

Architectural design of a digital publishing system

Dimitri Van Landuyt

Johan Grégoire

Sam Michiels

Eddy Truyen

Wouter Joosen

Report CW 465, October 2006

Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

Architectural design of a digital publishing system

Dimitri Van Landuyt

Johan Grégoire

Sam Michiels

Eddy Truyen

Wouter Joosen

Report CW465, October 2006

Department of Computer Science, K.U.Leuven

Abstract

This report describes the architecture of a digital publishing system, which is a next-generation end-to-end media platform using various wired and wireless communication channels for publishing, allowing personalized services based on user-profile and context. This report represents a logical step in the development of the publishing system, and is based as such on an in-depth domain analysis and a study of the major system requirements. The analysis comprises a study of complementary viewpoints on the problem domain, domain models of these viewpoints, and several use cases that reflect functional requirements of four major stakeholders (input sources, advertisers, news desks and media consumers). The objective of this report is to translate both the analysis and the inventory of system requirements into a well-defined architecture.

Architectural design of a digital publishing system

Dimitri Van Landuyt, Johan Grégoire,

Sam Michiels, Eddy Truyen and Wouter Joosen

Research group DistriNet

Department of Computer Science

K.U.Leuven, Belgium

{dimitri.vanlanduyt, johan.gregoire,
sam.michiels, eddy.truyen, wouter.joosen}@cs.kuleuven.be

March 28, 2007

Abstract

This report describes the architecture of a digital publishing system, which is a next-generation end-to-end media platform using various wired and wireless communication channels for publishing, allowing personalized services based on user-profile and context. This report represents a logical step in the development of the publishing system, and is based as such on an in-depth domain analysis and a study of the major system requirements. The analysis comprises a study of complementary viewpoints on the problem domain, domain models of these viewpoints, and several use cases that reflect functional requirements of four major stakeholders (input sources, advertisers, news desks and media consumers). The objective of this report is to translate both the analysis and the inventory of system requirements into a well-defined architecture.

Contents

1	Introduction	6
1.1	Approach	7
2	Viewpoint: Functional	8
2.1	General	8
2.2	Architecture Component Diagram	9
2.3	Internal Components	9
2.3.1	Planning System	9
2.3.2	Corporate News Desk	11
2.3.3	User Management System	12
2.3.4	Service	12
2.3.5	Service News Desk	13
2.3.6	Content Management System	13
2.3.7	Media Advertising System	14
2.3.8	Input Management System	15
2.4	External Components	15
2.4.1	Advertiser	15
2.4.2	Input Source	15
2.4.3	Media Consumer	16
2.4.4	Corporate News Desk Worker	16
2.4.5	Service News Desk Worker	16
2.5	Use Cases Applied on Component Diagram	16
2.5.1	Use cases related to Media Consumers	17
2.5.1.1	Subscribe to a service	17
2.5.1.2	Unsubscribe from a service	17
2.5.1.3	Pull an edition	17
2.5.1.4	Send viewing information	17
2.5.1.5	Send feedback information	18
2.5.2	Use cases related to Input Sources	18
2.5.2.1	Submit input to the publishing system	18
2.5.2.2	Retrieve the number of views of the stories built from their input	18
2.5.3	Use cases related to Advertisers	18
2.5.3.1	Submit a commercial to the publishing system	18
2.5.3.2	Retrieve the number of views of a commercial	19
2.5.4	Use cases related to News Desks (corporate news desks and service news desks)	19
2.5.4.1	Plan or adapt the overall corporate strategy	19
2.5.4.2	Plan and replan a service and their editions	19

2.5.4.3	Plan or replan a service	19
2.5.4.4	Plan or replan an edition	19
2.5.4.5	Verify new content	20
2.5.4.6	Replanning triggered by input	20
2.5.4.7	Add metadata to content or change it	21
2.5.4.8	Build a story from usable content	21
2.5.4.9	Build an edition from stories and commercials for a certain service	22
2.5.4.10	Assist the publishing system in lay-outing an edition	22
2.5.4.11	Push an edition	23
2.5.4.12	Create/Update task assignment	23
2.5.4.13	Other use cases related to News Desk Workers	24
3	Viewpoint: Integration With Additional Services	25
3.1	General	25
3.1.1	Internal additional services	25
3.1.2	External additional services	26
3.1.3	Adding additional Services	26
3.1.4	Service Delivery	29
3.1.5	Content Flows	31
3.1.6	Required and provided interfaces for Service components	32
3.2	Architecture Component Diagram	34
3.3	Internal Components	36
3.3.1	Service Controller	36
3.3.2	Service Management System	36
3.3.3	Service	37
3.3.4	User Management System	37
3.3.5	Planning System	38
3.3.6	Corporate News Desk	38
3.4	External Components	38
3.4.1	(External) Service Provider	38
3.5	Use Cases Applied on Component Diagram	39
3.5.1	Use cases related to Media Consumers	39
3.5.1.1	Subscribe to a service	39
3.5.1.2	Unsubscribe from a service	39
3.5.1.3	Pull an Edition	39
3.5.1.4	Send Viewing Information	40
3.5.1.5	Send feedback information	40
3.5.1.6	Push an Edition	40
3.5.2	Use cases related to external service providers	41

3.5.2.1	Add a new external Service	41
4	Viewpoint: Billing	42
4.1	General	42
4.2	Architecture Component Diagram	43
4.3	Internal Components	43
4.3.1	Billing System	43
4.3.2	Service	45
4.3.3	Service Controller	45
4.3.4	User Management System	46
4.3.5	Media Advertising System	46
4.3.6	Input Management System	46
4.4	External Components	46
4.4.1	External Financial Institution	47
4.4.2	Media Consumer	47
4.4.3	Advertiser	47
4.4.4	Input Source	47
4.5	Use Cases Applied on Component Diagram	47
4.5.1	Use cases related to Media Consumers	48
4.5.1.1	Pull Edition (synchronous)	48
4.5.1.2	Pull Edition (asynchronous)	48
4.5.1.3	Push an Edition (synchronous)	49
4.5.1.4	Push an Edition (asynchronous)	49
4.5.1.5	Create Invoice (asynchronous)	50
4.5.1.6	Notify of Payment	50
4.5.2	Use cases related to Advertisers and Input Sources	51
4.5.2.1	Submit a Commercial (synchronous payment)	51
4.5.2.2	Submit a Commercial (asynchronous payment)	51
4.5.2.3	Create Invoice (asynchronous payment)	52
4.5.2.4	Notify of Payment	52
4.5.3	Use cases related to payment to the Input Source	52
4.5.4	Use cases related to payment from or to the External Service Providers	52
4.5.4.1	Pull Edition (synchronous)	53
4.5.4.2	Pull Edition (asynchronous)	53
4.5.4.3	Create Invoice (asynchronous)	54
4.5.4.4	Notify of Payment	54
4.5.5	Use cases related to payment of the News Desk Workers	54

5	Viewpoint: Context-awareness And Tracking	56
5.1	General	56
5.2	User devices and communication channels	57
5.3	User Tracking	57
5.4	Architecture Component Diagram	58
5.5	Internal Components	58
5.5.1	Service Controller	58
5.5.2	Service	60
5.5.3	User Management System	60
5.6	External Components	60
5.6.1	Media Consumer	60
5.7	Use Cases Applied on Component Diagram	60
5.7.1	Send viewing information	60
5.7.2	Send context information	61
5.7.3	Other use cases affected by contest-awareness	61
5.8	Real-world scenario	61
6	Viewpoints combined	64
7	Layered view	66
7.1	Global layered view	66
7.2	The publishing system as a layer	67
7.3	Rationale for a layered architecture	69
7.4	Illustration of this layered view.	71
8	Deployment view	75
8.1	Quality Attributes	75
8.2	Deployment	76
8.2.1	Conclusion	79
9	Conclusion	79
10	Acknowledgements	80

1 Introduction

With recent breakthroughs in digital paper technologies a need arises for a next-generation digital publishing system to support the new capabilities of those technologies. This technological evolution presents traditional publishers with new challenges, and a gradual shift is expected to occur from a typical news broadcasting¹ situation over narrowcasting² to pointcasting³.

A key novelty is the technological ability to personalise the services offered to the media consumer. One main incentive for traditional news publishers to shift their business from broadcasting towards pointcasting will be the ability to offer its advertisers the ability to perform direct marketing. The reason is that advertisers typically form the main source of the publisher's revenue. Another incentive for this evolution is the fact that the publisher will be able to attract a larger consumer base, by personalising news services to the customer's needs and interests. Especially amongst younger demographic groups the traditional news broadcasting services have been losing interest recently. By shifting towards pointcasting, personalised editions of newspapers can be prepared and sent, based on the personal preferences of the media consumer. Interactivity and personalisation both add to the feeling that the media consumer is in control of the news he wants and not the publisher.

Another novelty is that these evolutions will change the role of the media consumer from a passive reader to an active participant in the news generation process. Indeed, the media consumer can comment on news items or even provide news items themselves. Communities are expected to arise around the services offered by the publisher where the fixed line between journalist and consumer becomes blurry.

This shift to personalised services also allows the publisher to offer additional services, related to the core newspaper service. These additional services aim at providing added value to the overall customer experience. This is opposed to the traditional newspaper, which follows a single service business model. One example of this shift is a moderated forum attached to news items where the media consumers can discuss the various news items. The sky is the limit for additional services, but they are all centered around the core business of the publisher which is selling news.

This document proposes an architecture for such a next-generation digital publishing system. Focus is both on providing an architecture that is backwards-compatible with legacy publishing platforms, and more strongly

¹The same service is offered to each customer.

²The same service offered to a group of customers with the same interests.

³Services are completely personalised and tailored to the customers needs and interests.

on tackling all novel challenges. The architectural design is the result of a close cooperation with an industrial partner named Concentra Media. Frequent meetings have led to better insights in the publishing world and to a better architecture. This document aims at documenting architectural decisions, and providing rationale for each of them. It also represents the evolution that the architecture went through.

1.1 Approach

We have identified several important viewpoints⁴. These viewpoints represent different concerns that were identified in the problem domain. This document presents, if necessary, an architecture component diagram for each of these viewpoints and shows how the architecture supports the various use cases involved in that viewpoint. Viewpoints can be seen as layers of functionality, stacked onto each other. The final result is a full-blown architecture in which design decisions are easy to trace back to the viewpoint or concern that triggered them. Evidently, total compatibility between these layers is guaranteed. This is shown after we've discussed all the viewpoints by providing a global architecture.

This work has been preceded by an in-depth domain analysis of digital publishing [?], and an inventory of the main system requirements for a next-generation digital publishing system [?]. As a validation of the architecture for a certain viewpoint, we have applied the most relevant use cases related to that viewpoint in order to verify that the intended interactions between the system and the actors are supported.

The identified viewpoints are:

1. **Functional:** This viewpoint comprises the basic and core business of the publisher, not taking into account functionalities such as offering extended services, providing context-awareness, As it is the first viewpoint, it offers the foundation for the following viewpoints. The core business of the publisher is publishing, which is the main topic for this viewpoint.
2. **Integration With Additional Services:** In this viewpoint, integration of the core service (news publishing) with additional services is presented. Many ways can be found to integrate a certain service with the core publishing service. Both external service providers and additional internal services are taken into account.

⁴Do not confuse these viewpoints with the term *viewpoint* in several books about software architecture.

3. **Billing:** The billing viewpoint shows how remuneration and charging of media consumers is dealt with within the publishing system, and how the billing concern influences the the system. It also deals with user tracking needed to successfully allow billing.
4. **Context-awareness And Tracking:** Context-awareness forms one of the new opportunities within publishing. The goal of it is to tailor services to the preferences, needs and more generally the context of the media consumer. Tracking deals with how this context of the media consumer is recorded or derived.
5. **Security:** It is clear that in a digital publishing system security is very important. The system needs to be conform with several laws and regulations. One example is formed by the privacy laws in effect, stating that consumer information must be protected. How that affects the architecture is shown in this viewpoint.

For each of these viewpoints, a component-and-connector view is provided onto the system, showing components and dependencies between components. Also, the main use cases are illustrated in the form of textual interaction elaborations, which show the interactions needed between these components to achieve the desired behavior for each use case.

The following viewpoints are worked out in this report: functional (Section 2), integration with additional services (Section 3), billing (Section 4) and context-awareness (Section 5).

Furthermore, a layered view (Section 7) is offered, illustrating how existing middleware platforms could be used to ease the development and to increase reuse.

Finally, a deployment view (Section 8) is provided on the architecture, which maps the identified components onto physical nodes (servers) and shows the connections needed between them. This view illustrates how certain quality attributes such as availability, scalability, etc are achieved.

2 Viewpoint: Functional

2.1 General

The functional viewpoint presents the architectural elements needed to provide the core business of the publisher. This core business is translated into a core service, which is more specifically the service of providing news stories to the media consumer. These news stories are mostly packaged in editions.

Additional issues needed to allow the publisher to perform its core business are also taken into account –user management, news management, input management and advertisement management.

2.2 Architecture Component Diagram

This Section identifies several main internal components, i.e. the Media Advertising System, the Corporate News Desk, the Service News Desk(s), the Planning System, the Input Management System, the Content Management System, the Service component and the User Management System. These will all be explained below.

There are also some external components (previously identified as actors) that interact with the publishing system in various ways. These external components can basically be considered as physical users that may or may not be using some device and/or software to allow interaction with the system. They are not shown explicitly in Figure 1 because this would overload the figure. These external components are the Advertiser, the Input Source, the Media Consumer, the Service News Desk Worker and the Corporate News Desk Worker. The interfaces between these external components and the system are explicitly shown, as they are necessary to explain the interactions between them and the system.

The component diagram also identifies the main interfaces that can be used by components to communicate with each other.

The architecture component diagram for the Functional viewpoint can be found in Figure 1.

2.3 Internal Components

To get a better understanding of each of the internal components, we will now give a brief overview of each of them. We will also give a brief explanation of their role and responsibilities within the architecture.

2.3.1 Planning System

The Planning System component is a centralised repository that keeps track not only of the overall corporate strategy, but also of the strategies associated with different services and the edition planning for each service. It is not only responsible for storing these policies and rules, but also for enforcing them. It is an essential component with an important role in the decision making of the publishing company.

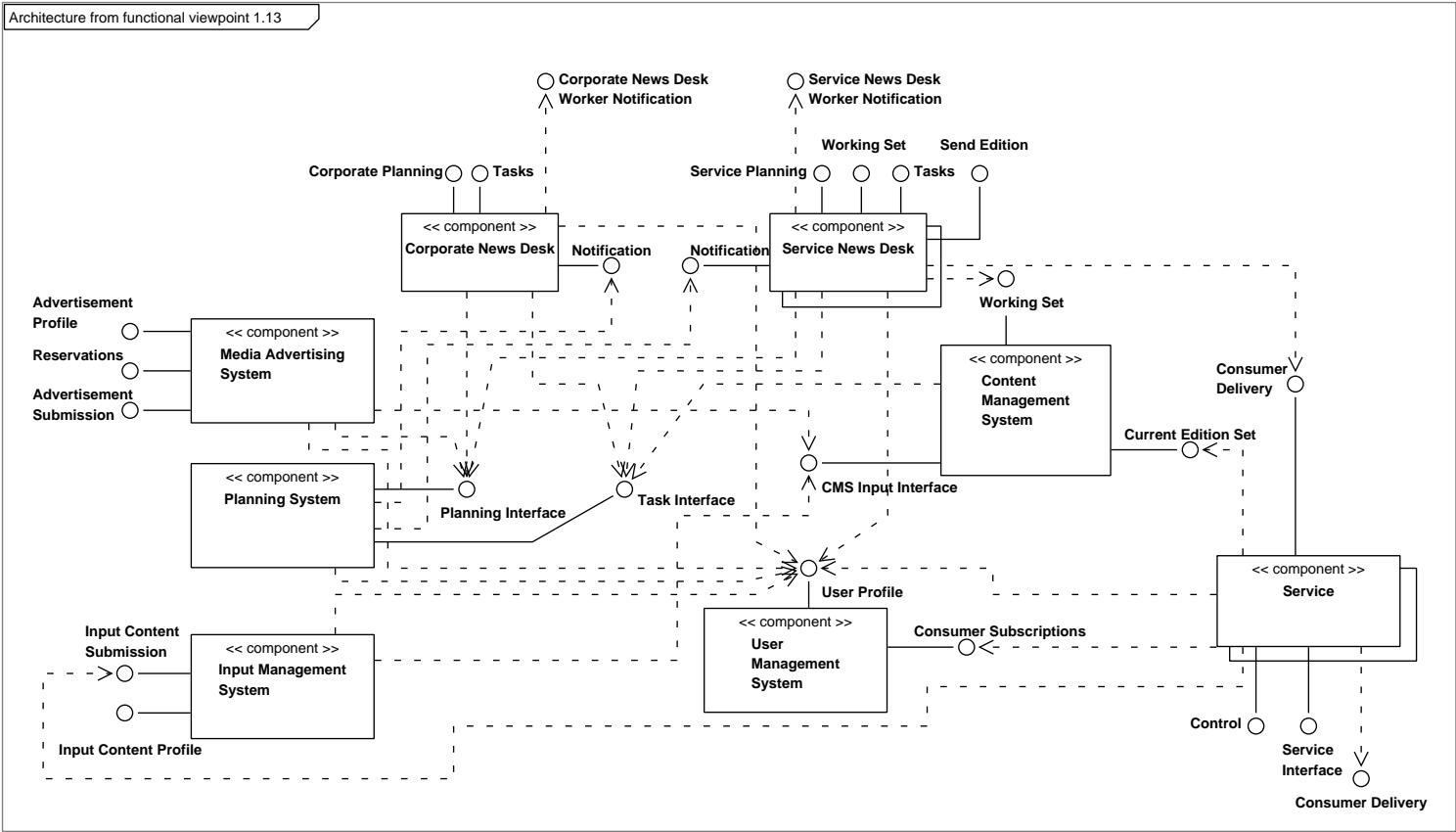


Figure 1: Architecture Component Diagram for the functional viewpoint.
10

Another responsibility of this component is the handling and the assignment of different tasks and delegation of these to the relevant news desk workers (the persons working at a news desk). This delegation must also be conform with the corporate and/or service strategy. The typical workflow of the publisher, which is elaborated more in depth in [?], revolves around this task mechanism.

