Bible Information Modelling

Patrick van Bommel
Department of Information Systems
Subfaculty of Computing Science, University of Nijmegen
P.O. Box 9010, 6500 GL Nijmegen, The Netherlands
E-mail: pvb@cs.kun.nl

Abstract: In this paper we address the question how information models of the Bible can be constructed. It is evident that building a generally acceptable and satisfactory model of the Bible is extremely difficult, if not impossible. Consequently, this paper contains no plug-and-play solutions. The intention of this paper is to examine the challenges in Bible information modelling. Specific attention will be given to navigation models and models of Bible contents. Also, models of Bible readers will be considered. In our project, reader models will be applied for the construction of personalized versions of the Bible. We propose several basic instruments for structuring information in the context of the Bible.

Keywords: Information Modelling; Content Analysis-Bible; Navigation Model

Received 19 March 2003; Accepted 21 April 2003

1. Introduction

The focus of this paper is *Bible information modelling*, where contents as well as structure of the Bible is to be described. We do not wish to provide complete information models. Rather, we introduce instruments to improve the accessibility of the Bible for readers as well as for researchers.

When contents and structure of complex information sources have to be modelled, we usually apply concrete *modelling techniques*. In earlier work (see e.g. [1]), we specified the formal syntax and semantics of such modelling techniques. Note that Bible information modelling involves traditional data modelling as well as document modelling.

It is evident that structuring mechanisms for Bible information have already been used for a long time. Well-known examples are concordances (e.g. [2]) and the literature approach (see e.g. [3]). More advanced modelling has also been examined. Previous work in this area has been focussed on the derivation of primitive modelling constructs for Bible texts, such as text hierarchies (see e.g. [4], [5]). Although this is an important step forward in Bible modelling, we do need more powerful instruments for structuring Bible information.

The instruments introduced in this paper, are based on high-level information structuring mechanisms aiming at abstraction, such that Bible access may become more flexible and more intelligent, and can be based on modern technologies. It is stressed here that the results of information modelling do not only affect computerized Bible applications. Manual access mechanisms may become more advanced as well. For computerized Bible applications, one may think of CD-ROM applications as well as Internet applications. We performed an in-depth analysis of typical techniques for modelling web sites in [6].

In our project we consider modelling from several perspectives. An overview of the models, their relations, and the possibilities for supporting readers, is given in Figure 1.

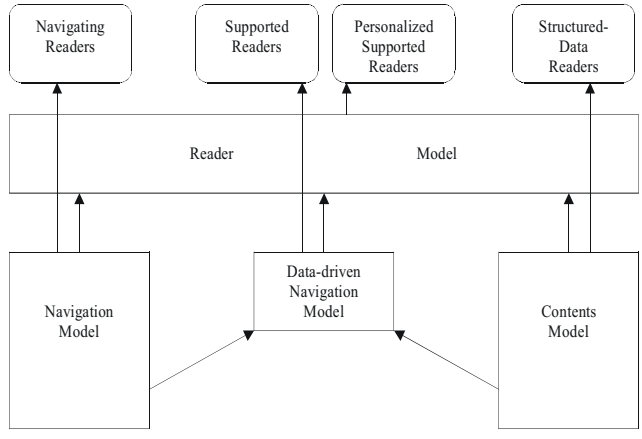

Figure 1 Overview of models and their relations

In order to illustrate the potential of the models discussed in this paper, we mention the following aspects:

a. Information contents modelling: what information is contained in the Bible? This is a sort of content analysis.

b. Structure modelling: what structures can be used to interrelate different parts of the Bible? This also includes navigation mechanisms.

c. Bible database specifications: how can Bible databases be specified? This includes automatic derivation of database structures.

d. Dialogue representations: how can dialogues between the information service at hand (i.e. the Bible) and their users (readers) be established?

The Bible kernel model, discussed in this paper, enables us to partially model the contents of the Bible. Besides this kernel model, other instruments are introduced, including Bible navigation mechanisms and belief modelling. It may be argued that the kernel presented in this paper is inaccurate, as we only discuss the simplified version here. However, entirely different approaches to information modelling can be embedded within our framework as well. As an example of a different modelling approach, in earlier research in another context we introduced *logbooks* as a general starting-point for information modelling (see e.g. [7]).

