should also be involved, because they already interviewed the marine biologists and they are responsibility for the user interface. Before the biologists test the system, the system performance should also be properly tested, meaning that we need in both session an analysis of the query times, system loads, frame rates, processed data.

Components: The entire system exists of different components. These components have to be evaluated in different ways. The creators of the components probably know the best manner to evaluate their components. In order to contribute in their scientific fields, they have to evaluate their components anyway. In this document, we already give some suggestions for each component, but it is up to the creator to determine an evaluation plan for their components. This evaluation plan should be finished in the 17 month of the project. These separate evaluation plans of each component are used as input to construct the evaluation plan of the entire system. The evaluation plan can be put on the wiki pages of the project. Based on the user's comments, new goals are set for the different components allowing everybody to improve their components on specific issues that occurred during testing. (For each component, we added an example of how the components can be evaluated, see Section 3. We, however, believe that the creators probably have more experience in their fields, which allows them to ignore these examples and add their own methods for evaluation).

We also evaluate the component interfaces (Section 3) described in this plan using the use cases (in the Project Plan), see Appendix A. Related to this we performed an estimation of the system resources necessary for this system, which is placed in Appendix B. Input from the other partners on these subject are important to understand each other better and to estimate the risks within this project.

2.4 Risk Management

In order to understand the risks in the project, we have come up with possible failures of the different component and how these failures will effect the rest of the system, see Section 3 for examples for each component. For the creators of the individual component, checking and contributing to these list of failures will help us to understand the difficulties which can be expected. This document will also be part of a wiki page, allow all partners to contribute in this process. Based on a list of failures and problems that exists in the 18 month of the project, we are going to discuss the final design decisions for the first prototype. This allows us to have a first system ready in the 24 month. In the risk analysis of the final system also the system resources will play a important role, a first discussion can be read in Appendix B.

3 Individual components

3.1 Fish Detection Component

3.1.1 Purpose:

The purpose of this component is to detect the fish in the video stream. The detection will basically locate the fish in each frame. After locating the fish, the contour of the fish will be saved in the databases. The Detection Component can also save the background image, which can be used for compression and fast scene recovery. Another task of this component is to describe the scene. For examples, it will detect if it is dark or light, how much pollution (green,dirt) is in the water. It can even be able to detect dirtiness on the camera lens. Techniques like deconvolution might be able to correct for dirt on the lens given older recordings of the scene where the lens is still clean.

3.1.2 Input: (Videos)

Video streams (necessary)

3.1.3 Output: (Fish Location)

Fish (necessary)
Fish location for each frame (x,y,binary mask or contour,date,time) (necessary)
Scene information (suggested)
Camera information (suggested)
Camera correction for dirtiness (for deconvolution purposes) (minor)

3.1.4 Evaluation:

The fish detection can be evaluated using a labelled dataset of fishes, where we can determine the false positive and negative rates and make a ROC given certain thresholds. It can be useful to also determine if there are difference in the ROC curves of different species or dates/times.

3.1.5 Possible Failures:

False positives: Fish is detected where no fish is present. This results in strange recognition results or a outlier in recognition. Fish recognition can detect parts of the false positive and throw them away, but this is not the task of the fish recognition component.
False negatives: Fish is not detected while it is present. This might give inaccurate result in statistics especially if the detection rating are different for certain species, but might be correctable with ground truth information on smaller subsets.
Incorrect scene information: Incorrect scene information can make it more difficult for the workflow component to activate the correct components for the further fish detection and recognition. For instance, a green correction filter has to be applied if the scene information was correct. This will result in inaccurate colour descriptions for the fish, making the recognition more difficult.

3.2 Fish Tracking Component

3.2.1 Purpose:

The fish tracking will follow the fish in the video, labelling at which position and in which direction the fish is going. It also provides information on how long the fish was visible in the image. More valuable output can be the interaction of the fish in relation with other fishes in the video, like analysis if a fish is pursued by another fish. Other events that can be detected are eating, resting, hiding, fighting, mating, schooling, panic. Another possibility is to make clusters of the behaviour patterns of fish to see if new behaviours can be described. (Notice that it is very likely Fish Detection/Tracking components are combined for performance reasons).

3.2.2 Input: (Videos)

Video streams (necessary)

3.2.3 Output: (Fish Location)

Fish path (necessary)
Interaction of fish (feature)
Clusters of fish behaviour (feature)

3.2.4 Evaluation:

The fish tracking can be evaluated using labelled data and whether or not it is able to keep track of the fish. Fish behaviour labelling has to be performed by biologists in order to both learn and evaluate this. It might also be interesting to look and visualise clusters for behaviour analysis.