The interfaces offered by this component are:

- **Task Interface:** This interface is used by the different workers to offer tasks to the Planning System component. These tasks must then be forwarded to the right worker for the job, conform with the planning and the strategy. This forwarding may be done automatically, or in case of doubt, the task will be presented to the Corporate News Desk, which can then further delegate it. This interface is also used by the workers to retrieve an overview of the tasks to be done.
- **Planning Interface:** This interface is used to query and adapt the planning and/or strategies. This can either be service planning, corporate strategy or concrete edition planning.

2.3.2 Corporate News Desk

The Corporate News Desk is responsible for the overall corporate strategy. Also, if new input is received and the Planning System has no information about who it must be presented to, it will be sent towards the Corporate News Desk. The Corporate News Desk can then decide what to do with it and which Service News Desk will have to receive it. The Corporate News Desk also handles the interaction with the Corporate News Desk Workers and thus offers some interfaces to them. In general, it offers the following interfaces:

- **Corporate Planning:** This interface is offered to the Corporate News Desk Worker to consult and/or adapt the planning of the corporate strategy.
- **Tasks:** This interface is used by a Corporate News Desk Worker to insert new tasks into the task system. This is mostly done to delegate certain tasks.
- **Notification:** This interface is used by the Planning System to notify a Corporate News Desk Worker of the addition of new tasks.

2.3.3 User Management System

The User Management System is the component that handles all information about all of the publisher’s users, which are the Media Consumers, the Advertisers and the Input Sources. This information contains amongst others contact information, such as the name and address. This component also keeps track of subscription information for each Media Consumer. This system offers the following interfaces:

- **User Profile:** This interface is used for reading and changing the profile of a user.
- **Consumer Subscriptions:** This interface is used for handling everything that is subscription-related.

2.3.4 Service

The Service component serves as a representation for a service that is offered to the Media Consumer. The component shown here is the core service offered by the publisher. Here, it is a basic newspaper-like edition-based service, that works with a centralised content management system. The Service that is depicted uses a Content Management System as back-end, together with a Service News Desk to edit stories and editions. Additional services can be offered to provide added value to the Media Consumers. How this is done is explained in the viewpoint about “Integration With Additional Services” (Section 3). This component offers the following interfaces:

- **Consumer Delivery:** This interface is used by a Service News Desk to trigger the release of an edition. It is a notification that a certain edition was added to the Current Edition Set. The Service component will fetch the edition and send it to all Media Consumers that are subscribed to the Service.
- **Service Interface:** This interface is offered to the Media Consumer for browsing through editions, lists of editions, It is also used to send viewing information and feedback from the Media Consumer to the Service.
- **Control:** This interface is offered to the Media Consumer for subscribing/unsubscribing to a service and to the system.

2.3.5 Service News Desk

Every Service News Desk is responsible for one or more services and the editions that will be released by these services. This includes the planning of these services and their editions but also the actual creation of the stories and editions. The interfaces offered are:

- **Service Planning:** This interface is offered to the Service News Desk Workers and will be used for the planning of a service and/or their editions.
- **Tasks:** This interface is used by a Service News Desk Worker to insert new tasks into the system.
- **Notification:** This interface is generally used to notify the Service News Desk Workers of the addition of new tasks. It is used by the Planning System.
- **Working Set:** This interface enables the Service News Desk Worker to access and use their Working Set. Each Service News Desk Worker has a Working Set, which contains all the content that he is currently working on. The Service News Desk component will basically just forward each message on this interface to the “Working Set” interface that the Content Management System offers.
- **Send Edition:** Once an edition is finished it might be necessary for the Service News Desk to push it towards the Media Consumer. This interface is used for that occasion.

2.3.6 Content Management System

This may be the most important component in the whole system. All content, such as articles, photographs and advertisements are stored and organized centrally in the Content Management System. The interfaces of this component facilitate the lifecycle of stories:

- **CMS Input Interface:** New content (regular content and/or advertisements) is passed to the Content Management System via this interface.
- **Working Set:** After reception of new content, it will at some point be inserted in the Working Set of a certain Service News Desk Worker. This working set is the collection of all content that this Service News

Desk Worker is currently working on. From then on this interface is used to work with that input in order to create a story out of it or include it in an edition.

- **Current Edition Set:** Once an edition is finished and ready to be published, it will be marked as a current edition of a certain service. The Current Edition Set interface can then be used to retrieve this edition in an easy way.

2.3.7 Media Advertising System

This component is used by the Advertisers to submit new advertisements to the system in order to have them published. The Media Advertising System also tags new advertisements with metadata so that it can be sent to the right Service News Desk later on. This tagging can be done by humans or it can be done automatically. It also keeps the list of outstanding reservations which are to be filled in with a concrete advertisement. It offers the following interfaces:

- **Advertisement Profile:** This interface is used by the Advertiser to retrieve information about the advertisements he has submitted to the system. An example is the number of times a certain advertisement has been seen by Media Consumers.
- **Reservations:** Through this interface the Advertiser can request a list of available reservations, in order to be able to pick one out for the submission of a new advertisement. Also, the actual placement of reservations is handled by this interface. A reservation traditionally contains the following information: the dimensions of the advertisement, the page placement, etc. The term “reservation” is overloaded to support future views of the concept. The second interpretation of the word is a reservation as a container of advertisement publication constraints such as the target group, how many times it can be shown to the Media Consumers, In both cases, the concept of a reservation is important to determine the price for the advertisement and to incorporate it in the Service Planning.
- **Advertisement Submission:** The actual submission of advertisements is done through this interface. It can be done by selecting a previously placed reservation or create a new one –in the case of the new kind of reservations discussed above.

2.3.8 Input Management System

The Input Management System is the component that handles incoming non-commercial content, such as articles or photographs, that is uploaded to the system by the Input Sources. It is responsible for adding metadata to this content with the goal of easy classification later on. This metadata can be added by humans or it can be done automatically. It also enables the Input Sources to check the profile of their content, for example the number of times it has been viewed by users. This component offers the following interfaces:

- **Input Content Submission:** This interface is used by Input Sources to send in their content.
- **Input Content Profile:** This interface is used by Input Sources to check the profile of the content sent in by them. This profile contains amongst other things the number of times someone has viewed their content and the number of times their content has been used in published editions.

2.4 External Components

In what follows an overview is given of each of the external components, with their interfaces. These external components can be seen as physical users, together with the devices and/or software that enable them to interact with the publishing system.

2.4.1 Advertiser

The Advertiser submits advertisements to the publishing system, by selecting a previously placed reservation or making a new one. More concretely, he interacts with the Media Advertising System (see also 2.3.7). He also has the option to check the profile of the advertisements uploaded by him (the number of views, ...). The Advertiser can also place new reservations.

2.4.2 Input Source

The Input Source submits new content to the publishing system. More concretely, he interacts directly with the Input Management System (see also 2.3.8). This content can be photographs, facts, articles or (partial) stories. He can also check the profile of the content already sent in by him (the number of views, ...).

2.4.3 Media Consumer

The Media Consumer is subscribed to one or more services (see also 2.3.4) of the publisher, and can request editions from these services. Usually he uses some kind of consumer device (eg.: e-reader) to do this. Also, the publishing system can push editions to him. He is offered the ability to send feedback to the publisher and possibly much more. This component offers the following interfaces:

- **Consumer Delivery:** This interface is used by the publishing system to perform a push of information to the Media Consumer, mostly editions.

2.4.4 Corporate News Desk Worker

The Corporate News Desk Worker is a person that works at the Corporate News Desk (see also 2.3.2). Abstraction is made of the exact physical location of this person. He is offered the functionalities that were described in 2.3.2. This component offers the following interfaces:

- **Corporate News Desk Worker Notification:** This interface is used by the publishing system –more specifically the Corporate News Desk– to notify the Corporate News Desk Workers about their tasks.

2.4.5 Service News Desk Worker

The Service News Desk Worker is a person that works at a Service News Desk (see also 2.3.5). Abstraction is made of the exact physical location of this person. He is offered the functionalities that were described in 2.3.5. This component offers the following interfaces:

- **Service News Desk Worker Notification:** This interface is used by the publishing system –more specifically the Service News Desk(s)– to notify the Service News Desk Workers about their tasks.

2.5 Use Cases Applied on Component Diagram

This section translates the most relevant use cases to the component architecture diagram. The use cases are classified based on the main stakeholders involved: the Media Consumer, the Input Source, the Advertiser and the News Desks. For a detailed description of the use cases we refer to the system requirements [?].

Each use case is presented in a compact format that illustrates every step as follows: Sending Component \rightarrow Message \rightarrow (Receiving Component):Interface. In other words, every step identifies the component that sends a message, the message itself, the component that receives the message and the interface of this component where the message arrives.

2.5.1 Use cases related to Media Consumers

2.5.1.1 Subscribe to a service

1. Media Consumer \rightarrow subscribe \rightarrow (Service):Control
2. Service \rightarrow submit subscription \rightarrow (User Management System):Consumer Subscriptions

2.5.1.2 Unsubscribe from a service

1. Media Consumer \rightarrow unsubscribe \rightarrow (Service):Control
2. Service \rightarrow submit unsubscription \rightarrow (User Management System):Consumer Subscriptions

2.5.1.3 Pull an edition

1. Media Consumer \rightarrow pull edition \rightarrow (Service):Service Interface
2. Service \rightarrow get edition \rightarrow (Content Management System):Current Edition Set
3. Service \rightarrow send edition \rightarrow (Media Consumer):Consumer Delivery

2.5.1.4 Send viewing information

1. Media Consumer \rightarrow send viewing information \rightarrow (Service):Control
2. Service \rightarrow send viewing information \rightarrow (User Management System):User Profile
3. Service \rightarrow update number of views \rightarrow (Content Management System):Current Edition Set

2.5.1.5 Send feedback information

1. Media Consumer → send feedback information → (Service):Service Interface
2. Service → send feedback information → (Input Management System):Input Content Submission
3. Input Management System → add content → (Content Management System):CMS Input Interface
4. *Execute use case: Verify new Content (2.5.4.5).*

2.5.2 Use cases related to Input Sources

2.5.2.1 Submit input to the publishing system

1. Input Source → submit input → (Input Management System):Input Content Submission
2. Input Management System → add content → (Content Management System):CMS Input Interface
3. *Execute use case: Verify new Content (2.5.4.5).*

2.5.2.2 Retrieve the number of views of the stories built from their input

1. Input Source → retrieve number views → (Input Management System):Input Content Profile
2. Input Management System → retrieve number views → (Content Management System):CMS Input Interface

2.5.3 Use cases related to Advertisers

2.5.3.1 Submit a commercial to the publishing system

1. Advertiser → get reservation list → (Media Advertising System):Reservations
2. Advertiser → submit commercial → (Media Advertising System):Advertisement Submission
3. Media Advertising System → add content → (Content Management System):CMS Input Interface
4. *Execute use case: Verify new Content (2.5.4.5).*

2.5.3.2 Retrieve the number of views of a commercial

1. Advertiser → retrieve number views → (Media Advertising System):Advertisement Profile
2. Media Advertising System → retrieve number views → (Content Management System):CMS Input Interface

2.5.4 Use cases related to News Desks (corporate news desks and service news desks)

2.5.4.1 Plan or adapt the overall corporate strategy

1. Corporate News Desk Worker → plan corporate strategy → (Corporate News Desk):Corporate Planning
2. Corporate News Desk → plan corporate strategy → (Planning System):Planning Interface

2.5.4.2 Plan and replan a service and their editions This use case was originally proposed in the analysis report on digital publishing [?] and the requirements analysis report [?], but it makes more sense to split it in the following two:

- Plan or replan a service (2.5.4.3)
- Plan or replan an edition (2.5.4.4)

2.5.4.3 Plan or replan a service

1. Service News Desk Worker → plan service → (Service News Desk):Service Planning
2. Service News Desk → plan service → (Planning System):Planning Interface

2.5.4.4 Plan or replan an edition

1. Service News Desk Worker → plan edition → (Service News Desk):Service Planning
2. Service News Desk → plan edition → (Planning System):Planning Interface

2.5.4.5 Verify new content This use case puts together two use cases that were specified in the analysis report [?], namely “Verify Input” and “Verify Advertisement”. The reason is that they are identical.

Trigger: New content arrives at the Content Management System

1. Content Management System → notify of input addition → (Planning System):Task Interface
2. Planning System → notify of task addition → (Corporate News Desk):Notification
3. Corporate News Desk → notify of task addition → (Corporate News Desk Worker):Corporate News Desk Worker Notification
4. *Execute use case: Create/Update Task (2.5.4.12) - Corporate News Desk creates a new verification task for a certain Service News Desk.*
5. Service News Desk Worker → add content to working set → (Service News Desk):Working Set
6. Service News Desk → add content to working set → (Content Management System):Working Set
7. Service News Desk Worker → verify content → (Service News Desk):Working Set
8. Service News Desk → verify content → (Content Management System):Working Set

Alternative (If the Planning System automatically knows to which Service News Desk to present the new content):

2. Planning System → notify of task addition → (Service News Desk):Notification
3. Service News Desk → notify of task addition → (Service News Desk Worker):Service News Desk Worker Notification
4. *Execute steps 5 - 8 from original scenario.*

2.5.4.6 Replanning triggered by input

Trigger: New content arrives at the Content Management System

1. Content Management System → notify of input/commercial addition → (Planning System):Task Interface

2. Planning System → notify of task addition → (Corporate News Desk):Notification
3. Corporate News Desk → notify of task addition → (Corporate News Desk Worker):Corporate News Desk Worker Notification
4. Corporate News Desk Worker → replan → (Corporate News Desk):Corporate Planning
5. Corporate News Desk → replan → (Planning System):Planning Interface

Alternative (*If the Planning System knows which Service News Desk should respond to this*):

2. Planning System → notify of task addition → (Service News Desk):Notification
3. Service News Desk → notify of task addition → (Service News Desk Worker):Service News Desk Worker Notification
4. Service News Desk Worker → replan → (Service News Desk):Service Planning
5. Service News Desk → replan → (Planning System):Planning Interface

2.5.4.7 Add metadata to content or change it

1. Service News Desk Worker → add content to working set → (Service News Desk):Working Set
2. Service News Desk → add content to working set → (Content Management System):Working Set
3. Service News Desk Worker → add or change metadata → (Service News Desk):Working Set
4. Service News Desk → add or change metadata → (Content Management System):Working Set

2.5.4.8 Build a story from usable content

1. Service News Desk Worker → create empty story → (Service News Desk):Working Set
2. Service News Desk → create empty story → (Content Management System):Working Set

3. Service News Desk Worker → add content to working set → (Service News Desk):Working Set
4. Service News Desk → add content to working set → (Content Management System):Working Set
5. Service News Desk Worker → edit story with content from working set → (Service News Desk):Working Set
6. Service News Desk → edit story with content from working set → (Content Management System):Working Set

2.5.4.9 Build an edition from stories and commercials for a certain service

1. Service News Desk Worker → create empty edition → (Service News Desk):Working Set
2. Service News Desk → create empty edition → (Content Management System):Working Set
3. Service News Desk Worker → add stories and commercials to working set → (Service News Desk):Working Set
4. Service News Desk → add stories and commercials to working set → (Content Management System):Working Set
5. Service News Desk Worker → fill edition with publications → (Service News Desk):Working Set
6. Service News Desk → fill edition with publications → (Content Management System):Working Set

2.5.4.10 Assist the publishing system in lay-outing an edition

1. Service News Desk Worker → add edition to working set → (Service News Desk):Working Set
2. Service News Desk → add edition to working set → (Content Management System):Working Set
3. Service News Desk Worker → change lay-out → (Service News Desk):Working Set
4. Service News Desk → change lay-out → (Content Management System):Working Set

2.5.4.11 Push an edition

1. Service News Desk Worker → push edition → (Service News Desk):Send Edition
2. Service News Desk → push edition → (Service):Consumer Delivery
3. Service → get edition → (Content Management System):Current Edition Set
4. Service → get subscription info → (User Management System):Consumer Subscriptions
5. Service → send edition → (Media Consumer):Consumer Delivery

2.5.4.12 Create/Update task assignment There are several possible scenarios for this use case. Each scenario follows a different path from the creation of the task to the delivery of the task to the person responsible for executing it.