The organization of this paper is as follows. In section 2, we consider a mechanism to model the different paths through the Bible. Different readers will read in different order, and even the same reader will not be consistent in this. We therefore need a flexible navigation mechanism. In section 3, we present the underlying data model for navigation, while section 4 focuses on data models for Bible contents. In contents modelling we go down to the level of concrete data items, also called *facts*. In section 5 we discuss the possibility of modelling the background of the reader. In our project such reader models will be used for the construction of *personalized* versions of the Bible. In the examples in this paper, concrete Bible verses are applied. We use the Authorized King James version [8] and the New King James version [9].

2. Supporting different paths through the Bible

There are several ways to read the Bible. Sometimes, we want to start at the first page, whereas in other occasions we want to start at a specific verse in some chapter. Sometimes we follow references or comments, and sometimes we jump between specific parts in order to focus on a given topic or theme.

In this section, we propose an instrument, which can be used as a support for different ways of reading. This instrument is based on *navigation mechanisms*. In section 2.1, the central notion of *information unit* is introduced. Then, we consider decomposition of information units (section 2.2) and references between information units (section 2.3). The underlying data model for navigational instruments will be presented in section

2.1. Information units

Bible navigation models are based on the notion of *information unit*. An information unit is a separate container for information. In order to deal with information units in a structured manner, we use *information unit types,* or unit types for short. We say that *an information unit is of a certain type*, or *an information unit has a certain type*. A unit type may be considered as a group or category of units.

In a given application, we let U be the set of information unit types. Some unit types are given in the following example:

> U = {*bible, part, book*, *chapter*, *section*, *verse*, *comment*}

Unit types are instantiated with unit *instances*, or units for short. In our theory, the universe of all instances is denoted by Γ. We use the instantiation function $\text{Inst}_U: U \rightarrow \wp(\Gamma)$ to assign sets of unit instances to elements of U. Note that elements of the powerset $\wp(\Gamma)$ are subsets of Γ. As an example, consider the following units:

> Inst_U(bible) = {*King James, Catholic}*
>
> Inst_U(part) = {*Old Testament, New Testament, Deuterocanonical books}*
>
> Inst_U(book) = {*Book of Job*, *Book of Proverbs*, *Gospel of Matthew*, *Book of Romans}*
>
> Inst_U(chapter) = {*Psalm 19*, *Psalm 51*, *Chapter 4*, *Chapter 15}*

Although in this case *Chapter 4* is declared to be an information unit of type *chapter*, it is not clear yet what the context of *Chapter 4* is. It may be a chapter in the Book of Job or in another book. In order to link information units to other information units, we need containments and references. These will be discussed in the next sections.

2.2. Hierarchical decomposition of information units

Links between unit types are modelled in the navigation model N = <U, C, R>, where $C \subseteq U \times U$ is a set of containment types and $R \subseteq U \times U$ is a set of reference types. We first consider containment types.

Containment types are used to specify structural rather than referential relationships between units. In our approach, containments specify hierarchical

decomposition of information units. A containment type $(v,w) \in C$ expresses that units of type *v* may contain units of type *w*. Typical containment types for structural navigation through the Bible are given as follows:

$C = \{$(*bible*, *part*), (*part*, *book*), (*book*, *chapter*), (*chapter*, *verse*)$\}$

So, for example, units of type *chapter* may contain units of type *verse,* and units of type *book* may contain units of type *chapter*. We can also read *books may contain chapters* and *chapters may contain verses*. Note that we say *may* contain, rather than *must* contain. Note furthermore that in this definition, books do not directly contain verses. But after a closer inspection, we see that this is too restricted, since there are books containing verses but no chapters. As an example, consider the book *Epistle of Jude*. As a consequence, we do need $(book,verse) \in C$.

can be used to make such restrictions explicit and to prevent future updates of $Inst_c$ from becoming illegal. An example of such a constraint is:

King James has no Deuterocanonical books:
$(KingJ,DeutCn) \notin Inst_C(bible,part)$