3.2.5 Possible Failures:

Path tracking failures: Fish is not followed correctly, usually caused by fish overlapping in images. This can result in for instance first following a clownfish and afterwards following a shark. Depending if the fish recognition methods use only the best frame or multiple frame different problems can be expected: In case of the best frame, you get a false negative. In case of multiple frame, strange fish descriptions can appear which will result as an outlier in the final fish recognition.
Fish interaction is incorrect: This gives the users an incorrect result when there is a query for certain behaviour or when the users request statistics.

3.3 Fish Description Component

3.3.1 Purpose:

This component describes the fish found after the detection stage. In order to describe a fish, multiple features are selected. This can be features that can be understood by the users, like the number of fins, the kind of tail, the colour. Probably, there will also be features that are important from a computer vision point, but these features might not be clear to users. Examples

are Gabor Filters which are able to measure texture in fish. Because the fish can be described in several ways, it will be important to cooperate with marine biologists to understand which descriptions are important for them.

3.3.2 Input: (Fish location)

Fish (necessary)
Fish location for each frame (x,y,binary mask,date,time)(necessary)
All frames containing the fish (necessary)
List of features/descriptions (suggested)

3.3.3 Output: (Fish description)

For each fish appearance, a vector of description values (necessary)
For each fish appearance, the descriptions suggested by marine biologists (suggested)
For each fish description suggested by marine biologists, a certainty value (suggested)

3.3.4 Evaluation:

The fish description is usually evaluated together with the recognition. Given that we know that certain species have certain descriptions, we can also evaluate these descriptions. In order to do this evaluation labelled data is necessary.

3.3.5 Possible Failures:

Failure in description values: It is possible to have incorrect values in the description value, usually computer vision uses a large set of description values to become robust against a single failure for recognition and clustering. Certain failures have an effect on both recognition and clustering, so searching for robust features or higher level features in those cases is necessary.
Failure in descriptions suggested by marine biologists: These features have to be even more robust than the description values, because they can be directly used for querying. With these descriptions, a certainty value can be added to indicate how good the description is observed. If marine biologists use multiple filter operations to specify their queries, this can help overcome failure in a single description.
Incorrect segmentation/registration: It is possible to find a tail where the head is. This can also result in outliers in the vector of description values and incorrect species recognition and clustering.

3.4 Fish Recognition Component

3.4.1 Purpose:

The Fish Recognition Component will recognise the species or family to which the fish belongs. Because the fish is visible in multiple frames, these frames might have to be combined to obtain more information about the fish. We also have to select the frame (time and place in video) that contains the best appearance of the fish. This can be done using the contour, but also based on

the number of features found by the fish description. If the fish is too far from the camera, it is possible that we can only determine the family to which the fish belongs and not the precise species. In computer vision, recognition of objects is a difficult task with a lot of uncertainties. These uncertainties can be expressed in probabilities or percentage, allows us to communicate a certainty value that we correctly recognised the fish. Another feature of this component can be that marine biologists can make their own fish model, based on pre-labelled images or high level features to search for certain specific fish.

3.4.2 Input: (Fish location)

Fish (necessary)
Fish location for each frame (x,y,binary mask,date,time)(necessary)
All frames containing the fish (necessary)
Fish models (suggested)
Specific Fish models/filters (feature)
For each fish appearance, a vector of description values (necessary)
For each fish appearance, the descriptions suggested by marine biologists (suggested)
For each fish description suggested by marine biologists, a certainty value (suggested)

3.4.3 Output: (Fish labels)

For each fish a genus,family or species name (necessary)
For each fish a certainty score that family/species is correct (suggested)
For each fish the best appearance (suggested)
For each fish a score how good appearance is (suggested)
Output of Specific Fish models/filters (suggested)

3.4.4 Evaluation:

Fish recognition can be evaluated based on labelled sets of fish. We can determine if the label is correctly found which allows us to show ranking plots. We can also determine a similarity score given that a certain class is correctly found, which basically allows us to compute the false positive and negative rate. We are able to compute ROC curves for these similarity scores. It can also be interesting to look at the recognition rates of the different species, some species can be easier to recognise as others. Notice that a ground truth set of labelled fishes is difficult to obtain, because a marine biologist has to label fishes which costs a lot of time. Tools for labelling which perform perdiction about the labelling can make this task more easy for biologists.