Scenario 1 (*Corporate News Desk Worker → Planning System → Corporate News Desk Worker*):

1. Corporate News Desk Worker → create or update tasks → (Corporate News Desk):Tasks
2. Corporate News Desk → create or update tasks → (Planning System):Task Interface
3. Planning System → notify of task addition → (Corporate News Desk):Notification
4. Corporate News Desk → notify of task addition → (Corporate News Desk Worker):Corporate News Desk Worker Notification

Scenario 2 (*Corporate News Desk Worker → Planning System → Service News Desk Worker*):

1. Corporate News Desk Worker → create or update tasks → (Corporate News Desk):Tasks
2. Corporate News Desk → create or update tasks → (Planning System):Task Interface
3. Planning System → notify of task addition → (Service News Desk):Notification
4. Service News Desk → notify of task addition → (Service News Desk Worker):Service News Desk Worker Notification

Scenario 3 (*Service News Desk Worker* → *Planning System* → *Service News Desk Worker*):

1. Service News Desk Worker → create or update tasks → (Service News Desk):Tasks
2. Service News Desk → create or update tasks → (Planning System):Task Interface
3. Planning System → notify of task addition → (Service News Desk):Notification
4. Service News Desk → notify of task addition → (Service News Desk Worker):Service News Desk Worker Notification

Scenario 4 (*Service News Desk Worker* → *Planning System* → *Corporate News Desk Worker*):

1. Service News Desk Worker → create or update tasks → (Service News Desk):Tasks
2. Service News Desk → create or update tasks → (Planning System):Task Interface
3. Planning System → notify of task addition → (Corporate News Desk):Notification
4. Corporate News Desk → notify of task addition → (Corporate News Desk Worker):Corporate News Desk Worker Notification

2.5.4.13 Other use cases related to News Desk Workers The following use cases are also provided to News Desk Workers, but they are not shown in detail, mostly because their elaboration is very straightforward, and it would fill too much space.

1. Create new Service News Desk Worker
2. Create new Corporate News Desk Worker
3. Consult News Desk Worker profile
4. Adapt News Desk Worker profile
5. remove Service News Desk Worker
6. remove Corporate News Desk Worker
7. place Service News Desk Worker at Service News Desk

3 Viewpoint: Integration With Additional Services

3.1 General

In this viewpoint, the functional viewpoint that was presented in Section 2 is extended to allow the integration of additional services in the publishing system. The aim of this Section is to explain how multiple services can be offered to the Media Consumer and how they can be integrated together to offer specialized functionalities. This viewpoint focuses mainly on service management.

Every service is represented by a Service component in the system. An example is the Service component that was shown in Figure 1. This service represented the core service of the publisher, which is providing news to the Media Consumer. A distinction is made between internal additional services and external additional services. Furthermore, a distinction can be made between services that extend already existing services, or services that are offered as being stand-alone.

3.1.1 Internal additional services

An internal additional service will make use of the Content Management System, which is the core of the publishing system. An example of such an additional service is an archive service, which allows the Media Consumer to search for editions that were published before. This Service is an internal service, in the way that it integrates in the publisher's architecture (it has access to the CMS), and provides extra functionalities that are close to the core service of the publisher (which is publishing news stories and is represented by the NewsPaper service). Another example of a possible internal additional service is the moderation service that was introduced in the analysis phase [?]. This service adds functionality to the core service, in the way that additional meeting places are linked to different topics, where the Media Consumers can discuss different news items. A Moderation Service component will have to be developed and plugged into the publishing system. It will have access to all internal components, such as the Content Management System. It might also have a separate Service News Desk that will be deployed for it. This would be an example of a service that extends an already existing service. An example of a stand-alone service would be a Calendar Service, offering the Media Consumer an overview of upcoming events.

3.1.2 External additional services

Also, the publisher can choose to offer additional external services, which are provided by a new kind of external component, the external service provider. For an example of a service that extends another service, one can imagine an external restaurant system, providing a database of different restaurants with ratings, that integrates with the general newspaper. Each time a user reads an article about food, an advertisement could be next to the article which presents a nearby restaurant where this food is served. The same service could also be offered as a stand-alone service. In this way the Media Consumer is offered a user interface in which he can formulate a query to find a restaurant of choice. These two services can be considered as two different services, both using the same external back-end, provided by the external service provider. This type of integration of external services into the publishing system will be explained next.

3.1.3 Adding additional Services

In order to allow new Service components such as the Archive Service or a Restaurant Service to be added, it must be well-defined which interfaces a Service component must at least offer and which dependencies a Service component must at least have within the system. To improve pluggability and benefit modifiability, these dependencies should be reduced to the absolute minimum and the addition of a new service must be as easy as possible. This will be an ongoing concern in this Section. The minimal requirements for a Service component will be discussed further on.

A new component that is called the Service Controller is introduced in this viewpoint. It is the single point of access for a service, and it is the missing link between Services and Media Consumers. If a Media Consumer for example needs an edition of a certain Service, this Service Controller will forward the request for a new edition to the correct Service component. The Service Controller is also the only access point into the publishing system from the point of view of the Service. If a Service needs information from the publisher, it will use the Service Controller to get it. This improves pluggability, achieves a better separation of responsibilities and allows better control over the different services.

The general mechanism is illustrated in Figure 2. This Figure shows an example situation of how multiple services –both internal and external– can be integrated with each other to provide the Media Consumer with additional functionalities. More concretely on this Figure, some internal Service components, such as a general Newspaper Service and an Archive Service,

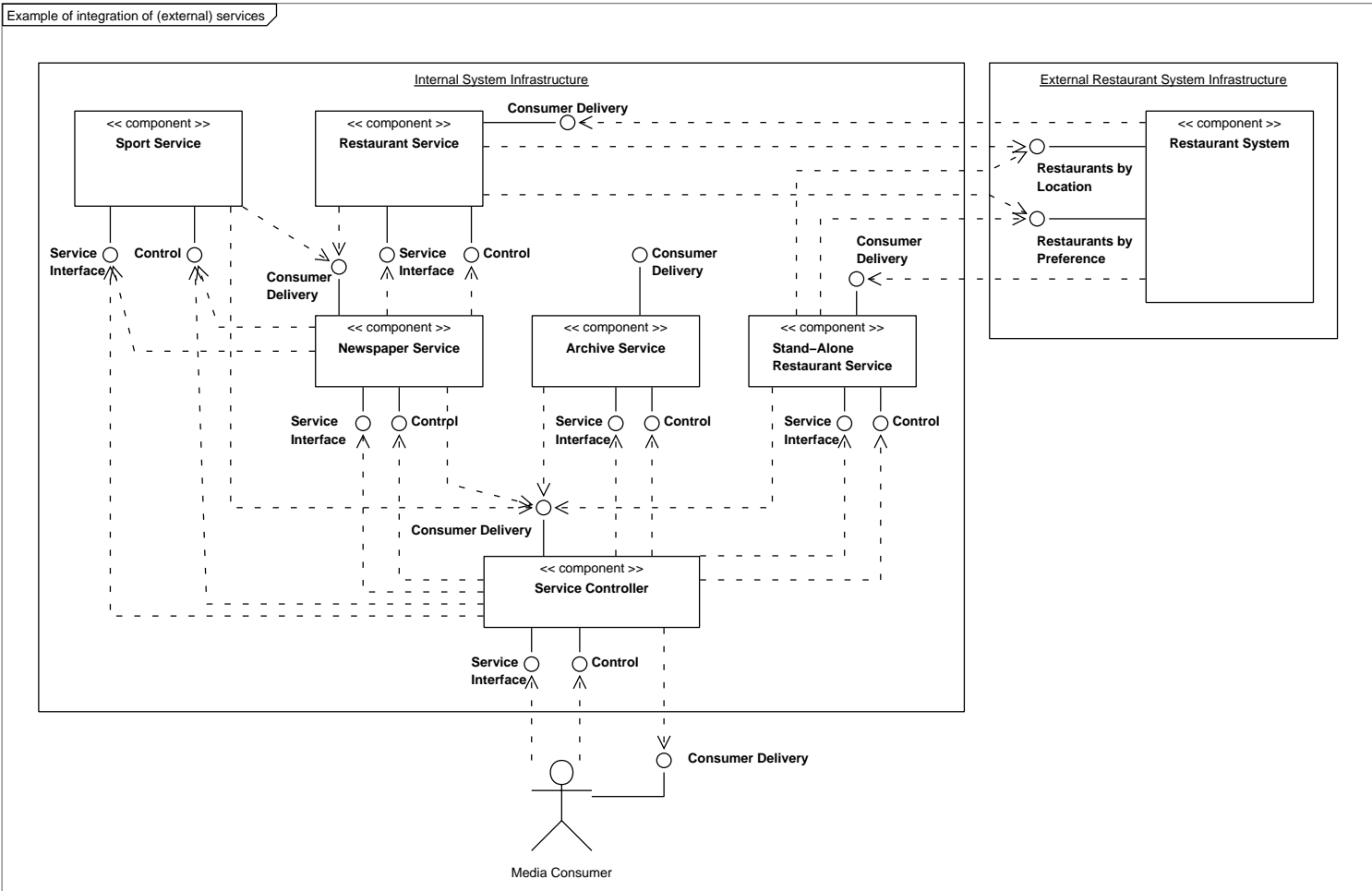


Figure 2: Example situation where many services are offered to the Media Consumer.

are shown, together with an external Restaurant Service, which allows the Media Consumer to search for restaurants. Also, a Sport Service is shown, which generates news stories.

Services that are presented as children of the same component, next to each other, are offered independently from each other. An example in Figure 2 is the relationship between the Newspaper service and the Archive service: the Media Consumer can decide to subscribe to only one of these two and the system will work equally well. If one service is “hooked into” another service, this means that this service offers its functionalities to the other service, and integrates with it. An example in Figure 2 is the relationship between the Newspaper Service and the Restaurant Service. The Restaurant Service is designed to work together with the Newspaper Service and provides additional functionalities to this general newspaper. This covers the example given previously where restaurant advertisements are included in the newspaper, based on content. In this case, the Newspaper Service must of course be designed to accept this type of external input. Of course, this kind of dependency between services is not limited to one level and a dependent service can also have dependent services of its own. The result is a *service dependency tree*, in which all children of a node are dependent of that node, and all neighbours on one level are independent from each other. The root of this tree is the Service Controller. The list of all direct children of the Service Controller is the list of services that are offered directly to the Media Consumers. The Media Consumer is only aware of these services.

In Figure 2, the Stand-Alone Restaurant Service is an independent service that uses the same external back-end as the Restaurant Service, but it is however a differently implemented component. The Restaurant Service will provide functionality to integrate the restaurant information with a newspaper while the Stand-Alone Restaurant Service will for example provide a user interface that enables the Media Consumer to formulate a query and search for certain restaurants.

External Service components will not perform a lot of computation. Each will function as local proxy for an external service, possibly translating requests and forwarding them. In the case of an external service, such as the two Restaurant components, each request will be translated and forwarded to an external back-end, in our case the “External Restaurant System Infrastructure” (think of the Adapter design pattern). If for example the user changes location while reading his newspaper, this can trigger the Restaurant Service component to request a list of restaurants nearby by using the “Restaurants by Location” interface of the Restaurant System. Upon receiving this list it will then integrate these results with the newspaper.

This mechanism makes the installation of new services very easy. A pos-

sible external service provider must negotiate with the Service Management System, which in turn can present offers to the Corporate News Desk, using the task mechanism that was previously explained in Section 2. Once a new service is approved and an agreement is made between the publisher and the external service provider, the external service provider must provide an implementation of the new Service component and send it to the Service Management System. The Service Management System will further handle the deployment of this component into the system and make sure the new service is offered to the Media Consumers.

3.1.4 Service Delivery

Another aspect that is not handled yet is how the multitude of possible user devices and delivery channels is handled by the publishing system. For this purpose a new layer is introduced, consisting of so-called Service Delivery components. These components perform the translation from a general format (an internal content format, like for example an XML-based format) to a device-specific format. For example, a Cellphone Service Delivery component is specialised in transforming this general format into SMS-format. This component is then further responsible for the delivery to the different Media Consumers, for example by sending it to a telecom operator which handles the delivery of SMSes. This also works the other way around: when feedback is received from the Media Consumer by SMS, it is translated into the general content format by the Cellphone Service Delivery component. When this translation is complete it will be sent back to the Service (via the Service Controller) for which the feedback is meant. In fact, the Service Delivery layer makes abstraction of possible interactions needed between the publisher and telecommunication operators. These interactions are part of the service delivery.

For each of the different Delivery Methods, there will be a Service Delivery component. These components have fixed interfaces, which improves pluggability. This Service Delivery mechanism is demonstrated in Figure 3, which is an extension of the example of Figure 2, showing also how delivery to different devices works.

In this Figure, three Service Delivery components are shown: the Paper Delivery component (for printed paper), the Cellphone Delivery component (for delivery through cellphones) and the WebInterface Delivery component (for delivery on the web).

The Paper Delivery component receives an edition for which the formatting has been done into pages and article composition. Also the

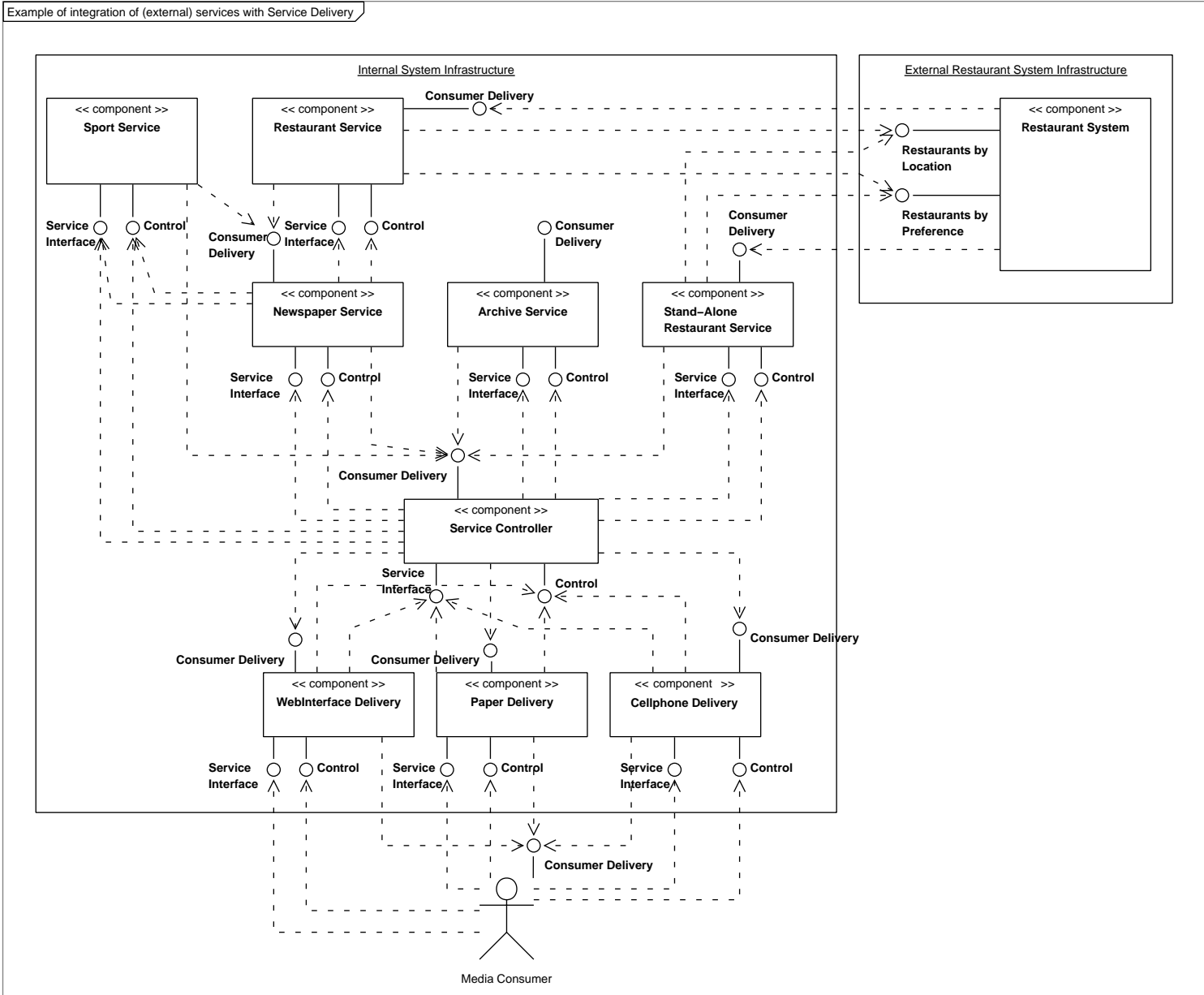


Figure 3: Illustration of Service Delivery components.

layouting is specified. This edition is however still in a generic format. It then translates it into a format that is ready to be printed and handles the delivery of the printed newspaper editions. The other way around, if the Media Consumer sends in feedback (for example by means of a letter or an E-mail), this component translates this feedback into the generic format and forwards it back to the Service Controller, which eventually will present it to the Service for which the feedback was meant. In that way, comments coming through different channels can be thrown together.