In the formulation of constraints, an important aspect is the *expressiveness* of constraint languages. The constraint mentioned above does not need additional operators. Some constraints however do require additional operators, such as (universal or existential) quantifiers. An example of such a constraint is:

The Epistle of Jude has no chapters:

$$_{x \in \Gamma}\,[(Jude,x) \notin Inst_C(book,chapter)]$$

In information models, integrity constraints play a vital role. Some basic constraints in the context of Bible information modelling are treated in section 3.3. These

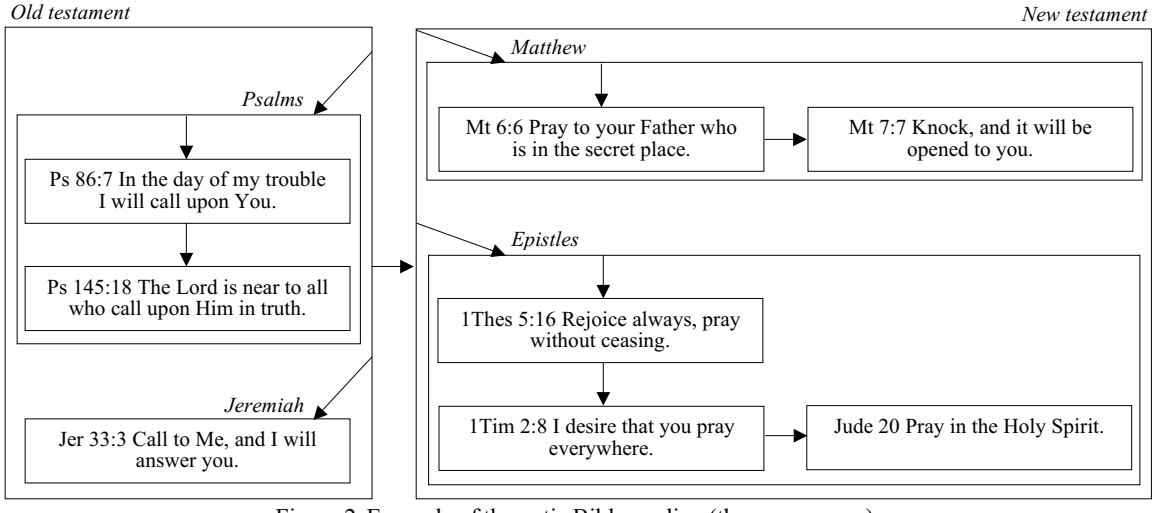

Figure 2 Example of thematic Bible reading (theme = *prayer*)

Containment types are instantiated with containment instances, or containments for short. We use the instantiation function $Inst_C: C \rightarrow \wp(\Gamma)$ to assign sets of containment instances to elements of C. As an example, consider the following containments:

$Inst_C(bible,part)=\{(KingJ,OldT),(KingJ,NewT),(Cathol,OldT),(Cathol,NewT),(Cathol,DeutCn)\}$

$Inst_C(part,book)=\{(OldT,Psalms),(NewT,Matthew),(DeutCn,Manasseh)\}$

$Inst_C(book,chapter)=\{(Psalms,Psalm19),(Matthew,Chapter5)\}$

Note that in this instantiation of containment types, we see $(Cathol,DeutCn) \in Inst_c(bible,part)$. This indicates that the Deuterocanonical books (also called Apocryphal books) are in the Catholic Bible, but not in the King James Bible. To be more precise, integrity constraints

constraints deal with uniqueness and mandatory roles in facts. Navigation models can be used for a variety of purposes. Here we give an example in the context of *thematic Bible reading*. In Figure 2 we have modelled a navigation structure for verses about *prayer* (verses found in [9])

In the model in Figure 2, we see several examples of containments. Some of them are (Old testament, Psalms), (Old testament, Jeremiah), (New testament, Matthew), and (Matthew, Mt 6:6). Containments are often shown using arrows. However, not all arrows in Figure 2 are valid containments. For example, although we see an arrow from unit *Ps 86:7* to unit *Ps 145:18*, it is evident that Ps 145:18 is not contained in Ps 86:7. This is a typical example of a reference between information units. These will be discussed in the next section.