3.4.5 Possible Failures:

Incorrect species: It is possible to label the species incorrectly, this can result in inaccurate statistical information, a certainty score of the label can already correct for this. Furthermore, some measurements on the appearance of the fish (resolution, blur, etc) can indicate how difficult the fish recognition was. It is also possible to run statistics on only high quality fish in the database.
Incorrect best appearance: If it finds a second best appearance, this will be not a big problem. In worse cases, however, it can influence the fish recognition results and more important the

user interface. For instance, if user would like to view the fish, this user can not be bothered with bad quality images of the fish.
Mistake by running Specific Fish models/filters: This gives outliers in the queries of the marine biologists. Flexible training procedure where users can select the outliers and build new filters can be a solution which has to be considered.

3.5 Fish Clustering Component

3.5.1 Purpose:

This component allows us to make clusters of fish that are very similar to each other. By making these clusters, we also determine the fish that are not inside these clusters (outliers). These outliers can be very interesting for marine biologists, because this methodology enables us to recognise fish that are unknown to us. Another possibility is to allow marine biologists to specify filters to search for certain interesting properties certain fish might have. This can allow marine biologists to determine how certain fish species evolve, by looking at different variations of for instance tails of a single fish species.

3.5.2 Input: (Fish location)

Fish (necessary)
Fish location for each frame (x,y,binary mask,date,time)(necessary)
All frames containing the fish (necessary)
Fish models (suggested)
Specific Fish models/filters (feature)
For each fish appearance, a vector of description values (necessary)
For each fish appearance, the descriptions suggested by marine biologists (suggested)
For each fish description suggested by marine biologists, a certainty value (suggested)

3.5.3 Output: (Fish clusters)

Similar Fish (feature)
Cluster name (feature)
Interesting Outliers, fish that are different (feature)
Graphic representation of clusters (suggested)

3.5.4 Evaluation:

The fish clustering can also be evaluated based on labelled sets of fishes, however it can also be a good idea to look here at other evaluation measures. One of the ideas would be to cooperate feedback in the user interface on how useful the fish clustering is. Another idea is to look at measurements for semi-supervised learning. The visualisation of cluster already gives a good indication if these methods work, labelling the correct and incorrect neighbors allows us to evaluate the clustering methods from a human prospective.

3.5.5 Possible Failures:

Clusters are not meaningful: We anticipate that the clustering puts fish of the same species/family together in one cluster. It can however be that the clustering results are not logical from the human perspective. In this case, different clustering techniques or different description have to be used to improve this.
Clusters are too large: The cluster might be too large so that it will be hard to visualise them. It can also have the effect that the same species will have multiple clusters.
Too many outliers: The number of outliers is very large, which does not allow marine biologists to look at the really interesting data because they probably only see the noisy data. Solution can be found by using only description with large certainties so that more noisy data can be ignored.

3.6 Query Engine

3.6.1 Purpose:

The Query Engine allows searching through all the information stored in the storage facilities. The Query Engine is closely related with the RDF/XML/SQL datastore definition, which defines the manner in which they expect that the information will be stored (probably in cooperation with the information provider). After storing the data properly, the query engine should be able to retrieve this information and convert it to a useful format for the user interface and/or users. The challenge of the query interface is to deal with the large amount of data in the storage facilities. Other challenges arise from the fact that not all data is trustworthy. The query engine also has to deal with the fact that certain queries are not possible due to limitations in computer resources.

3.6.2 Input: (Query Answer/Information Requests)

Database information (necessary)
Request for information (necessary)
Meta information (suggested)

3.6.3 Output: (Query information/Representable information)

Query information (necessary)
Representable information (necessary)
Link to other related sites (suggested)

3.6.4 Evaluation:

The query engine should probably be evaluated based on both computation time and usability. The standard questions of the biologist can be used to test the computation time, while feedback of users can help to evaluate the usability. Another interesting idea is to obtain statistical information on the user behaviour and use this to improve and evaluate the both the query answering and the user interface.

3.6.5 Possible Failures:

Generate time consuming queries: The amount of data in the database makes it difficult run complicated queries, because they can require a lot of computational resources. The interface can build in limitations so that users can on make the system unavailable for the rest of the world. Of course, we should also search at solution on the database component side.
Generate queries with too much output records: The database might return millions of records on the query, so user interface should be able to deal with that. Automatic filter operation based on previous queries can help users deal with all the data.
Generate queries with mistakes: The users are usually not aware that the system can also make mistakes, only by showing the image based results will they know that there are outliers. Filtering and annotation of outliers can help to remove the largest number of outliers, still 100% in fish detection/recognition is not expected and should be communicated to the users.