The Cellphone Delivery component is responsible for translating a generic edition, which was sent by a Service (for example the Newspaper Service), into an SMS. This is done as follows: the Service News Desk of the Newspaper Service prepares shorter versions of the news stories and composes an edition for cellphones of them. It pushes this edition specifically for release by cellphone. The Service Controller sends this edition to the cellphone Delivery component, which then translates the generic edition into an SMS and sends it to the telecom operator which handles the delivery to the list of addressees.

The WebInterface Delivery component is responsible for converting a generic edition to a web specific edition (probably HTML). After that conversion, this component will then include the edition into the website of the publisher and allow access to it for all subscribed Media Consumers.

3.1.5 Content Flows

At this point it is possible to demonstrate the entire flow of content throughout the system. For generic services, this flow is demonstrated from service until Media Consumer. Everything behind Service can not be called generic: internal services have the entire publishing infrastructure as back-end and are centered around the Content Management System, while external services may have a totally different back-end –highly dependent of the type of the service. The generic flow of content is shown in Figure 4. A generic service delivers editions in a generic format, which are sent to the Service Controller for release on a channel (for a certain delivery method). The Service Controller gets the list of addressees from the User Management System of that particular Service and forwards these editions to the correct Service Delivery component. This component is responsible for the translation to a specific format and delivery to the Media Consumer, which –as said before– consists of the physical user together with devices and software.

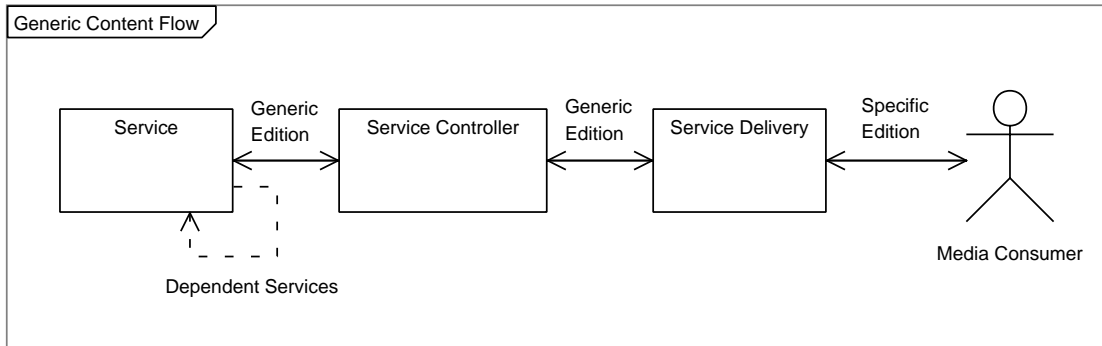


Figure 4: The generic content flow.

It is interesting to demonstrate this flow of content with the examples of Figure 3. This is done in Figure 5. Different Services are shown, such as a Sports Service, a Newspaper Service, an Archive Service and both Restaurant Services. Some of these services use the same back-end. This is for example the case with the Newspaper Service and the Archive Service, because the Archive service must search in all previous editions of the Newspaper service. Others use different back-ends, such as the Restaurant Service. This is an illustration of the diversity that is possible when integrating different services. Also the different Service Delivery components are shown: the Paper Delivery component, the WebInterface Delivery and the Cellphone Delivery.

For the purpose of readability we will not show the Service Delivery components anymore in the rest of this document. It can be considered as being embedded within and part of the Service Controller.

3.1.6 Required and provided interfaces for Service components

In order to allow easy development of Service components, their provided and required interfaces must be made explicit. The provided interface lists the methods it must at least offer to allow the publishing system to interact with it, while the required interface lists the methods it must at least know to provide its functionality. This is shown in Figure 6.

A Service component must offer the “Control Interface”, which represents the communication between the publishing system and the service regarding issues such as user profiles, subscriptions,

If it is expected that a service will have children in the dependency tree,

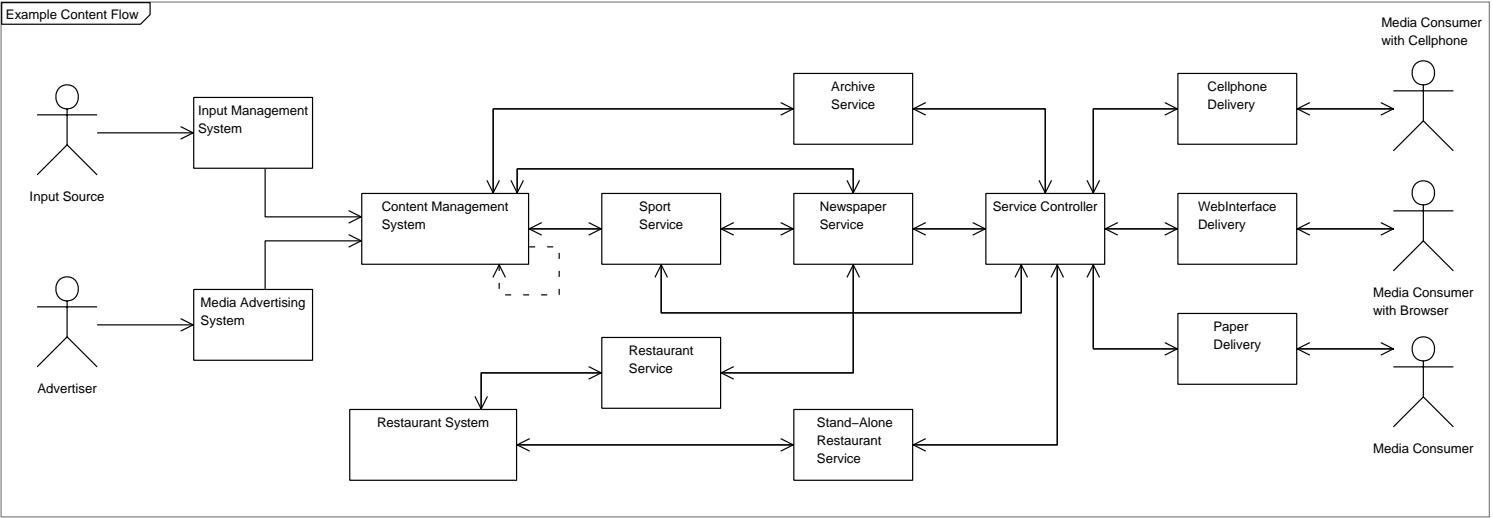


Figure 5: An example content flow.

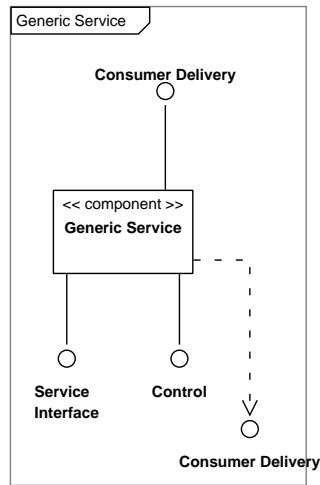


Figure 6: A generic service.

it must offer “Consumer Delivery” to the children in this tree. A leaf service of this tree does not have to offer this interface. For example, the restaurant service will not offer this interface, since it is not expected that this service will receive content from other services.

If the service is not just a broadcast service, but also a point-to-point service, from which editions can be pulled by the Media Consumer then the “Service Interface” must be offered to allow requests for editions by the Media Consumer to be done.

In any case, a Service component must be aware of a “Consumer Delivery” interface of a different component (his parent in the service dependency tree) allowing it to broadcast editions to his parent in the tree (and eventually to the Service Controller which will then be responsible for delivery).

3.2 Architecture Component Diagram

This Section presents an architectural component diagram for the viewpoint of integration with additional services. Again we identify the main components, most of which were already shown in the functional viewpoint.

The general architecture component diagram for this viewpoint can be found in Figure 7. This diagram is an extension of the one given in Section 2.2 and is meant to be completely backwards compatible.

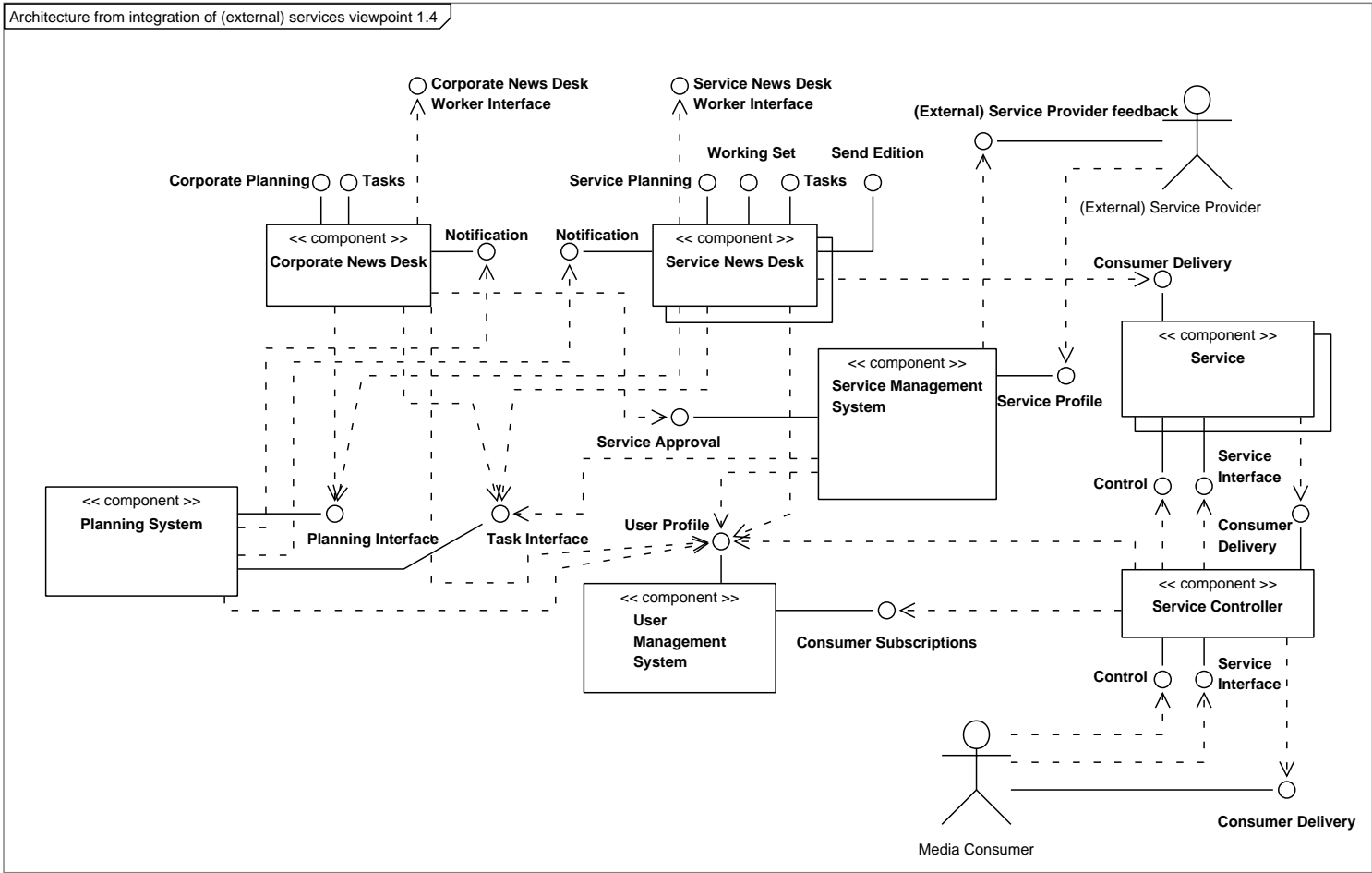


Figure 7: Architecture Component Diagram of the viewpoint of integration with additional services.

3.3 Internal Components

An overview of the relevant internal components with respect to this viewpoint is given. In most cases references back to the appropriate sections in the functional viewpoint for more detailed information are added. We begin with the major two new components: the Service Controller and the Service Management System.

3.3.1 Service Controller

It is possible and even realistic to have multiple services installed that are offered to the Media Consumer. When a request for a certain Service arrives (for example, a “pull an edition”-request), the Service Controller component forwards that request to the correct instance of the Service component. The Service Controller holds a database of services and knows how to contact each service. It is placed between the Media Consumer and the Service components. It offers the following interfaces:

- **Service Interface:** This interface is responsible for forwarding requests to the correct destination Service. For example when a “get edition(Newspaper Service)” request arrives at this interface, it will result in a “get edition”-call to the Newspaper Service component.
- **Control:** This interface provides the functionality for a.o. subscriptions within the publishing system. Subscriptions are thus in the hands of the Service Controller and not in each of the separate services. This improves pluggability and gives the publisher certainty and control about the correctness of the subscriptions. An extra notification about subscriptions is sent to the Service component on which a Media Consumer is subscribing himself (see 3.3.3). This is especially necessary when dealing with a subscription to an external Service.
- **Consumer Delivery:** This interface is used by each of the services to present information to the Media Consumer. The Service Controller can be seen as an intermediate layer which enables the publishing system to keep track of for example statistics for each Service, and is responsible for presenting the content to the right Media Consumer.

3.3.2 Service Management System

This component is responsible for the communication with the different service providers. Service providers are a new kind of stakeholders. They are considered as the persons or companies that want to offer a new Service to the

client base of the publisher by installing a new Service at the publisher's site. The Service Management System is responsible for keeping track of these Service Providers, and enabling them to deploy a new Service component. It offers the following interfaces:

- **Service Profile:** This interface is used to check and change the profile of a service. It is also used for adding new services.
- **Service Approval:** This interface is used by the Corporate News Desk to approve newly requested services.

3.3.3 Service

This component is the actual representation of a service. A large variety in implementations of this component is possible. This component was introduced in 2.3.4 and it offers the interfaces that were introduced there. A service must at least offer the following interfaces:

- **Control:** This interface is now only used for notifying the Service of a new subscription. The actual subscription in the publisher's UMS is already handled by the Service Controller, but extra an notification is sent to the Service component by the Service Controller. This notification could be useful for example when an external service is involved.
- **Service Interface:** This interface is used for fetching editions and lists of editions from this service.
- **Consumer Delivery:** This interface is used by other Services or a Service News Desk to trigger the delivery of an edition to Media Consumers.

3.3.4 User Management System

The User Management System that is presented here is the same User Management System that was introduced in 2.3.3. It offers the same interfaces. Extra functionality is added though, namely the fact that it also keeps information and profiles of the External Service Providers. These are also considered as customers of the publisher, since they will be paying for access to the user base of the publisher. In this profile, additional information is kept such as for example statistics about how much the service is used.

3.3.5 Planning System

This component is the same as the component that was discussed in 2.3.1. It is responsible for knowing the corporate strategy and making semi-automated decisions that are in accordance with this strategy. More concretely with respect to the integration of services, the Planning component also includes the overall strategy concerning external services. It encapsulates for example the types of services that can be accepted and the types of services that will not be approved. For example, if the publisher aims to provide general economic stories only, it will not accept an external service that provides commercials for sport shoes. The Planning System will never automatically approve a new Service. This is due to the high responsibility the decision of approving a new Service implies. The Corporate News Desk will use this information when making a decision about a new Service. The interfaces that were introduced in 2.3.1 are sufficient for this general purpose.

3.3.6 Corporate News Desk

The Corporate News Desk is responsible for entering the Corporate Strategy into the Planning component. Also, the Planning Component can ask the Corporate News Desk to make decisions, based on this strategy. More concretely in this viewpoint, the Planning component can ask for approval and acceptance of a new service. The Corporate News Desk is supposed to base this decision on the overall corporate strategy. This component remains the same component that was introduced in 2.3.2. No new interfaces are needed.

3.4 External Components

One new external component (as mentioned before, an external component comprises of the physical user, together with devices and/or software) is introduced in this viewpoint:

3.4.1 (External) Service Provider

The (external) Service Provider represents the person or company that wants to install a new Service component within the publisher's infrastructure, in order to offer new services. This person has access to the Service Management System (see 3.3.2) and uses the interfaces offered by it.