2.3. References between information units

In this section, we consider reference types RÍU´U. A reference type (v,w)ÎR expresses that units of type *v* may refer to units of type *w*. Note that this is a possibility rather than a requirement. So, there may be units of type *v* that do not refer to any unit of type *w*. Typical reference types are:

R = {(verse,verse), (verse,chapter), (verse,book)}

Reference types are instantiated with reference instances, or references for short. We use the instantiation function $Inst_R$:R®Ã(G) to assign sets of reference instances to elements of R. As an example, we consider the following references:

$Inst_R$(verse,verse)={(John10:34,Ps82:6), (Rom8:36,Ps44:22), (Heb11:5,Gen5:24)}

$Inst_R$(verse,chapter)={(John10:14,Ps23),(John14:27,Heb4)}

The (verse,verse) references given above are found in [9]. Note that the reference type (verse,book)ÎR is used for references from a verse to a book. This reference type is used only in special cases, where the target of a reference is not in particular verses or chapters, but in an entire book.

Although (verse,verse)∈ R is a reflexive reference type, we do not allow reflexive instances. This can be stated explicitly in the following constraint:

No reflexive references:

$_{x\in\Gamma}$ [(x,x) ∉ $Inst_R$(verse,verse)]

Note that, in some cases, reference type (verse,verse) may lead to unnatural instantiations in the case of verse sequences. Then, it would be more natural to have an explicit unit type *verses*∈ U and a reference type (verse,verses)∈ R. It is obvious that *verses* is not an elementary unit type, as it may be simulated by multiple instances of *verse*. As an example, the following instantiations are equivalent:

$Inst_R$(verse,verses)={(John10:14,Ps23:1-3)}

$Inst_R$(verse,verse)={(John10:14,Ps23:1), (John10:14,Ps23:2), (John10:14,Ps23:3)}

3. Underlying data model for navigation

In this section, we discuss the underlying data model for our navigation mechanism. We first consider units, containments, and references in section 3.1. Then we address the topic of Bible queries in section 3.2. Finally, section 3.3 presents integrity constraints and organization of units in pages.

3.1. Units, containments, and references

For the notation of data models, we adopt the well-known graphical convention, where a circle represents an object type and a box represents the role played by an object type in a fact type. This circle-box notation is widely used (see e.g. [1], [7], [1]) and is more or less equivalent with entity-relationship notations ([2]). Object types are connected via fact types, expressing that objects of a given type can *play roles* in facts of a given type.