3.7 User Interface Component

3.7.1 Purpose:

The User Interface Component will allow the user to search for information and will then represent this information to the user. The user interface is connected to the query engine, which will retrieve the information for the users. The information provide by the query interface can be linked to other related project, for instance Taiwan fish database, fishbase.org and Catalogue of Life. The first purpose of the website is to provide an interface to the experts and other visitors to search in a relative easy manner through the enormous amount of data. A special area can be developed for specialists (like marine biologists), so that they can login and that their searches will be remembered. Here, they can also ask for specific features they want to add to the website or special request to add extra information in the storage facilities.
Of course, the website is also the portal of the project, making it an interesting medium for communication. In this case, we can think about explaining also the underlining techniques, education in both computer science and biology, setting up a fish label community or processing other underwater cameras.

3.7.2 Input: (User input/Representable information)

User input (necessary)
Representable information from database (necessary)

3.7.3 Output: (Interface/User requests)

Web interface (necessary)
Search functions (necessary)
Request for information (necessary)
Special request interface (suggested)
Visualise fish clusters (feature)
Obtain extra information from related sites (suggested)

3.7.4 Evaluation:

The User Interface can be judged on usability and aesthetics. For both, feedback of users is probably essential. A good evaluation of the interface in the beginning of the project is to evaluate if the interface is able to answer the questions of the marine biologists.

3.7.5 Possible Failures:

Unclear/complicated interface: The user/biologist cannot find certain description, results from the website are unclear to the users. To overcome this problem user feedback is necessary, so next to feedback we already have from our experts, we can add feedback about the website.
Unknown site for marine biologists: The website is not known to marine biologists other than the people participating in the project, website can be found easily by to all people interested in this subject, linking with other web resource helps and correct content on main site is can attract more people.
Broken links with related sites: Links to other website, for instance background information about a certain species does not work any more. Making a cache of the other website can solve this problem. Allowing users (biologists) to maintain the website is also an interesting idea.

3.8 Work-flow Component

3.8.1 Purpose:

The purpose of the Work-flow component is to organize the work that has to be done. Because the different components have different requirements on both CPU, memory and hard-disk consumption, this can be a difficult task. The Work-flow component looks in the database to identify what information can be processed and will execute the appropriate components. The workflow component has to give priorities to certain processes, like the user interface and query engine if they are used. The assignment of the priorities has to be done from a user perspective, this means that he probably always want the user interface to be available, but it is also important to perform the computer vision tasks if the user is not querying information. The Work-flow component can also get a special request from the User Interface and it will create a processing chain to generate the requested information. The Work-flow component however needs detailed information of the other components, like version, purpose, average memory usage, average CPU demands, average I/O demands, average run-time. This information has to be contributed when adding a new component to the system, so the Work-flow component can handle different kinds of information requests and will schedule the correct components for this.

3.8.2 Input: (Special request + Query work)

Component Information (necessary)
Added Videos (necessary)
Added Fishes (necessary)
Special requests from User Interface (suggested)

3.8.3 Output: (Execute work)

Run components (necessary)
Arguments of component (add links to the information that needs to be processed) (necessary)

3.8.4 Evaluation:

The workflow component can be evaluate on the scheduling schemes it produces and the overall effect this has on the amount of work processed. We can compare different schemes by handling the exact same amount of data and compare the schema that performed this task in the most efficient way. For the evaluation, we can also look at the processor utilisation.

3.8.5 Possible Failures:

Fish Detection/Recognition Component fails: In this case, no results or incomplete result are stored in the database. This should however be detected by the workflow component, making sure that the developers can check their code and post a newer version of the component
Not enough resources: There are not enough resources to keep the system working. For instance, it is not possible to run both fish detection and recognition components to process all the videos with fish and still keep the user interface working. In this case, alerts should be generated. These alerts should contain resources that are the bottleneck. In this case, developers can find other solution which require less or other resources. We do not have to be able to process all the data in the beginning with the same rate we acquire it. In the final version, we aim to reach this goal. *A special request requires too much resources:* The amount of time/resources for a certain special request are too much, in this case the user should be informed and alternatives can be offered to the user.
Special requests are not supported by components: The user should be informed that the request is impossible at the moment, while a log can be created so that developers can observe which special requests are important for the users.

3.9 Database Component

3.9.1 Purpose:

The database component allows the different components to store and query information. There will be different kind of information, like videos, images, numerical data and strings. The database will be according to the structure defined by CWI. For both the fish detection and recognition component, there will be a simple interface to query lists of unprocessed videos and fishes. The query interface to these components can be simple, while CWI has a more powerful query interface for the user interface. The database component allows all the other components to share the processed information with each other, but all components must have their unique identifier when they store information. This allows us to be able to see which component is contributing the information.