3.5 Use Cases Applied on Component Diagram

Some of the use cases that were introduced in the functional viewpoint (see 2.2) are extended or overridden. We also introduce some new use cases.

3.5.1 Use cases related to Media Consumers

3.5.1.1 Subscribe to a service

Overrides: Subscribe to a service (2.5.1.1).

1. Media Consumer → subscribe → (Service Controller):Control
2. Service Controller → submit subscription → (User Management System):Consumer Subscriptions
3. Service Controller → submit subscription notification → (Service):Control

In the case of external services, the notification can be forwarded further to the external service provider, depending on the implementation of the service.

3.5.1.2 Unsubscribe from a service

Overrides: Unsubscribe to a service (2.5.1.2).

1. Media Consumer → unsubscribe → (Service Controller):Control
2. Service Controller → submit unsubscription → (User Management System):Consumer Subscriptions
3. Service Controller → submit unsubscription notification → (Service):Control

In the case of external services, the notification can be forwarded further to the external service provider, depending on the implementation of the service.

3.5.1.3 Pull an Edition

Overrides: Pull an Edition (2.5.1.3).

1. Media Consumer → pull edition → (Service Controller):Service Interface
2. Service Controller → pull edition → (Service):Service Interface

3. *This step is service-specific: a new edition is produced or fetched by the service.*
4. Service → send edition → (Service Controller):Consumer Delivery
5. Service Controller → send edition → (Media Consumer):Consumer Delivery

3.5.1.4 Send Viewing Information

Overrides: Send Viewing Information (2.5.1.4).

1. Media Consumer → send viewing information → (Service Controller):Control
2. Service Controller → send viewing information → (User Management System):User Profile
3. Service Controller → send viewing information → (Service):Control
4. *This step is followed by service-specific steps, like for example step 3 of 2.5.1.4.*

3.5.1.5 Send feedback information

Extends: Send feedback information (2.5.1.5).

Step 1 of the original use case is substituted by:

- 1.a. Media Consumer → send feedback information → (Service Controller):Service Interface
- 1.b. Service Controller → send feedback information → (Service):Service Interface

3.5.1.6 Push an Edition

Overrides: Push an Edition (2.5.4.11).

The service receives a trigger from its backend to send a new edition.

1. Service → send edition → (Service Controller):Consumer Delivery
2. Service Controller → get subscription info → (User Management System):Consumer Subscriptions
3. Service Controller → send edition → (Media Consumer):Consumer Delivery

3.5.2 Use cases related to external service providers

This use case was not included in the original analysis document (see [?]), but is introduced to illustrate the mechanism of installing new services into the publisher's infrastructure. It was also added in the requirement analysis phase [?].

3.5.2.1 Add a new external Service

1. (External) Service Provider → make service proposal → (Service Management System):Service Profile
2. Service Management System → add service approval task → (Planning System):Task Interface
3. Planning System → notify of task addition → (Corporate News Desk):Notification
4. Corporate News Desk → approve new service → (Service Management System):Service Approval
5. Service Management System → approve → ((External) Service Provider):(External) Service Provider Feedback
6. (External) Service Provider → upload new service → (Service Management System):Service Profile

Service Management System installs the new Service (and its back-end, if applicable) and updates the User Management System and the Service Controller. It also fixes the link between the new Service and the Service Controller or the parent Service.

4 Viewpoint: Billing

4.1 General

This viewpoint handles the financial aspects of the system. Indeed, the publisher must be paid for publishing an advertisement, and for offering services to the media consumer. At the same time, the publisher must pay its input sources for the content that is used.

In general, two cases with regards to payment can be identified:

- **Synchronous:** In this case, the system has to wait until this payment is actually executed and confirmed by the financial institution. The system requires hard confirmation before the procedure can continue and access can be granted to the resource. In other words, the payment must happen *on-line*, or *synchronously*. An example is a micro-payment where a Media Consumer must call a certain number with his cellphone to perform the payment. After that he gets access to an article. In this case, the cellphone is a payment method and not a delivery channel, as it was introduced in the previous viewpoint.
- **Asynchronous:** In this case the system does not have to wait until the payment is confirmed by the financial institution. The publisher trusts that the user will –in the end– pay his bills. The payment happens *off-line*, or *asynchronously*. An example of this type of payment is a subscription-based payment scheme, where at the end of each month the Media Consumer is presented with a list of editions that were consumed by him and the price he has to pay for it. Evidently, the publisher will in the end still want some kind of confirmation that this payment has succeeded, but it will not wait for it to grant the user access to the resource(s).

All payment methods can be classified as synchronous or asynchronous, even though this distinction may in some cases not be straightforward. An example is a pre-paid cellphone card which represents a certain amount of money. Payment by cellphone is an example of synchronous payment. Indeed, a request for a certain edition can not be fulfilled until the phone credit on this pre-paid card is decreased by the price of the edition. The card is then just a representation of money.

Billing and payments are handled in the system by redirecting payment requests for a certain person or institution to so-called External Financial Institutions. These financial institutions will then contact this person or institution via different, external channels, which are not controlled by the

publisher anymore. For example, when the payment method of choice is payment by cellphone, the External Financial Institution is a telecom operator, and the act of payment is sending an SMS to a certain number, which causes some kind of payment (a decrease of phone credit). The channel is the link between telecom operator and the person for which payment is requested, which is a new logical channel that is out of control for the publisher. The publisher will trust the External Financial Institutions to handle these payments correctly and to send a callback notification about the success or failure of this payment. For example, when the user has actually sent the payment SMS, the publisher will receive a notification about this.

The billing strategy –this encompasses who has to pay when and how– is fully in the hands of each Service. Some services prefer the pay-per-view (synchronous payment) model, others prefer the subscription-based system (asynchronous payment) model.

This explains payment between the Media Consumer and the publisher. Abstraction is made of the relation between the publisher and the external service provider(s). This is handled in agreements between them. In general, the publisher could take the role of *reseller* or *referrer*. This is out of scope of the document.

4.2 Architecture Component Diagram

This Section presents an architectural component diagram for the Billing viewpoint. Again we identify the main components, most of which were already shown in the previous viewpoints.

The architecture component diagram for the Billing viewpoint can be found in Figure 8.

4.3 Internal Components

An overview of the relevant internal components with respect to billing is given. In most cases references back to the appropriate sections in the functional viewpoint for more detailed information are added. We start with the only new component in this viewpoint: the Billing System.

4.3.1 Billing System

The Billing System is the component in the publishing system that keeps information about all payment methods and all External Financial Institutions. It also keeps track of pending payments: these are payments which are requested from the External Financial Institution, but not yet confirmed.

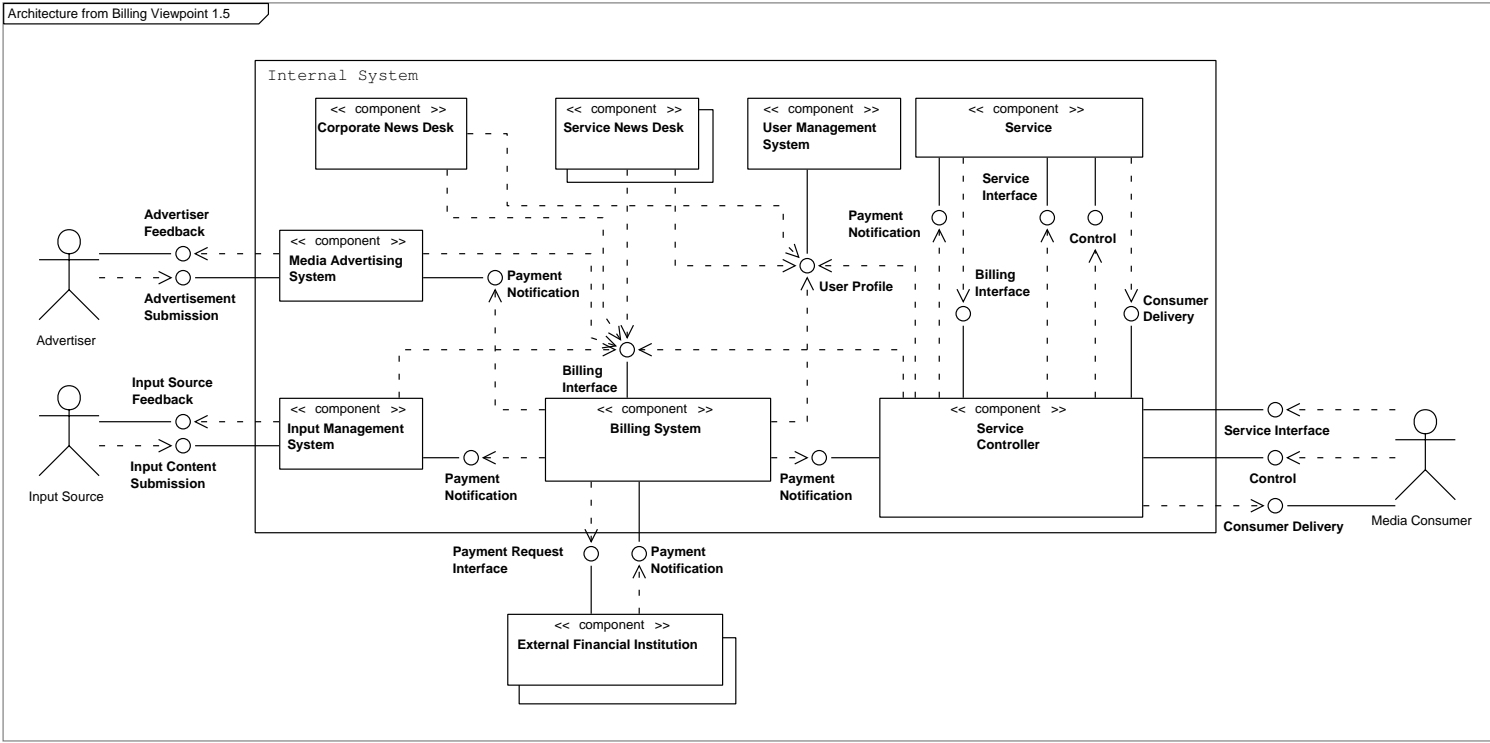


Figure 8: Architecture Component Diagram of the billing viewpoint.

It is also responsible forwarding the payment notifications from the External Financial Institutions to the interested parties. It offers the following interfaces:

- **Billing Interface:** This interface is used to request payments from users. As discussed before, payment can be synchronous or asynchronous. The Billing System internally knows which External Financial Institution to address in order to finish the payment correctly for each type of payment.
- **Payment Notification:** This interface is used by the External Financial Institutions to notify the publisher when a payment transaction has been completed successfully.

4.3.2 Service

This component represents a Service that is offered to a Media Consumer and to which he can subscribe, just as it was already defined before. There is one new interface (other than those introduced in 2.3.4 and 3.3.3):

- **Payment Notification:** This interface is used to notify the Service component of the completion of a certain payment.

4.3.3 Service Controller

This component is the same component that was introduced in 3.3.1. It is generally responsible for having control over services. More in particular with respect to billing, its responsibility is making sure that each transfer of information from a Service to a Media Consumer is recorded within the publishing system, because this allows the publisher to bill the service provider, based on usage statistics. It offers new interfaces:

- **Payment Notification:** This interface is used by the Billing System to notify the Service Controller about the completion of a certain payment for a certain Service. The Service Controller can then further propagate this notification to the correct Service.
- **Billing Interface:** This interface is used by the Service components to invoke the publishing system to request a payment from a certain user.

4.3.4 User Management System

The User Management System is the same as in 2.3.3 and 3.3.4. It contains the User Profiles of all users, which are the Advertisers, the Input Sources, the Media Consumers and the External Service Providers. These profiles consist amongst others of:

- **Financial Information:** For each user, information is kept on how to bill him. For example bank account numbers will be stored here, or cellphone numbers.
- **Reading overview:** In the asynchronous payment case, a general list of transactions is kept here to allow the publisher to keep an overview of payments that are outstanding, and create an invoice based on this list. This happens asynchronously, when some constraint is met. An example is the invoice that is generated after each month.

No new interfaces are needed and mostly “User Profile” is used to fetch information.

4.3.5 Media Advertising System

The Media Advertising System is the same as the one introduced in Section 2.3.7. It will invoke the billing mechanism when Advertisers are placing reservations or uploading advertisements. It offers one new interface:

- **Payment Notification:** This interface is used by the Billing System to notify the Media Advertising System that the payment has been completed.

4.3.6 Input Management System

The Input Management System is the same as the one introduced in Section 2.3.8. It will invoke the billing mechanism when Input Sources are uploading new content. It offers one new interface:

- **Payment Notification:** This interface is used by the Billing System to notify the Input Management System that the payment has been completed.

4.4 External Components

An overview of each of the relevant external components involved in the billing viewpoint is given below. We begin with the only new external component: the External Financial Institution.

4.4.1 External Financial Institution

This is a new external component, which offers the publishing system the functionality to request payments from a certain person or to perform payments to a certain person. Examples are banks or telecom operators. It offers the following interface:

- **Payment Request Interface:** This interface is used to request a certain payment from a certain person or institution to a certain person or institution.

4.4.2 Media Consumer

There are no real differences for the Media Consumer (described before in 2.4.3) with regards to billing. All the interactions between the publishing system and the Media Consumer needed to accomplish billing are done indirectly via the External Financial Institution (see also 4.4.1). The publishing system is only responsible for sending the payment requests to these External Financial Institutions and these will in turn contact the user via some channel (eg.: cellphone). It does not provide new interfaces.

4.4.3 Advertiser

There are no real differences for the Advertiser (described before in 2.4.1) with regards to billing. All the interactions needed to accomplish billing between the publishing system and the Advertiser are done indirectly via the External Financial Institution (see also 4.4.1). It does not provide any new interfaces.

4.4.4 Input Source

There are no real differences for the Input Source (described before in 2.4.2) with regards to billing. All the interactions needed to accomplish billing between the publishing system and the Input Source are done indirectly via the External Financial Institution (see also 4.4.1). It does not provide any new interfaces.

4.5 Use Cases Applied on Component Diagram

In this Section, several use cases that were introduced in Section 2.5 and extended in Section 3.5.1 are extended further to introduce billing.

4.5.1 Use cases related to Media Consumers

These use cases are for pulling an edition (which can be an article, a newspaper or anything else). As explained before, the distinction between synchronous and asynchronous can be made with regards to billing. Some use cases will be an elaboration of the synchronous case, some will explain the asynchronous case.

These use cases must be considered as extensions of the original use case in 2.5.1.3.

4.5.1.1 Pull Edition (synchronous) This is sometimes also called pay-per-view or pay-per-edition.

Extends: Pull Edition (3.5.1.3), step 3 is replaced by the following steps.

- 3.a. Service → request synchronous payment → (Service Controller):Billing Interface
- 3.b. Service Controller → request synchronous payment → (Billing System):Billing Interface
- 3.c. Billing System → get financial information → (User Management System):User Profile
- 3.d. Billing System → request payment → (External Financial Institution):Payment Request Interface
- 3.e *Execute use case: Notify of Payment (4.5.1.6).*
- 3.f Service → send edition → (Service Controller):Consumer Delivery

4.5.1.2 Pull Edition (asynchronous) An example of the asynchronous case is typically when the Media Consumer has a subscription for a newspaper and pays for it at the end of each month.

Extends: Pull Edition (3.5.1.3), step 3 is replaced by the following steps.

- 3.a Service → request asynchronous payment → (Service Controller):Billing Interface
- 3.b Service Controller → request asynchronous payment → (Billing System):Billing Interface

3.c Billing System → update user profile → (User Management System):User Profile

The list of outstanding payments in the user profile is updated.

3.d Service → send edition → (Service Controller):Consumer Delivery

4.5.1.3 Push an Edition (synchronous)

Extends: Push an Edition (3.5.1.6), step 3 is replaced by the following steps:

3.a. Service → request synchronous payment → (Service Controller):Billing Interface

3.b. Service Controller → request synchronous payment → (Billing System):Billing Interface

3.c. Billing System → get financial information → (User Management System):User Profile

3.d. Billing System → request payment → (External Financial Institution):Payment Request Interface

3.e *Execution of use case: Notify of Payment (4.5.1.6).*

3.f. Service → send edition → (Service Controller):Consumer Delivery

4.5.1.4 Push an Edition (asynchronous)

Extends: Push an Edition (3.5.1.6), step 3 is replaced by the following steps:

3.a Service → request asynchronous payment → (Service Controller):Billing Interface

3.b Service Controller → request asynchronous payment → (Billing System):Billing Interface

3.c Billing System → update user profile → (User Management System):User Profile

The list of outstanding payments in the user profile is updated.

3.d Service → send edition → (Service Controller):Consumer Delivery

4.5.1.5 Create Invoice (asynchronous) This use case is related to both asynchronous payment use cases (4.5.1.4 and 4.5.1.2). It is used for generating an invoice for the Media Consumer, where payment is requested for all consumed editions. This use case is triggered by the Service for which the invoice is created, when some constraint is met. An example is an invoice generated at the end of the month.