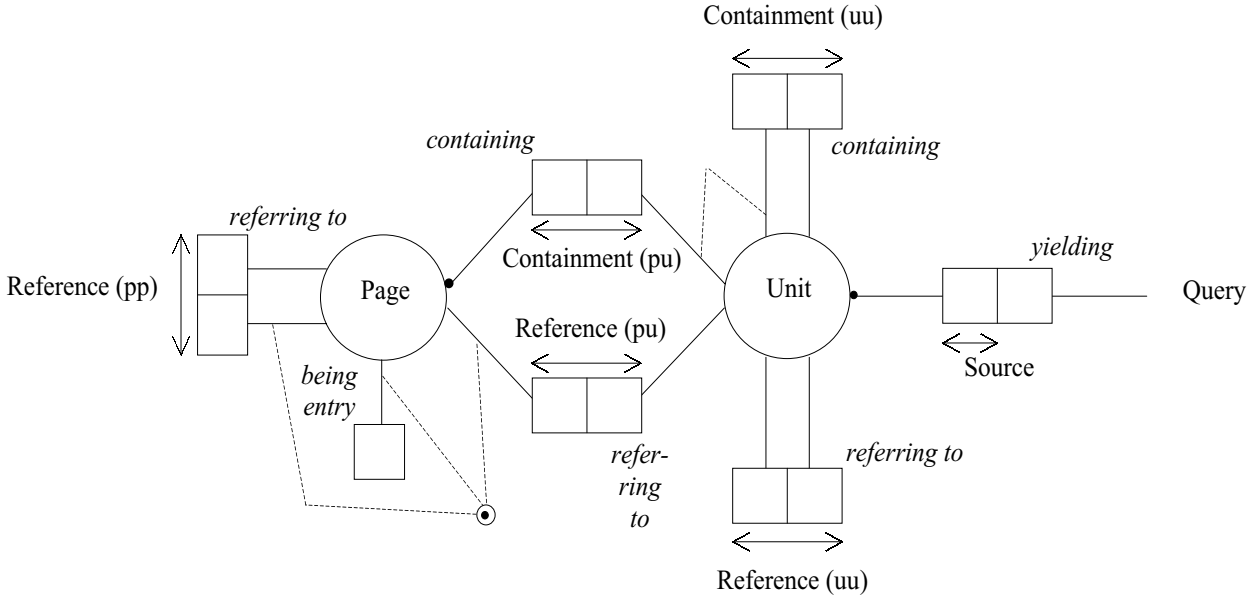

Figure 3 Underlying data model for navigation mechanism

The central notion of information unit is given in the circle in the middle part of Figure 3. We see fact type *containment (uu)* for unit-unit containments, which were discussed in section 2.2. Fact type *reference (uu)* is intended to record unit-unit references, as discussed in section 2.3.

Besides units, containments, and references, the model in Figure 3 contains several other aspects. In the right part we see *queries*, while the left part contains *pages*. Also we see some additional symbols, such as arrows and black dots. These will be explained in the rest of this section.

3.2. Bible queries

Each information unit presents information in some form. The construction of units and the information contained in them, can be based on *queries*. In the underlying data model from Figure 3, queries are given in the right part. A query yielding a given unit is recorded by fact type *Source*. For the formulation of queries, we can use structural properties as presented in section 2, such as verses occurring in chapters in books. Moreover, queries can be based on data models for the contents of the Bible. Although we do present some contents-driven queries here, a data model for Bible contents will be presented in a later section (see section 4).

In this paper, we use a path-based query language, as is often done in the context of data models (see for example [3ter Hofstede, A.H.M., Proper, H.A & van der Weide, Th.P (1993)

> Formal Definition of a Conceptual Language for the Description and Manipulation of Information Models,
> *Information Systems.*
> 18(7).]). Query paths for *traversal* of information organized as trees have been introduced in e.g. [4].

In order to illustrate the possibilities, we examine several parameterized queries containing variables *x*, *y*, *z*. First consider the following basic queries:

> Chapter *x* in Book *y*
>
> Verse *x* in Chapter *y* in Book *z*

Using such queries, we can specify a specific Bible text and put that text in a given information unit. The basic queries given here are specifically intended for the selection of chapters or verses (or other structural components), based on the containment structure. As an example, we can have units containing the information specified by the following queries:

> Chapter 6 in Book 'Gospel of Matthew'
> Verse 12 in Chapter 37 in Book 'Exodus'

Often, the selection of text is also based on *keywords* or *topics*. Then, selection is not only based on the containment structure for units, but also on the topics a unit contains. We may for example have the following queries for the selection of specific verses:

> Verse about Topic *x* in Book *y*
> Verse about Topic *x* or Topic *y* in Book *z*
> Chapter in Book in Part *x* about Topic *y*

Here the second query shows that we can combine topics in logic expressions based on *and* and *or*, as is usual in information retrieval. Examples of topic queries are the following:

> Verse about Topic 'crucifixion' in Book 'Gospel of John'
>
> Chapter in Book in Part 'Old Testament' about Topic 'shepherd'

The first query specifies all verses in the Gospel of John about the crucifixion. The second query specifies all combinations of (chapter, book) about the topic 'shepherd'. As an illustration, this contains (Psalm 23, Psalms) and (Chapter 34, Ezekiel).

More complex queries can be expressed on the basis of data models as will be presented in section 4.2. Examples of these queries are:

> Action performed by Person *x* at Location *y*
>
> Action performed by Person *x* in Book *y*
>
> Person performing Action *x* concerning Person *y* in Book *z*

The first two parameterized queries specify actions satisfying the conditions expressed by *x* and *y*, whereas the third parameterized query specifies persons satisfying the conditions expressed by *x*, *y* and *z*. Examples of these queries are:

> Action performed by Person 'Jesus' at Location 'Galilee'
>
> Action performed by Person 'Pashhur' in Book 'Jeremiah'
>
> Person performing Action 'Talking about' concerning Person 'Jesus' in Book 'Acts'

These queries specify the following information. Firstly, we have all actions performed by Jesus at Galilee. Secondly, we have all actions performed by Pashhur in the book Jeremiah. Thirdly, we have all persons who talked about Jesus in the book Acts.