3.9.2 Input: (Store Information/Query)

Videos (necessary)
Fish Location (necessary)

Fish Tracking (necessary)
Fish Description (necessary)
Fish Recognition (necessary)
Fish Clustering (necessary)
Component which inserted the information (necessary)

3.9.3 Output: (Get Information/Answer Query)

Videos (Frame bases) (necessary)
Fish Location (necessary)
Fish Tracking (necessary)
Fish Description (necessary)
Fish Recognition (necessary)
Fish Clustering (necessary)
Component which inserted the information (necessary)

3.9.4 Evaluation:

The database can be tested by measuring the time it takes to query certain records. This can be performed together with testing the query engine. The queries define by biologists give a good starting point for evaluation of the database. Generating random queries can also give some insight into the system, but that might test the capabilities of the entire system more than the database component. Finally testing the correctness/data lost of the information in the database gives insight into the robustness of the chosen database solution.

3.9.5 Failures:

Time consuming queries: Despite that is one of the challenge of the project, in some cases, there are not enough resources to compute it. In these cases, we can run queries on part of the database. Detection of these queries is important while we want the system to be available for other users. Communication of this problem through the user interface, by giving alternative options can help the user in his awareness.
Storage full: This can be a serious problem, several solution are possible like using more hard discs or using (better) compression methods.
Hard-disc broken: In the case of hard-disc failures, it is necessary it have a backup at all time, using distributed solutions can already provide us with a framework which automatically deals with these kind of failures. *Hardware/Server unavailable:* The machine that contains the database might be unavailable, in this case backup servers are needed, some distributed framework already have solutions in these cases.
Too many read/write operation simultaneously: The framework can not deal with both the queries and the storage request to the database, in this case the work-flow manager should be able to stop/decrease the workload. It can be handy to make a profile of the expected workload and schedule the work done by additional components according to that profile.

4 Component Integration

The partners should make their components as quickly as possible available on the servers of NCHC. The main reason is that all the data is then available on the servers of NCHC. The partners should be able to access the data from the database at NCHC as soon as possible. We are going to make a manual how to access the data for the database and also how to retrieve the videos. This should then be applicable for all components. In the beginning of the project, we are going to use a simple database solution, later on a different database solution can be used, but a similar interface to the database should be provided to store and retrieve information. The partners should make the development environments at the host institute similar to the environment at NCHC, allowing them to put their software without much trouble on the NCHC system. There should be a version management system at NCHC to backup both the database (information) and the components, allowing partners to recover to older versions.

5 Dependencies

The component of each of the partners is a computer program. The components are in most case independent of the other components except for three issues, namely database access, access to videos and information of other components.

Database Access: Each program needs input from one or multiple partners in the project. The first components that needs to be developed is a (temporary) database component. The database component provides access to all the data in the project (including the videos). Besides the database component, a manual should be in place which explains how other components can retrieve and store information in the database. In the beginning of the project, we assume that a simple SQL database and interface to this database will be sufficient for all the partners. Better solutions (distributed databases, triple stores, etc) can be developed during the project, once we have more information to make better decisions on these issues.

Fast method to retrieve video stream/frames: This issue is highly related to the database access. There should be an interface to access to the video recording, where both user/components can immediately access the frame of interest. The video recordings are probably not stored in the database, but they still are use by almost all partners in the project. For both components and marine biologists, we cannot expect that they are going to decode or watch all frames until the scene/fish of interest comes. Returning a frame in the middle of a video can be difficult given certain decoding schemes, so a smart solution have to be developed in order to meet this requirement.

Information about other components: Although every component should be able to access the database, it is necessary to have the same definitions and have clear definitions. In order to start, UNICT and UEDIN make a first database design with data that they expect to create. CWI will be involve in this design and will help to improve this based on the user requirements. Once we have this first database design, CWI will manage/maintain the datastore definitions. This means that adding, changing and removing definitions must be discussed with CWI, because they are the experts and know what is necessary.

5.1 Platform

The NCHC is responsible for the computers and the platform where all the software is running. For this reason, the NCHC can decide which operating system, software packages and other components can run on the system. Because a lot of software depends on the platform, it is important that NCHC provides all partners access to their system as soon as possible (3 months). This allows partners to see the resources that are available, but also allows them to ask for specific libraries, software, databases, etc. The partners are responsible to test their components on the system provided by NCHC. NCHC is responsible to support the partners and to find alternative solutions if the platform is not able to run programs required by the partners.

5.2 Development Environment

Because the component of the Fish4Knowledge project are most still under development, we need to be able to test these components. In order to do this, we suggest to first create a Test Environment. This Test Environment allows us to share information in the early stages of the project, experiment with the data of other users, properly test the components. The Live Environment will be obtain from one of the first stable version of the Test Environment. Finally users can define a Specific Environment for themselves, to test specific idea using the stable components of the other partners.