Precondition: In both asynchronous use cases (4.5.1.4 and 4.5.1.2), a list of outstanding payments was stored in the user profile.

Trigger: The Service triggers this use case when some constraint is met (for example after the end of a certain period or when a certain predefined amount of editions has been reached).

1. Service → commit asynchronous payments → (Service Controller):Billing Interface
2. Service Controller → commit asynchronous payments → (Billing System):Billing Interface
3. Billing System → get outstanding payments list → (User Management System):User Profile
4. Billing System → get financial information → (User Management System):User Profile
5. Billing System → request payment → (External Financial Institution):Payment Request Interface

4.5.1.6 Notify of Payment

Precondition: The Service component is waiting for notification of this payment.

1. External Financial Institution → send payment notification → (Billing System):Payment Notification
2. Billing System → send payment notification → (Service Controller):Payment Notification
3. Service Controller → send payment notification → (Service):Payment Notification

4.5.2 Use cases related to Advertisers and Input Sources

Again, the distinction between synchronous and asynchronous billing can be made, but it can be expected that this will mostly happen asynchronously.

4.5.2.1 Submit a Commercial (synchronous payment)

Extends: Submit a Commercial (2.5.3.1). Step 2 is explained in more detail.

- 2.a. Advertiser → submit commercial → (Media Advertising System):Advertisement Submission
- 2.b. Media Advertising System → request synchronous payment → (Billing System):Billing Interface
- 2.c. Billing System → get financial information → (User Management System):User Profile
- 2.d. Billing System → request synchronous payment → (External Financial Institution):Payment Request Interface
- 2.e. *Execute use case: Notify of Payment (4.5.2.4).*

After notification, the newly uploaded advertisement is accepted.

4.5.2.2 Submit a Commercial (asynchronous payment)

Extends: Submit a Commercial (2.5.3.1). Step 2 is explained in more detail.

- 2.a. Advertiser → submit commercial → (Media Advertising System):Advertisement Submission
- 2.b. Media Advertising System → request asynchronous payment → (Billing System):Billing Interface
- 2.c. Billing System → update user profile → (User Management System):User Profile
The list of outstanding payments in the user profile is updated.
- 2.f. *The newly uploaded advertisement is accepted.*

4.5.2.3 Create Invoice (asynchronous payment) This use case is triggered by an internal event, when some constraint is met.

Precondition: In the asynchronous use case (4.5.2.2), a list of outstanding payments was stored in the user profile.

Trigger: The system automatically generates the invoice at the end of the fixed period of time or when the bill exceeds a threshold amount.

1. Media Advertising System → commit asynchronous payments → (Billing System):Billing Interface
2. Billing System → get outstanding payments list → (User Management System):User Profile
3. Billing System → get financial information → (User Management System):User Profile
4. Billing System → request payment → (External Financial Institution):Payment Request Interface

4.5.2.4 Notify of Payment

Precondition: The Media Advertisement System is waiting for the notification of this payment.

1. External Financial Institution → send payment notification → (Billing System):Payment Notification
2. Billing System → send payment notification → (Media Advertising System):Payment Feedback

4.5.3 Use cases related to payment to the Input Source

Since the payment of Input Sources happens exactly analogous to payment of Advertisers, the payment of Input Sources will not be explained in further detail. The only difference is that in general Input Sources get paid by the publisher, while Advertisers have to pay the publisher.

4.5.4 Use cases related to payment from or to the External Service Providers

External Service Providers are the people or organizations behind additional Services. In some cases, the publisher must pay them because they add value

to the core Service of the publisher. An example is a news corporation who sends in news items, which get integrated with the newspaper service. In other cases, the External Service Provider must pay the publisher, because they get access to the (possibly large) user base of the publisher. An example is an external Advertising service. In general, the payment can happen synchronously or asynchronously. In the synchronous case, a Media Consumer can only get an edition from an external service if this external service has received or performed a payment for it. In the asynchronous case, a list of all editions that were sent to the Media Consumers is kept within the Service Controller, who in turn will trigger the payment after a certain amount of time or when a certain condition is met.

4.5.4.1 Pull Edition (synchronous)

Extends: Pull Edition (4.5.1.1, between step 3.e. and 3.f) or Pull Edition (4.5.1.2, between step 3.c. and 3.d.). Between these steps the following steps are inserted.

- 3.e.1. or 3.c.1. Service Controller → request synchronous payment → (Billing System):Billing Interface
- 2.e.2. or 3.c.2. Billing System → get financial information → (User Management System):User Profile
- 3.e.3. or 3.c.3. Billing System → request payment → (External Financial Institution):Payment Request Interface

The External Financial Institution contacts the External Service Provider to perform the micropayment. The External Financial Institution will send a notification when the payment is done successfully.

- 3.e.4. or 3.c.4 *Execute use case: Notify of Payment (4.5.4.4).*

4.5.4.2 Pull Edition (asynchronous)

Extends: Pull Edition (4.5.1.1, between step 3.e. and 3.f.) or Pull Edition (4.5.1.2, between step 3.c. and 3.d.). Between these steps the following steps are inserted.

- 2.e.1. or 3.c.1. Service Controller → request asynchronous payment → (Billing System):Billing Interface
- 2.e.2. or 3.c.2. Billing System → update user profile → (User Management System):User Profile

The list of outstanding payments in the user profile is updated.

4.5.4.3 Create Invoice (asynchronous) This use case is closely related to the previous use case. It is for generating an invoice for the External Service Provider, where payment is requested for all consumed editions. This use case is triggered by the Service Controller.

Precondition: In the asynchronous use case (4.5.4.2), a list of outstanding payments was stored in the profile of the External Service Provider.

Trigger: The Service Controller triggers this use case (for example after the end of a period or when a certain condition is met).

1. Service Controller → commit asynchronous payments → (Billing System):Billing Interface
2. Billing System → get outstanding payments list → (User Management System):User Profile
3. Billing System → get financial information → (User Management System):User Profile
4. Billing System → request payment → (External Financial Institution):Payment Request Interface

4.5.4.4 Notify of Payment

Precondition: The Service Controller is waiting for the notification of this payment.

1. External Financial Institution → send payment notification → (Billing System):Payment Notification
2. Billing System → send payment notification → (Service Controller):Payment Notification

Note that also the use cases for pushing an edition will be different, but this will be omitted because the mechanism is mainly analogous to the use cases in 4.5.1.3 and 4.5.1.4.

4.5.5 Use cases related to payment of the News Desk Workers

The mechanism that was described above can be reused to handle payment of the News Desk Workers. Since payment of the News Desk Workers will always be asynchronous, there is no need for a Payment Notification to the

News Desks. The Billing System can asynchronously handle these payments and the corresponding News Desk does not need a callback. The elaboration of these use cases is kept out for reasons of readability.

5 Viewpoint: Context-awareness And Tracking

5.1 General

Context-awareness allows the adaptation of newspaper services to the specific context in which the Media Consumer currently resides. This adaptation will mostly be expressed in the way that the content, sent to the Media Consumer, is adapted to his current situation and preferences. More concretely, context-awareness can be based on a number of factors:

1. **Localization:** Context-awareness can be achieved by basing the content, sent to the Media Consumer, on his geographical location. An example is a car advertisement appearing to people who are riding the bus, or a user receiving the regional news of the region where he is actually is residing at the moment.
2. **Preferences and user profile:** Context-awareness can be achieved by basing the content, sent to the Media Consumer, on their personal preferences. An example is a fish restaurant advertisement appearing to someone who likes fish meals. All information that is kept in the user profile can be used. Examples are the device type of the Media Consumer or background knowledge about him. In this way, different editions can be prepared and sent to for example a married Media Consumer and a single Media Consumer.
3. **Time:** Another way to achieve context-awareness is by taking into account time. If for example market research shows that Media Consumers prefer to read short and informative articles in the morning and to read the longer versions in the afternoon, this time factor can be taken into account. Another example scenario is the Media Consumer who has entered his calendar. Suppose now that the publisher can infer that the Media Consumer has not much work to do during the next five minutes. It may respond to this by sending a short edition with short versions of the main news stories, which is supposed to be readable during these five minutes.

Closely related to this is the concept of *user tracking*. To achieve context-awareness, the system must keep track of the user's location, preferences, time information and put all of this info in the user's profile. The more up-to-date this information is, the more effective these context-awareness mechanisms can be.

5.2 User devices and communication channels

The possibility of keeping track of the user's preferences and context is of course highly dependent on the technology and delivery channels that connects the Media Consumer to the system.

Some channels only have *broadcast* capabilities –like the traditional paper-printed newspaper. In such cases it is only possible to get static preferences at the moment of subscription, or none at all. This would only allow for very limited personalisation, because it would incur too many costs to generate a personalised newspaper for each Media Consumer and deliver it. Also, because of the static nature of these profiles, the context-awareness that would be achieved would not be very up-to-date and thus not very effective.

On the other hand there is the *pointcast* case. This case occurs when there is a technological ability of sending information back to the publisher. This allows direct interaction between the publisher and the media consumer, and provides the publisher with a way of offering dynamical and personalised services. This personalisation can be based on static preferences or dynamically detected preferences. For example, in the case of cellphone delivery, it is more or less technically possible to determine the Media Consumer's localization automatically. Personalised editions can then be sent to this Media Consumer, based on the Media Consumer's preferences.

5.3 User Tracking

It is thus important to show how this information will be acquired by the publishing system, which is the process called *user tracking*. This shows how preferences and localization information are acquired by the publisher and how they are stored into the system.

Not all Services are interested in the same information. For example, a Restaurant Service may only be interested in the gastronomic preferences of the user and his location. All the other information is irrelevant for this Service and should not be sent to him, in order to preserve the Media Consumer's privacy. The publishing system must support this and it should be well-defined which type of information can be sent to the external services and which type of information should not be sent. This has of course a lot of legal implications.

Another aspect to show is how the publishing system will use this information to create personalised editions and send these to the Media Consumers. A difficult issue here is to show how this information will be propagated to different interested and possible external services. The more interesting case here is of course the pointcast case as it shows a lot of new possibilities for

the publisher.

User tracking is done in the following way. In the cases where the Media Consumer’s device and Delivery Channel support it, use cases such as “send context information” and “send viewing information” are executed. Basically, the User Management System of the publisher stores all information about the Media Consumer. This information can be the information that comes directly from the Media Consumer, or it can be information that is inferred automatically from the information received by advanced techniques such as data mining methods. For example, when a Media Consumer’s behavior has changed in the last week, in the way that he has read the entire “Travel”-section of the newspaper, the publishing system may come to the assumption that he is currently looking for a holiday destination. This can then be recorded in his user profile.

The User Management System sends this new information to the Service Controller, which contains a small database of information about Services and what they are interested in. This component is in turn responsible for sending the information on a need-to-know basis to each Service. For example, the Restaurant Service will be registered in the Service Controller as a service that is interested in gastronomic preferences and localities. The Service Controller will then forward all information about gastronomic preferences or locality to the Restaurant Service.

5.4 Architecture Component Diagram

This Section presents an architectural component diagram for the Context-awareness viewpoint. Most of the components that were introduced in the previous viewpoints are re-used –albeit sometimes with extra functionality.

The architecture component diagram for the Context-awareness viewpoint can be found in Figure 9.

5.5 Internal Components

5.5.1 Service Controller

The Service Controller shown in Figure 9 is the same component that was introduced in the integration viewpoint and further extended in the billing viewpoint. One new interface is added though:

- **Context Information:** This interface is offered to the Services to request context information of a certain user. This allows the Service to make personalised editions. The Service Controller has the responsi-

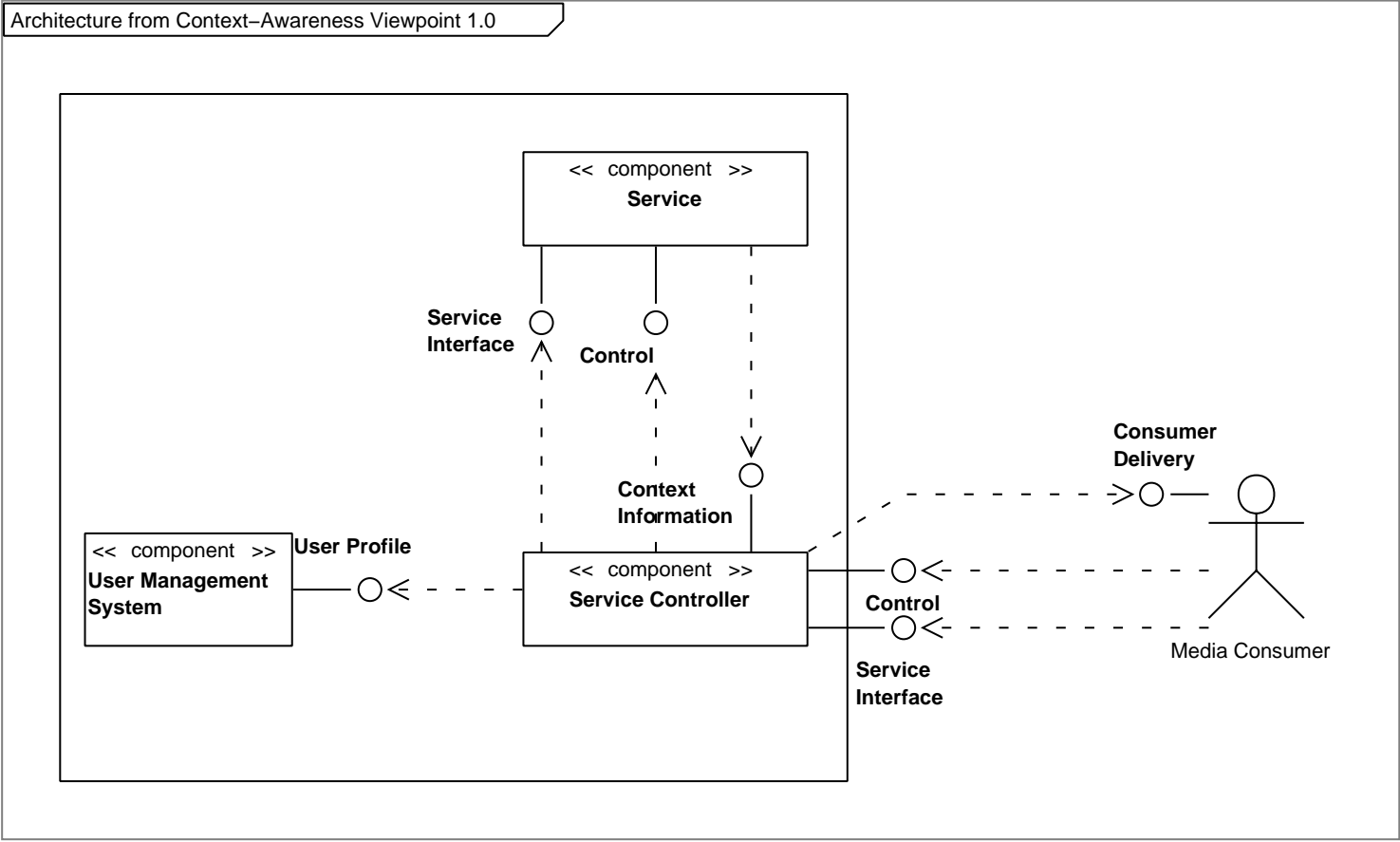


Figure 9: Architecture Component Diagram of the context-awareness viewpoint.

bility to get the relevant information from the user profile and to make sure that the privacy of the Media Consumer is never violated.

5.5.2 Service

The Service shown in Figure 9 is the same component that was introduced in the functional viewpoint, further extended in the integration viewpoint and in the billing viewpoint. No new interfaces are offered. The responsibility of the Service in light of context-awareness is to make personalised editions, tailored to the preferences and needs of the Media Consumer.

5.5.3 User Management System

The User Management System shown in Figure 9 is the same component that was introduced in the functional viewpoint, further extended in the integration viewpoint and in the billing viewpoint. No new interfaces are offered. The responsibility of the User Management System in light of context-awareness is to keep track of the personal preferences and context information of the Media Consumer, and keep these in his profile.

5.6 External Components

5.6.1 Media Consumer

The Media Consumer shown in Figure 9 is the same component that was introduced in the functional viewpoint, further extended in the integration viewpoint and in the billing viewpoint. No new interfaces are offered. The Media Consumer communicates his context and his preferences (directly or indirectly) to the publishing system. This information is offered to interested Services and enables these to create personalised editions.

5.7 Use Cases Applied on Component Diagram

5.7.1 Send viewing information

Overrides: Send viewing information (see 2.5.1.4)

1. Media Consumer → send viewing information → (Service Controller):Control
2. Service Controller → send viewing information → (User Management System):User Profile
3. Service Controller → send viewing information → (Service):Control

depending on the Service's back-end, this information can be stored. For example, it can be stored in the Content Management System of the Newspaper Service that was introduced in the Functional Viewpoint, like it was done in step 3 of 2.5.1.4. Also, this information could be sent to Services that are children of this Service in the dependency tree.