3.3. Enhancements

Information units are often organized into pages. The notion of page can be interpreted in several ways. Sometimes we have fixed-length (e.g. hard copy or paper) pages, whereas in other occasions we have logical pages (e.g. web pages).

We will explain the fact types in Figure 3 dealing with pages. A page contains units, which is called a page-unit containment structure. Pages and their units are recorded in fact type *containment (pu)*. Also, units may refer to pages, which is called a page-unit reference structure. Units and their referenced pages are recorded in fact type *reference (pu)*. Moreover, pages may refer to other pages, which is recorded in fact type *reference (pp)*. Finally, a page can be an entry page (e.g. a portal), which is recorded in a unary fact type consisting of a single role called *being entry*.

Note that the data model in Figure 3 can also be used for the formulation of queries concerning the navigation structure. Examples of such queries are:

> Unit contained in Page referring to Page being entry
>
> Unit referring to Unit *x*
>
> Page containing Unit yielded by Query *x*

Note furthermore that we can apply integrity constraints as is usual in data modelling ([10], [7], [1]). In Figure 3 we see two kinds of constraints: uniqueness constraints (denoted by arrows) and total role constraints (denoted by black dots).

We first explain the uniqueness constraint. Usually, a fact type represents a many-to-many relationship, which is denoted by an arrow over two roles. In e.g. fact type *reference (uu)* the many-to-many relationship allows us to have units referring to more than one unit and units referred to by more than one unit. On the other hand, we see that fact type *source* has a uniqueness constraint over a single role. This expresses that the information contained in a unit is yielded by at most one query.

A total role constraint expresses that all objects of a given type *must* play a given role in a given fact type. For example, we require units to contain information and pages to contain some unit. In Figure 3 this is expressed by total role constraints as follows: each unit must have a source query and each page must contain a unit. Furthermore, each unit must be contained in a page or in another unit. Finally, each page must be referred to by a unit or by another page, or they must be an entry page. These requirements are important to guanrantee that all units and pages are reachable.

4. Bible contents modelling

In this section, we consider models for the *contents* of the Bible. Models for the *structure* of the Bible and for *navigation* through that structure, have been introduced in section 2. Note that modelling the Bible contents does not mean we ignore structure completely. In contents modelling, the attention is focussed on information structure, rather than document structure.

We first describe the intention of contents modelling in section 4.1. Then we present a *kernel model* for contents specifications in section 4.2. In this kernel model, a distinction is made between fact *types* and fact *instances*. Section 4.3 gives fact instances. In section 4.4, we focus on a more advanced concept in information modelling: composition of fact types and composition instantiation.

4.1. Intention

Bible information models are expressed in terms of object types and relation types between object types (also called fact types). These object types allow us to perform a variety of tasks, manually as well as computerized. A basic task is:

> **Component modelling: define models for specific parts of the Bible. This basic task results in a fundament for all other tasks.**

Several additional tasks are possible, focussing on e.g. sequencing, comparing, and versioning:

a. *Component sequencing:* present parts of the Bible in a predefined order or in a user-driven order.

b. *Component comparison:* compare different Bible parts. This structuring mechanism is an aid in finding syntactic and semantic relations between different Bible parts. In this context, we also consider differences, similarities, and references.

c. *Formulation comparison:* compare different formulations of the same Bible part.

d. *Component versioning:* offer Bible parts to readers in different modes. Here, different versions may be presented to different readers, depending on their background. Also, different versions may be presented to the same reader, depending on the development of the reader's knowledge or belief.

Note that several strategies for the definition of *versions* may be considered. A major challenge here is to apply knowledge of the reader's background in order to generate *personalized* versions of the Bible.