5.2.1 Test Environment:

The Test Environment enables the partners to test their components. The Test Environment should give a similar interface to the components as the Live Environment. The amount of data in the test environment can be relative small to allows the developers to test their component. The Test Environment should provide a good backup facility, which allows user to return to a previous state, if they made some mistakes. It maybe also a good idea that the partners are able to export this environment to there own institutes to test there.

5.2.2 Live Environment:

The Live Environment will be visible for the users. A component in the live environment should first be tested in the Test Environment. Everything that the component saves in this environment will be saved forever.

5.2.3 Specific Environment:

For some big software changes, like changing the database to a distributed database, a Specific Environment can be created by a partner allowing them to work on the specific issues of their components without bothering other partners. They can also ask other partners to change small part of their components to be compatible with their new environment. If they finish their new component(s) and it works with the stable components of the other partners, they can update

the Test Environment into their Specific Environment.

The advantage of using different environments is that we support both small and big changes during the development stage. We allow the component of the different partners to interact at an early stage, which makes integration at a later stage easy.

5.3 Timetable

We have define the milestones to monitor the project progress of the project and to give us deadlines for finishing work on the project. Notice that some of these milestones are already mention in the Fish4Knowledge project plan (Project Timing), others are more related to the development of the system. The timetable 1 is for now defined in month, but some milestones probably will also get a date attached to them.

Milestones	Months
Access to NCHC computers (NCHC)	3
Simple Database of Fish Detections (UNICT)	4
Test environment at NCHC computers (NCHC)	4
Interface to Simple Database (UNICT,UEDIN,CWI)	4.5
Interface to Video/Frames (NCHC)	4.5
User needs (CWI)	5
First Prototype Fish Detection (UNICT)	5
First Prototype Fish Description/Recognition (UEDIN)	12
Testing various database solutions (CWI)	15
Separate Components Evaluation Plans (All)	17
Final Database design decision (All)	18
First Prototypes of all components (All)	20
Evaluation Plan + Date Evaluation meeting (UEDIN,CWI)	20
Evaluation meeting (UEDIN,CWI)	23
Usability and Improvement assessment (UEDIN,CWI)	24
Prototype database component (possibly distributed) (NCHC,CWI)	27
Final Prototype of all components (All)	30
Evaluation Plan II + Date Evaluation meeting II (UEDIN,CWI)	30
Evaluation meeting II (UEDIN,CWI)	34
Usability and Improvement assessment (UEDIN,CWI)	35

Table 1: Milestones for the development of the Fish4Knowledge system

A Verification (Use cases)

The verification of the component interfaces is done based on the example queries, which can be seen as use cases for the system. We are going to discussed these use cases separately to see which components are involved and how they cooperate. Furthermore, we will discussed the possible bottlenecks of the system and how we think we can deal with these issues in this design.

A.1 What species and numbers of fish appeared in the last N days?

User Interface: Show all the results of the query engine, probably in a diagram (might even give some standard deviations.
Query Engine: Query all fish recorded at certain time/date for these fishes count from which species/family they are. (The bottleneck is the query assuming 10^{10} fishes, distribute databases can help here.)
Database Component: It will store both the fish location (which include time/date) and the fish species/family.
Fish Detection: Given the videos, it will find all the fishes in the videos and return for each fish the location and time.
Fish Recognition: Given the fish, it will determine the species/family of the fish which are detected in the video. (Results of fish recognition not always accurate, this might give some biased results.)
Workflow Component: Run fish detection and recognition components when new videos or fish are added.

A.2 What unrecognised fish were detected? Do they cluster by appearance?

User Interface: Show all the results of the query engine, probably with a ranking on certain fields, like good appearance.
Query Engine: Query all fish with a low certainty score that family/species is correct? Probably want to use also the score how good the appearance of the fish is. Given interesting fish, you might want to find fish with similar vector of description values or descriptions define by marine biologists.
Database Component: It will store for each fish certainty score that family/species is correct, score how good appearance, vector of description values or descriptions defined by marine biologists.
Fish Detection: Given the videos, it will find all the fishes in the videos.
Fish Cluster: Given the fish, that are not in cluster with recognised fish or that are not similar to most recognised fish.
Workflow Component: Let marine biologists make special models/filters that allow to filter out some of the fish species that are not interesting.

A.3 Show me examples of fish from species X?