5.7.2 Send context information

This is a new use case we introduce in this viewpoint.

1. Media Consumer → send context information → (Service Controller):Control
2. Service Controller → send context information → (User Management System):User Profile

The context information is sent to all interested Services:

3. Service Controller → send context information → (Service):Control

5.7.3 Other use cases affected by contest-awareness

Other use cases, such as “pull edition” and “push edition” are affected by context-awareness. More concretely, extra service-specific steps are added to these use cases in order to allow the service to make a personalised edition for each Media Consumer. For this purpose the new interface “Context Information” is offered by the Service Controller to the Service. Via this interface, a Service can request extra information about a Media Consumer's preferences and context.

5.8 Real-world scenario

In this Section, a real-world scenario is worked out to show how this context-awareness mechanism works in practice. The scenario is as follows:

Martha and Nicholas have decided to spend this weekend at the countryside, going around “the old castles route”. They receive a detailed map with the main interest points related to their preferences. This map is updating as they move forward in the route. The map also includes references to restaurants and sights of interest. Martha is mainly interested in the Middle Ages, churches and paintings and she clicks on the device to request more information about it.

Preconditions:

- The couple is registered within the publishing system and has a Consumer Profile.
- The couple's preferences (customs, history, local dishes) are registered within this profile, entered statically by them or learned dynamically by their reading behavior.
- More particularly, Martha's interests for the Middle Ages, churches and paintings are stored within her profile.
- The couple's regular residence is known to the system (this can be by experience or can be entered in the profile).
- Other than being subscribed to the newspaper service, they are also subscribed to an external service that offers touristic maps together with commercial offerings for tourists. This external service is offered by the publisher as extra functionality for their customers.
- This external touristic map service is registered within the publisher's architecture (more concretely, in the Service Controller) as being a service that is interested to know when there are long-term unusual changes to Media Consumer's localization information. This can trigger the assumption that the Media Consumer is on a touristic holiday.

The scenario is as follows:

- The couple travels to the countryside.
- The device automatically sends context information about the change in localization of the couple: use case: **send context info** (5.7.2) is executed.
- The publisher notices that the new location is unusual for the couple, and that the change in location is long-term, which means they are not just flying over the area, but actually staying there for a while. This happens within the User Management System.
- The User Management System sends this new information –this event– to the Service Controller.
- The Service Controller component sends the change in localization to the external service that offers the touristic maps, because this service was registered as being interested in this type of event.

- The external service receives the information together with the personal preferences of the couple and generates the personalised map as an edition. This edition is pushed (use case: **push edition** (2.5.4.11)) via the Service Controller to the Media Consumer.
- Further interactions between the external touristic map service provider and the Media Consumer are possible, but they will all go via the Service Controller, which allows the publisher to keep track of how many information is sent, and how billing must be done afterwards.

6 Viewpoints combined

Figure 10 contains all the viewpoints previously presented in a combined manner, giving a full overview of the architecture. Color codings are used to show in which viewpoint the origin of each element can be found. This overview proves the compatibility of each of the diagrams previously shown.

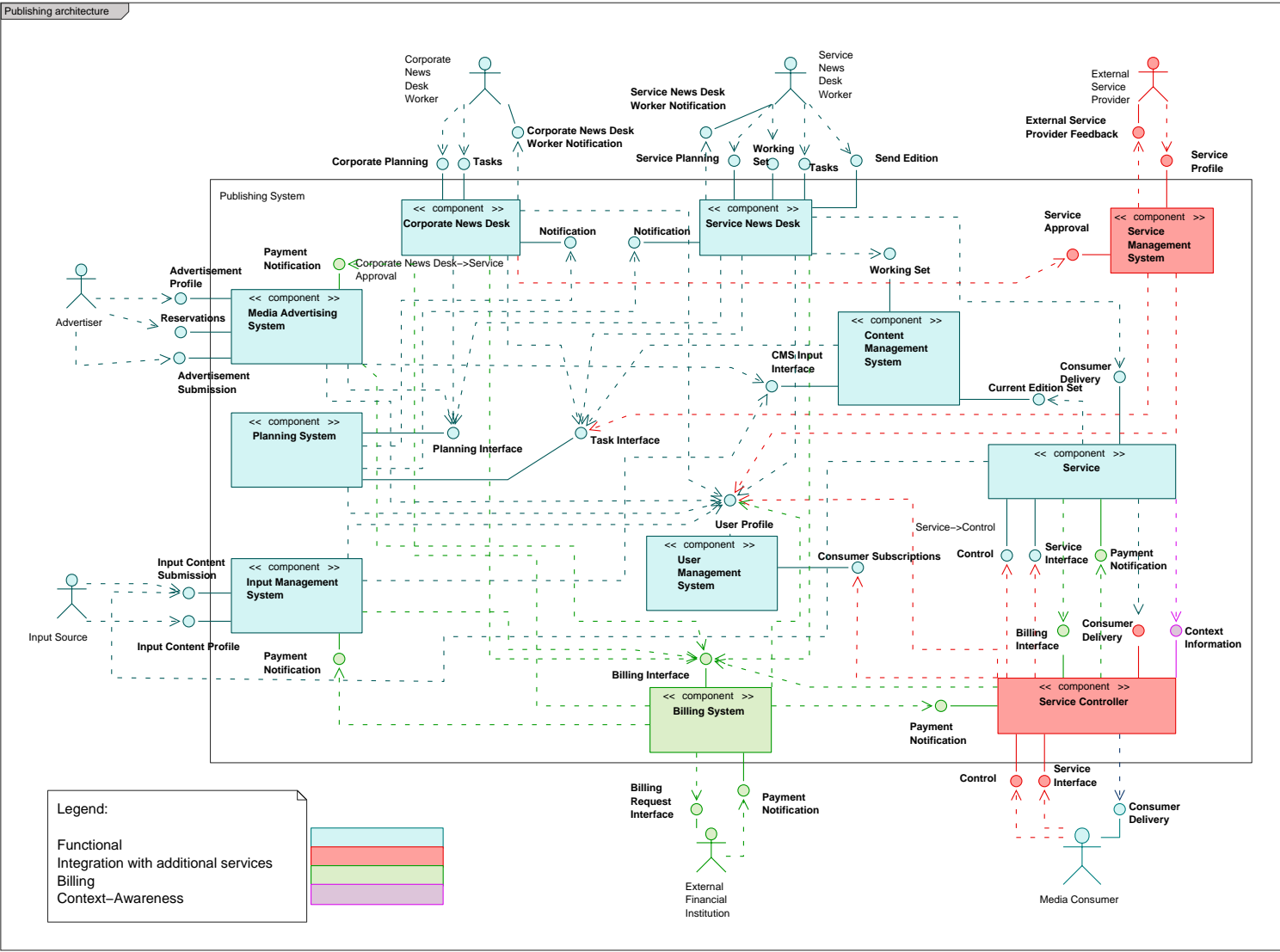


Figure 10: The full architecture, achieved by merging the separate viewpoints into one figure.

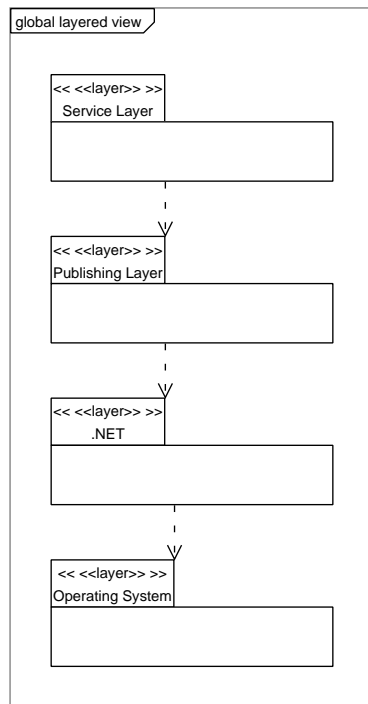


Figure 11: A layered view on the publishing system.

7 Layered view

In this Section a layered view on the architecture is presented. An issue that remained unclear until now is how existing lower-level layers or frameworks –that offer certain of the used concepts– can be reused in order to lower the entire development effort and increase reuse of existing technologies. These issues will also be addressed in this section.

7.1 Global layered view

A global view on the layering is shown in Figure 11. This view follows the basic assumption that the system will be deployed and implemented on top of the .NET platform, which in turn is deployed onto a certain operating system. One advantage of choosing the .NET platform is that it offers platform-independence: .NET assemblies are portable across different operating systems.

It must be stressed that this layered view is presented as a global archi-

tectural view and not a detailed design view. Only the global structure of the system will be explained and not the in-depth technical design. This consists mainly of interdependencies between components and frameworks stacked on top of each other.

7.2 The publishing system as a layer

One of the striking issues in Figure 11 is that what we previously considered as one system, is actually split in two layers. Indeed, from the point of view of a service the publishing system can be seen as a domain-specific middleware layer, in which it is deployed and with which it interacts. This layer provides a service with abstraction of lower-level interactions, and offers primitives – often called *enablers* or *enabling services* – such as user management, billing, context-awareness, delivery of content, orchestration of different services, and many more.

A short overview of the most important primitives offered by the publishing system to the service layer (and thus to the stakeholder, implementing these services) follows:

- **Media Consumer Management:** The publishing system takes care of media consumer subscriptions for each service. It also handles issues such as the tracking of media consumers, which is essential for a service to achieve personalisation.
- **Billing Facilities:** The external service provider should not worry about mechanisms to handle the billing of media consumers. These mechanisms are offered by the publishing platform. Abstraction is made of how payments happen between external service providers and the publisher. The reason is that this depends highly on the type of services used and on the agreements made between the service provider and the publisher.
- **Service Delivery:** The publishing framework offers the services abstraction of different types of channels that can be used to deliver their content. These channels will be categorized according to capabilities, and this grouping can be used to prepare editions for. An edition will be in a general XML-based format, independent of concrete device technologies. For example, a service could make a general edition of a story for “small-screen devices without hyperlinking capabilities”, which could then be delivered onto a GSM via SMS, or onto a PDA via WiFi. This allows services to be efficiently offered to different types of end-user devices.

- **Content Management:** For internal services, the publishing system offers the ability to use the central Content Management System for persistent storage and to link to existing data of other internal services. An example is an internal forum service which can also use the Content Management System to store all forums posts linked to a certain story in an edition. As discussed before, an internal service is a service that is closely coupled with the core newspaper service.
- **Planning and Policy:** For internal services, the publishing system offers a task mechanism, used by the service news desk that is linked to this internal service. This task mechanism allows the internal service to follow a certain service policy. The internal service can thus incorporate in the Planning System.

These primitives can be used in the implementation of a certain service, and by doing so the development effort for that service is kept low and reuse high.

Another advantage of presenting the publishing system as a layer, is that a certain degree of control can be kept over each service. The publisher, which is the stakeholder owning the infrastructure, has established a customer base which is of vital importance for his business. Keeping control over external services is a very important publisher concern, especially when dealing with the integration of third-party services that possibly aren't 100% trustworthy. One of the running examples is the restaurant system that integrates within the newspaper by sending personalised restaurant advertisements. Clearly, the publisher wants control over the interactions between the restaurant system and the media consumer, for reasons such as billing and privacy.

When presenting the publishing as a layer on which services are deployed, the Service Controller could be seen as the binding between the service layer and the publishing layer. Indeed, it was defined earlier to be the single point of access for a service into the publishing system. Thus, in order to successfully implement a new service, only the API of the Service Controller should be known to the developer of this new service, and the new service must only be aware of this Service Controller.

The layer of this system that offers all these primitives to each service could be implemented by means of a so-called "service-oriented architecture". This is a middleware layer which is centered around the concept of a service. It manages many facets such as service management, user management, Using an already existing service-oriented architecture has the benefit that interoperability with external parties is facilitated since service-oriented architectures typically use open standards to achieve this. To be a

good candidate for this system, a service-oriented should at least offer the primitives listed above.

A more detailed view on the different layers is shown in Figure 12. Note that not all component instances are shown for reasons of simplicity. Because the actual publishing layer will be on top of the .NET framework, it will make use of the primitives and classes that are offered by this framework. Some examples are depicted. For example, the publishing system will use the .NET webservice facilities to enable standardized interprocess-communication. Many existing libraries and primitives will be used to aid the development of the publishing system and to avoid reinventing the wheel. Indeed, it makes a lot of sense to use existing libraries that have proven correctness and offer reusable primitives. An example is using existing –and proven– algorithms of encryption instead of trying to develop one yourself. This approach allows faster development and allows developers to focus on what really matters in the publishing system.

Again, the .NET framework itself will use lower-level primitives that are offered by the operating system. For example, .NET remoting or .NET WebService facilities will in turn make use of the TCP/IP primitives that are usually offered by the operating system.

7.3 Rationale for a layered architecture

Some advantages were mentioned before, but an overview of advantages for following this layered approach follows:

- **Control over services:** Because services might represent external actors interacting with the system (the External Service Providers), they can not always be trusted. Forcing services to interact with the Media Consumers through the publishing system instead of directly connecting these two actors allows the publisher to keep control over interactions between both. The publisher is not in the role of just a connector between (a.o. third-party) services and Media Consumers, but it is in the role of both a connector and a controller.

One illustration of this control is the fact that information about Media Consumers is passed on to the services on a need-to-know-basis, due to privacy concerns. An example of this is the Restaurant Service that adds advertisements to the newspaper. If a Media Consumer requests an edition, he is not aware that this request will trigger a request to the Restaurant Service for related advertisements. So in fact, the Media Consumer requests information from a service without being aware of that. It is clear that this implicit and maybe unwanted request for

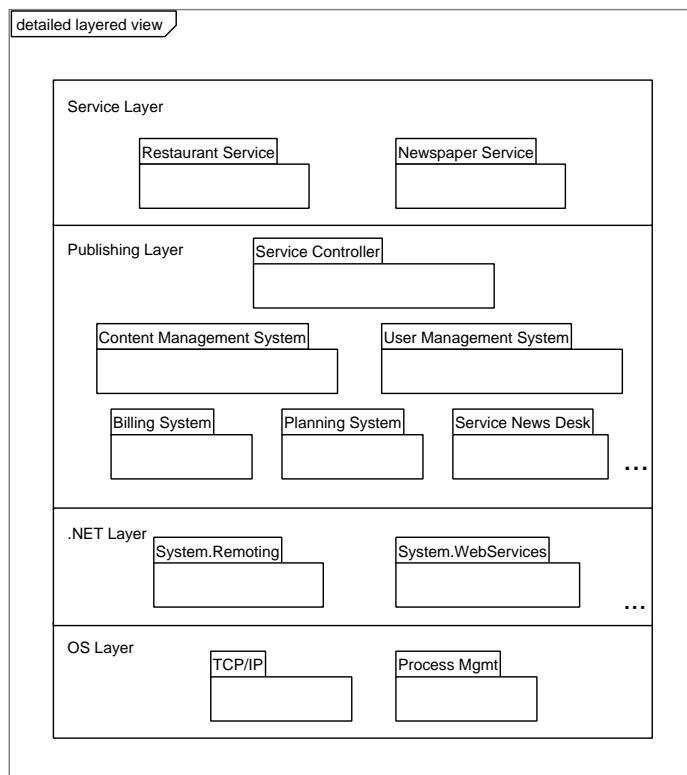


Figure 12: A more detailed view on the layering in the architecture.

information should not give the restaurant service access rights to the entire user profile of this Media Consumer. Indeed, the publishing platform will want the ability to limit the user information sent to the restaurant service to information about food preferences and perhaps location. Of course, much depends on the Service Level Agreement between the publisher and the service provider, and on the agreements between the Media Consumer and the publisher.

- **Reuse of primitives offered:** Because the publisher already offers primitives such as for example billing, the service provider who wants to develop a new service can easily reuse these primitives. This allows him to lower the entire development effort for such a service, because he does not have to focus on implementing different billing mechanisms and making agreements with different financial institutions to achieve this. The reuse of all these primitives is clearly an advantage that benefits the easy development of new services.

Of course, this decision represents a trade-off that was made. One major disadvantage for this approach is:

- **Centralisation:** Because each service must access the publishing system in order to reach the Media Consumers, this publishing system –which is a single point of access– is a single point of failure. Indeed, if the publisher becomes unavailable, nobody will be able to access his newspaper and related services. Also, this centralisation affects performance: the publishing system risks becoming a bottleneck for the entire distributed system. In this architecture, this disadvantage will be covered by using deployment tactics that increase availability and performance. We refer to Section 8 where we go into detail about these deployment tactics.

7.4 Illustration of this layered view.

The layered view allows an “Open System Interconnection (OSI) model”-type view on the system. It shows how different layers offer each other different primitives and how certain requests travel through the application stack. An illustration of this type of view is given in Figure 13.

In the publishing system, the different layers are presented. The grey boxes represent primitives that are offered in that layer to the above layer.

The focus in this document is on the specifics of the publishing system and not on user device specifics. Because of that, no technology-specific view can be presented on a Media Consumer (which is the human and his user device).