It is not clear whether there are fundamental differences between conceptual modelling of Bible information and

conceptual modelling of information in administrative environments. However, when we specify models of the Bible, we should consider aspects that are usually outside the scope of traditional information systems. Some of these aspects are in the area of contents modelling, whereas others are in the area of user modelling.

4.2. Kernel model

In this section we discuss a kernel model, to be used as a base for information modelling in the context of the Bible. We adopt the graphical convention, where a circle represents an object type and a box represents the role played by an object type in a fact type.

In [7] we have proposed a possible kernel model, called *logbook*. In our current paper we use a more concise model. We have object types *person*, *action*, and *location*. Using these object types, we can introduce relations between persons and their actions and locations. The model is shown in Figure 4.

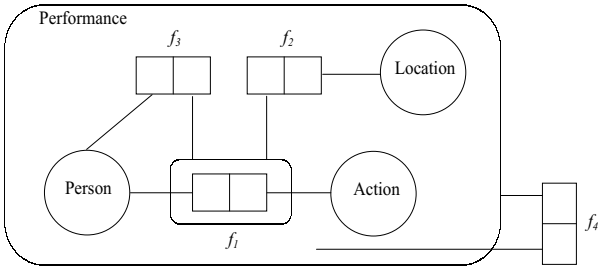

Figure 4 Graphical representation of kernel model

Object types are connected via fact types, expressing that objects of a given type can *play roles* in facts of a given type. As an example, fact type f_1 expresses that persons can perform actions, and vice versa, that actions can be performed by persons. In the model in Figure 4, we have the following basic fact types:

f_1	Person performs Action
f_2	Fact type f_1 occurs at Location
f_3	Fact type f_1 concerns Person

Fact type f_1 expresses that persons can perform actions, or, more precisely, objects of type *person* may perform objects of type *action*. An example fact *x* of type f_1 could be *Jesus said: follow me*. In this paper, we prefer a liberal verbalization of facts, rather than a more rigid verbalization such as *Person 'Jesus' performs Action 'saying: follow me'*.

Next we consider fact type f_2. This fact type expresses that facts of type f_1 may occur at a specific location. In other words, fact type f_2 allows us to record the locations where actions are performed by persons. An example fact of type f_2 could be *fact x occurs at Galilee*. Finally, we consider fact type f_3. This fact type expresses that facts of type f_1 may concern specific persons. An example fact of type f_3 could be *fact x is said to Peter*.

Note that the model in Figure 4 also contains fact type f_4. This fact type deals with the composition of f_1, f_2 and f_3 and will be discussed in section 4.4. We will now first deal with fact instances.

4.3. Instantiation of kernel model

In this section we consider fact instances. These instances are based on the Authorized King James version ([8]). We use a subscript notation for naming our fact instances. As an example, fact $f_{1.8}$ is an instance of type f_1. Although we use numbers here, instances are given in arbitrary order. In cases where explicit ordering is relevant, we use navigations mechanisms. These have been treated in section 2.

For the purpose of convenience, we mention the instances of a single fact type together, rather than switching between fact types. This however is not a strict requirement. We first consider instances of fact type f_1, expressing *Person performs Action*:

$f_{1.1}$	(Mt 28:18)	Jesus said: all power is given unto me.
$f_{1.2}$	(Mt 18:21)	Peter said: how oft shall my brother sin, and I forgive him?
$f_{1.3}$	(Mt 19:27)	Peter said: we have forsaken all, and followed thee.
$f_{1.4}$	(Acts 26:23)	Christ should suffer.
$f_{1.5}$	(Acts 26:23)	Christ should rise from the dead.
$f_{1.6}$	(Acts 26:23)	Christ should shew light unto the people.
$f_{1.7}$	(Acts 26:22)	I say none other things than the prophets and Moses did.
$f_{1.8}$	(Rom 1:3)	The gospel concerns Jesus Christ our Lord.
$f_{1.9}$	(Rom 1:4)	The gospel declared the resurrection from the dead.
$f_{1.10}$	(Rom 1:2)	God had promised the gospel by his prophets in the scriptures.

$f_{1.11}$ (Eph 2:20) You are built upon the foundation of the apostles and prophets.