User Interface: Show all the results of the query engine, probably with a ranking on certain fields, like good appearance.
Query Engine: Query all fish with a certain family or species name. Probably want to use also a certainty score that family/species is correct. (Can query recent appearance of fish first, which limit a search over 10^{10} fishes.)
Fish Detection: Given the videos, it will find all the fishes in the videos and return for each fish the location and time (which can limit the search to recent fish first.)
Fish Recognition: Given the fish, it will determine the species/family and a certainty score that family/species is correct.

A.4 Show me examples of a fish with description X?

User Interface: Show all the results of the query engine, probably with a ranking on certain value of the description, can also suggest also other related descriptions.
Query Engine: Query all fish with a certain vector of description values or descriptions define by marine biologists. Probably want to find a certain value in a range here. (Can query recent appearance of fish first, which limit a search over 10^{10} fishes.)
Fish Detection: Given the videos, it will find all the fishes in the videos and return for each fish the location and time. (which can limit the search on recent fish first.)
Fish Description: Given the fish, it will determine the vector of description values or descriptions define by marine biologists.

A.5 What other species were also present when species X was seen?

User Interface: Show all the results of the query engine, probably in a diagram (might even give some standard deviations) and might be interesting to observe this over time.
Query Engine: Query all fish with a certain family or species name. Select the time /date field and query all fish with similar time/date field and count from which species/family they are. The bottleneck is the double query, assuming 10^{10} fishes and then search them twice or more. The map-reduce platform can be a solution, search for fish in same frames and filter out if species X is present, after that perform a count operation.
Fish Detection: Given the videos, it will find all the fish in the videos and return for each fish the location and time, so fish in the same frame have matches in time/date.
Fish Tracking: Can use the fish behaviour fields to see of different fish interacted with each other.
Fish Recognition: Given the fish, it will determine the species/family of the fish.

A.6 Are the observed numbers of species X increasing in the past 3 years?

User Interface: Show all the results of the query engine, probably in a diagram (might even give some standard deviations).

Query Engine: Query all fish with a certain family or species name, count the fish given the time periods you have defined (month, years).
Fish Detection: Given the videos, it will find all the fish in the videos and return for each fish the location and time. (Result on fish detection not always accurate, this might give some biased results.)
Fish Recognition: Given the fish, it will determine the species/family. (Result on fish recognition not always accurate, this might give some biased results.)

B System Resources

B.1 Introduction

One of the biggest risks in this project is the amount of data with needs to be processed. In this appendix, we want to give an estimation of the resources necessary to process this data. Although, there is not enough accurate information available to give precise number, the following calculation give an idea how to estimate this more accurate in the future. Information on the performance of the component is vital to keep these estimation up to date and will help to make the design decisions necessary for this project.
These numbers are based on the following assumptions on the input of our system:

- 11 video streams (2 high resolution)
- videos record from 6-18 (12 hours of video)
- 10 minutes give 3000 frames
- fish in video: 1 new fish for each 10 frames
- number of frames containing fish: 1 fish is capture in a average of 10 frames

B.2 Computer Vision Components

We have determined the system resources of the computer vision components based on experience in similar projects. We assume that the components memory usage is enough to let other programs run parallel to these components. This means that they cannot to use more than half of the amount of total memory (approximately 2GB) on that machine. This allows the other components e.g. workflow component, database or query engine to run parallel to these process. The estimated time necessary for a single component to run on a single CPU, given the data that they have to process, is given below: fish detection component

- ideal: realtime streaming (single CPU)
- expected: 20 minutes on 10 minute video stream(single CPU)
- worse case: 60 minutes on 10 minute video stream(single CPU)

fish tracking component (probably runs faster if combined with fish detection)

- ideal: during detection
- expected: add 1 minutes on 10 minute video (single CPU)
- worse case: add 2 minutes on 10 minute video (single CPU)

fish description component:

- ideal: 1 sec per fish (single CPU)
- expected: 5 sec per fish (single CPU)
- worse case: 10 sec per fish (single CPU)

fish classification component (probably runs faster if it uses information from fish description):

- ideal: 0.1 sec per fish (single CPU)
- expected: 1 sec per fish (single CPU)
- worse case: 2 sec per fish (single CPU)

fish clustering component (probably runs faster if it uses information from fish description):

- ideal: 0.1 sec per fish (single CPU) might have training time
- expected: 1 sec per fish (single CPU)
- worse case: 2 sec per fish (single CPU)