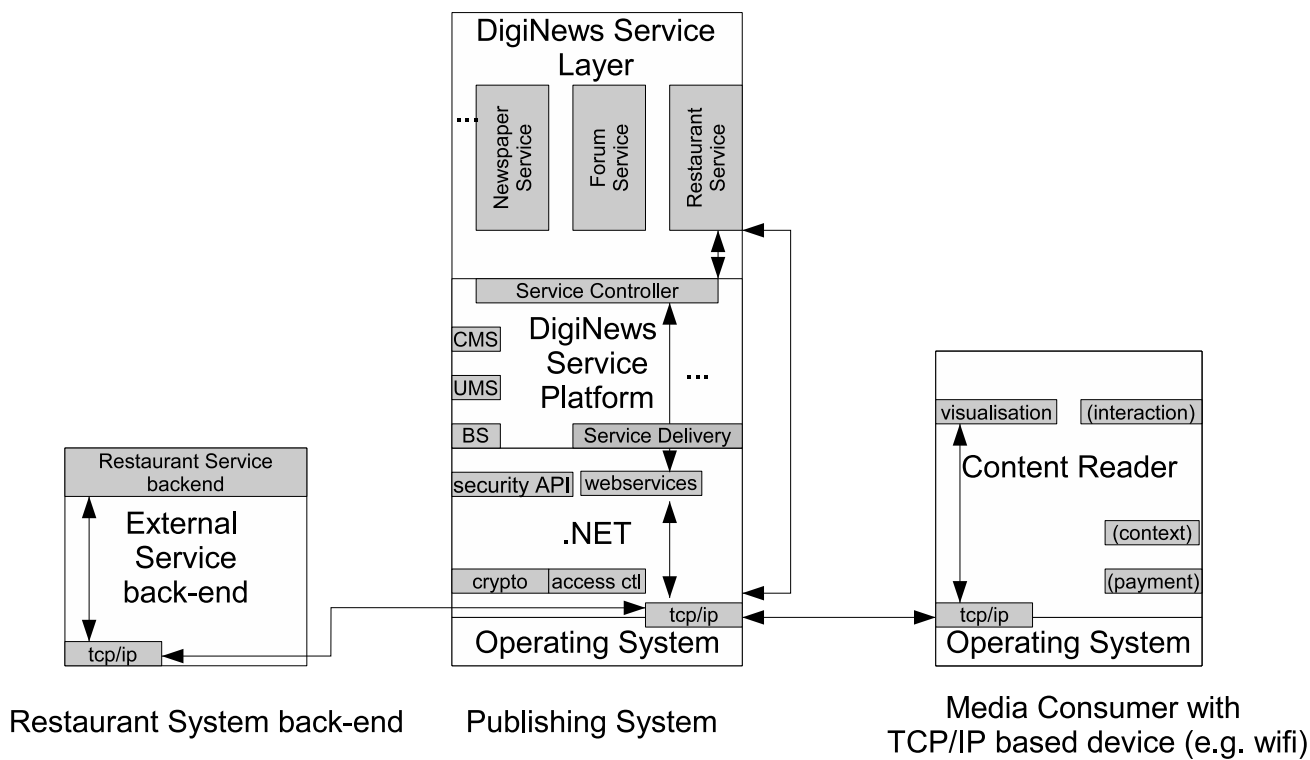


Figure 13: An illustration of how the layered view allows an OSI model-type view.

Figure 13 only shows one concrete example of a customer with its device. In this example, delivery of content happens through the TCP/IP stack, which could be part of the operating system on that specific device. Evidently, the device will have some kind of visualization functionality, which consists of the logic to render the delivered content onto the hardware display that is attached to it. A possible other functionality that a device may offer is the functionality enabling interaction with the publishing platform, by sending viewing information back. It may also offer functionalities enabling context-awareness, by sending context information to the publisher (for example localization information originating from a GPS hardware module attached to the device). It may even offer functionalities allowing payments to be done (for example a smart card reader). The sky is the limit for types of devices and device capabilities and much innovation and promising research is still being done in this field.

The middle block in Figure 13 is a representation of the publishing system. The upper layer is the service layer. In this particular situation it contains three services: a newspaper service, a forum service and a restaurant service. The restaurant service is integrated in the newspaper service, in the way that it adds advertisements which will be presented as part of the newspaper. These services are built on top of the second layer, the publishing layer. As discussed before, the single point of access for a service in this second layer is the service controller. The service controller provides abstraction for many of the lower-level interactions needed for example to accomplish billing. It also takes care of user management and much more. Because this layer was chosen to be built in the .NET platform, many of the concepts and primitives offered by .NET will be used to achieve this. Some examples are the .NET cryptography routines, or the .NET webservice libraries. Again, because the .NET library is platform independent, it provides an abstraction layer for the concrete operating system that runs underneath. However, some of the underlying concepts implemented by the operating system will be used, such as TCP/IP in this case, which enables the communication over an IP network.

The interactions shown (represented by arrows) are the interactions needed for execution of the “pull edition” use case to the newspaper, in the situation where an external restaurant system is involved.

Figure 13 also shows how the Restaurant Service component, which was a very basic component, responsible for translating and forwarding requests to its back-end bypasses the publishing layer to achieve this. Indeed, the publisher does not need control over what content is sent between these two systems.

The added value for this illustration is that it may clarify how lower-level

primitives offered by .NET and the operating system are used to allow the required communication between processes.

8 Deployment view

The deployment view provides a mapping of the previously defined component instances onto physical nodes (servers) in a computer network. This type of view is called an “allocation view” because it allocates certain component instances to concrete servers or nodes. It shows how this allocation affects certain quality attributes such as a.o. performance, scalability, availability. Because providing a strictly fixed deployment topology would be too restrictive in the scope of this document, this deployment view will show some general “deployment tactics”, and give an example of a concrete situation in which they will be applied. These tactics can be reused, depending on the actual scale at which the publisher wants to deploy the application. First, an overview is presented of the quality attributes that are relevant and affected by deployment. Then, the deployment tactics are presented, and their effect on these quality attributes are shown.

8.1 Quality Attributes

This section will give a short overview of the quality attributes that are affected by the deployment of the publishing system.

- **Performance:** Performance is the quality attribute that represents how well the software performs. A more concrete example of a performance requirement is a requirement stating that the request for an edition by a media consumer must be done in a reasonable and predefined timespan.
- **Scalability:** The presented system must show no significant performance degradation when it is scaled up towards more usage. One example is the scenario where the publisher’s client base grows, which should not lead to radical performance degradation, and this scaling-up must by definition be possible.
- **Availability:** Since unavailability means loss of business continuity for the publisher, the system should be fault-tolerant. For example when a certain network connection drops, or a server crashes, the newspaper must still be maximally available.
- **Recoverability:** This is a measure for the ease to fix the system after a software or hardware failure. An example of such a requirement is, when a hardware server suffers a hardware failure, this must be fixed after maximally half an hour.

- **Security:** Even though security has also a functional side (it is in the list of viewpoints), it is also affected at the deployment level. One example is the achievement of certain security goals by placing a firewall. Another example is the use of replication that could be used to avoid denial of service (DoS) attacks, so that if one server gets overloaded, other servers are still available to take over.
- **Modifiability:** Modifiability is a quality attribute that represents how easy it is to make modifications to the system. One example of a modification in the publishing system is the addition of a single service. This is affected by deployment because adding a service means deploying a new component on a server.

It must be stressed that these quality attributes are not the only ones of interest when building a publishing system. They are the ones that are affected by the actual deployment. Other quality attributes have either already been taken care of (e.g. extendability, modifiability: amongst other places in the Section about Integration with Additional Services (Section 3)), or out of scope for this document (e.g. usability, affordability, ...). The main quality attributes on which the focus lies in this deployment view are availability and modifiability.

8.2 Deployment

The deployment plan in Figure 14 presents a mapping of the previously presented components onto network nodes. These nodes represent physical servers and are shown as 3 dimensional boxes in a deployment view. A concrete deployment situation is shown in Figure 14. The physical servers that are placed in the publisher's infrastructure are in the white rectangle. Figure 14 illustrates the three deployment tactics that were used:

- **Replication:** This deployment tactic is applied on the Service Controller machine and Data Server machine. It is represented by a grey node with a number in the lower right corner. The number shows the number of times the contained components are replicated to increase availability, scalability and to allow load-balancing. The exact meaning of this notation is made more explicit in Figure 15. In the deployment plan of Figure 14, the Service Controller machine is replicated five times and the Data Server machine is replicated three times. Of course, this number depends on the availability one wants to achieve in a particular situation and thus on the expected workload for these servers.

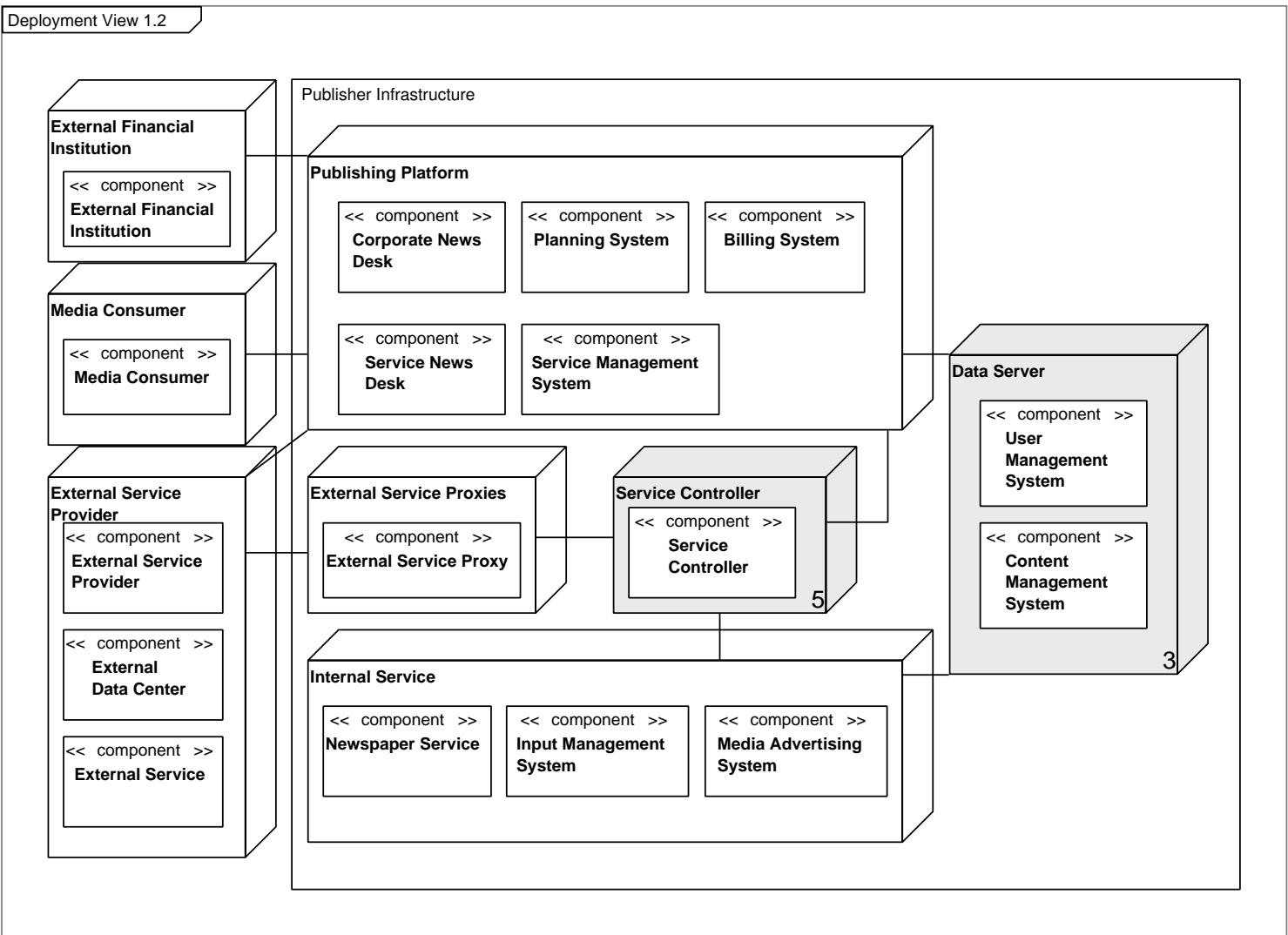


Figure 14: A concrete deployment plan, showing the mapping of the components onto physical nodes in one particular situation, illustrating the different deployment tactics that were used.

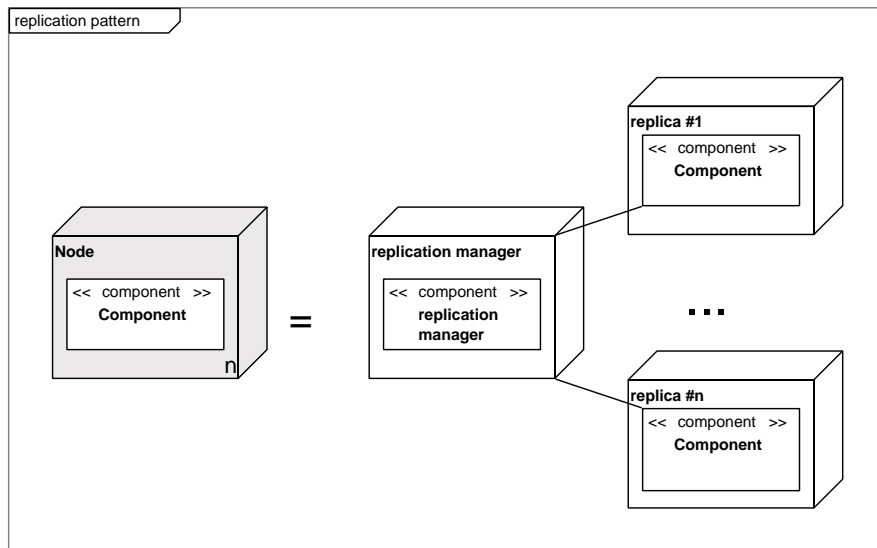


Figure 15: The replication tactic used.

The replica manager handles synchronization of the different replicas, together with possible load-balancing over these replicas.

- Internal Service Deployment:** As shown in Figure 14, the internal newspaper system is deployed onto a single machine. This illustrates this deployment tactic, which states that each internal service will be deployed on a separate machine. Each internal service is expected to generate quite some computational load (because of the need to personalise editions, etc). By employing this tactic, guarantees are made that these services have no influence over each other: if one service gets overloaded, this will not affect the other internal services in any way. They will perform independently of each other. In this way, availability is benefitted, but also modifiability, because adding a new service is as easy as placing a new server. The deployment of existing servers will not be affected by this in any way.
- External Service Deployment:** External Service Providers who want to offer a certain service to the Media Consumer must develop and deploy a very basic Service proxy component in the infrastructure of the publisher, as was explained in Section 3. These proxy components will basically translate and forward each request to such an external system of the third-party service provider. This operation is not expected to

introduce a large computational load, and that is the reason why these proxies will all be deployed on the same machine, namely the “External Service Proxies” node. Adding a new external service will then be as easy as deploying a new Service proxy component on this machine, in the benefit of modifiability.

8.2.1 Conclusion

As explained before, this is no fixed deployment. Figure 14 shows one particular deployment situation. The deployment shown here is for the situation with only one internal service (a Newspaper service) and one external service (e.g.: Restaurant service), and where the Service Controller is replicated five times and the Data Server three times.

The added value for this view is that it explains and demonstrates the deployment tactics that are used to support mainly the quality attributes modifiability and availability (but also other quality attributes, such as scalability). As mentioned before, an important fact of these deployment tactics is that they can be reused.

9 Conclusion

This report provides a definition of an architecture that can be used to implement a digital newspaper system. This architecture is developed in an incremental manner: for each of the concerns that were identified in [?] such as functionality, context awareness, billing, etc, the requirements are applied to the architecture. In the end, these different concerns are integrated in a full view on the architecture. Furthermore, a layered view and a deployment view are provided.

The ultimate goal is that this report provides enough information so that implementers are able to implement the digital publishing system, while still leaving certain decisions open for the programmers to better suit the concrete context. The architecture puts certain constraints on the implementer, but it remains open in the way that certain choices w.r.t. actual implementation are left open to allow future innovations to be adopted in this system.

10 Acknowledgements

- Kristof Vandebroek: for the initial work on this document.
- Koen Buyens: for a thorough quality check.

References

- [1] W. Joosen, S. Michiels, E. Truyen, K. Vandebroek, and D. Van Landuyt. A comprehensive model for digital publishing. Technical report, K.U.Leuven, Dept. of Computer Science, Leuven, Belgium, Apr. 2006.
- [2] T. Mahieu, W. Joosen, D. Van Landuyt, K. Buyens, and E. Truyen. System requirements on digital newspapers. Technical report, K.U.Leuven, Dept. of Computer Science, Leuven, Belgium, Mar. 2007.