B.3 Computer Vision Scenarios

B.3.1 Ideal Scenario

In the ideal case, this means that the fish detection is able to analyse a video stream as fast as the camera record them on a single processor. Given that we have 12 hours of video data from 11 videos, this means that will need 11 processes for a half day, or 5.5 process for a day (24 hours). For the fish tracking, we do not need extra time because it uses the results of the fish detection. The total number of fish is: 11(videos) $\times$ 12(hours) $\times$ 6(10 minutes) $\times$ 300(frames) $\times$ 1(fish) $=$ 237600 fish per day. The fish description takes around 1 second for a single processor to compute one fish, this means that in a day we can do 24(hours) $\times$ 60(minutes) $\times$ 60(1 seconds) $=$ 86400 fish descriptions. Fish recognition and clustering usually only takes into account the features calculated by the fish description, which takes less computation time. Both components use around the 0.1 second to compute one fish on single CPU, this means that in a day we process 24(hours) $\times$ 60(minutes) $\times$ 600(0.1 seconds) = 864000 fishes. Given that we have 237600 fish, we divide it with the number of fish that each component can process. This gives us for the fish description (237600 $\div$ 86400) = $\pm$ 2 CPU and for the fish recognition and clustering (237600 $\div$ 864000 =) $\pm \frac{1}{4}$ CPU. The total number of CPU necessary in this scenario are 5 $\frac{1}{2}$ (fish detection/tracking) + 2 (fish description)+ $\frac{1}{2}$ (fish recognition/clustering) = 8 CPUs for each day

B.3.2 Expected Scenario

In the expected scenario, the fish detection takes twice the time, as can be observed in section 7.2. Given that we have 11 video streams, in this case it takes around the 11 CPU to process all the video stream. In order to compute the fish tracking, it costs 1 minute of CPU time for every 10 minutes of video stream. We have 11 video streams $\times$ 12 hours $\times$ 6 video stream of 10 minutes = 792 video stream. Given that it takes 1 minute on single CPU to compute, (792 $\div$ 24 hours) 60 minutes = $\pm$ $\frac{1}{2}$ CPU. The calculate in fish recognition are similar as before, given that find 864000 fishes and we can do:

- Fish description: 24(hours) $\times$ 60(minutes) $\times$ 12(5 seconds) $=$ 17280 on single CPU
- Fish recognition: 24(hours) $\times$ 60(minutes) $\times$ 60(1 seconds) $=$ 86400 on single CPU
- Fish clustering: 24(hours) $\times$ 60(minutes) $\times$ 60(1 seconds) $=$ 86400 on single CPU

That means we need 237600 $\div$ 17280 = $\pm$ 14 CPU for fish description and 237600 $\div$ 86400 = $\pm$ 2 CPU for fish recognition or clustering. The total number of CPU necessary in this scenario are 12 (fish detection) + $\frac{1}{2}$ (fish tracking) + 14 (fish description)+ 4 (fish recognition/clustering) = 30 $\frac{1}{2}$ CPUs for each day

B.3.3 Worse Case Scenario

In the worse case scenario, the fish detection takes six times more than real-time. Given that we have 11 video streams, in this case it takes around the 33 CPU to process all the video stream. For the fish tracking we use a similar calculation, given that we have 792 video streams and it takes 2 minute on single CPU to compute the stream, (792 $\div$ 24 hours) $\div$ 30 (2 minutes) = $\pm$ 1 CPU for a day. The calculate in fish recognition are also similar as before, given that find 864000 fishes and we can do:

- Fish description: 24(hours) $\times$ 60(minutes) $\times$ 6(10 seconds) = 8640 on single CPU
- Fish recognition: 24(hours) $\times$ 60(minutes) $\times$ 30(2 seconds) = 43200 on single CPU
- Fish clustering: 24(hours) $\times$ 60(minutes) $\times$ 30(2 seconds) = 43200 on single CPU

That means we need 237600 $\div$ 8640 = $\pm$ 27 $\frac{1}{2}$ CPU for fish description and 237600 $\div$ 43200 = $\pm$ 5 $\frac{1}{2}$ CPU for fish recognition or clustering. The total number of CPU necessary in this scenario are 33 (fish detection) + 1 (fish tracking) + 27 $\frac{1}{2}$ (fish description)+ 11 (fish recognition/clustering) = 72 $\frac{1}{2}$ CPUs for each day

B.4 Query Engine/Database

The Database and Query interface will probably run distributed on multiple computers, this means that given that we receive a query, the Database and Query Engine get priority above all the other processes. Resolving a query should take up to a second and because the expected group of users is not that large, we expect that it is not a problem to run these programs next to the computer vision programs. On modern computers, it is possible to give priorities or pause processes for small amount of time, which might also be a good idea in these case. The workflow component can play an important role in distributed scheduling of the components. (Input from CWI and NCHC is required here)

B.5 User Interface/Workflow component

Web interface running webserver for specific group of people, so it requires only 1 CPU. Workflow component probably needs also only 1 CPU or less because the computation of the scheduling methods should not take too much processing resources.