$f_{1.12}$ (1Cor 15:3) Christ died for our sins according to the scriptures.

Note that the above fact instances contain different verbs, each of which may be in the present, past, future, etc tense. In cases where each verb is required to correspond with a separate fact type, a further refinement of the kernel model can be made. Such refinement will of course also increase the influence of the Bible translation used for building the instances.

Next we consider facts, in which facts of type f_1 are nested. These facts are instances of type f_2, expressing *Fact type f_1 occurs at Location*. Some instances are:

$f_{2.1}$ (Mt 28:16) Fact $f_{1.1}$ happened in Galilee.

$f_{2.2}$ (Mt 19:1) Fact $f_{1.3}$ happened in the region of Judea beyond Jordan.

Note that fact type f_2 allows us to define locations from different perspectives. As an example, consider the following instances:

$f_{2.3}$ Fact $f_{1.4}$ happened at Location 'A'.

$f_{2.4}$ Fact $f_{1.4}$ is said at Location 'B'.

Here, fact $f_{2.3}$ indicates the location where Christ suffered, while fact $f_{2.4}$ indicates the location where the statement 'Christ should suffer' was made. When aiming at *conciseness* and *flexibility*, these different perspectives in information modelling are desirable. However, in cases where *accuracy* of modelling is required, a more complex (refined) model must categorize facts $f_{2.3}$ and $f_{2.4}$ into separate fact types.

Next we consider other facts, in which facts of type f_1 are nested. These facts are instances of type f_3, expressing *Fact type f_1 concerns Person*. Some instances are:

$f_{3.1}$ (Mt 28:16) Fact $f_{1.1}$ was said to the eleven disciples.

$f_{3.2}$ (Mt 18:21) Fact $f_{1.2}$ was said to Jesus.

$f_{3.3}$ (Mt 19:27) Fact $f_{1.3}$ was said to Jesus.

$f_{3.4}$ (Acts 26:19) Fact $f_{1.4}$ was said to King Agrippa.

$f_{3.5}$ (Acts 26:19) Fact $f_{1.5}$ was said to King Agrippa.

$f_{3.6}$ (Acts 26:19) Fact $f_{1.6}$ was said to King Agrippa.

$f_{3.7}$ (Acts 26:19) Fact $f_{1.7}$ was said to King Agrippa.

Again we see the notion of conciseness and flexibility here. As a consequence, in the instantiation of *Fact type f_1 concerns Person*, a variety of instances of *concerns* may be chosen. For example, it can be *fact $f_{1.x}$ concerns person y*, or it can be: was said to, was said by, was said about, was promised to, was promised by, caused the death of, etc.

4.4. Composition of fact types

In section 4.2 we considered fact type f_2 and f_3 containing fact type f_1. In section 4.3 we considered instances of such nested facts. This principle is called *objectification* of fact type f_1 as this fact type now behaves as an object type playing roles in other fact types. For more background information on objectification, see e.g. [1] or [10]. In case a single fact type contains multiple fact types, we apply *composition of fact types*. Such a composition of fact types is also called a schema type (see e.g. [5]).

In our kernel model, the composition of fact types f_1, f_2, and f_3 is called Performance. Using this composition, we introduce the following fact type:

f_4 Fact type f_1 concerns Performance

Next we consider instances of fact type f_4. We examine relations between instances of fact type f_1. Some examples are the following:

$f_{4.1}$ Fact $f_{1.7}$ concerns facts $f_{1.4}$, $f_{1.5}$, and $f_{1.6}$.

$f_{4.2}$ Fact $f_{1.10}$ concerns facts $f_{1.8}$ and $f_{1.9}$.

$f_{4.3}$ Fact $f_{1.12}$ concerns facts $f_{1.7}$, $f_{1.10}$, and $f_{1.11}$.

These instances may be understood as follows. Fact $f_{4.1}$ expresses that the statement in Acts 26:22 about the prophets and Moses, concerns the statements in Acts 26:23 that Christ should suffer, rise from the dead, and proclaim light. Fact $f_{4.2}$ expresses that the statement in Rom 1:2 about the promise of the gospel by the prophets, concerns the statements in Rom 1:3-4 that Jesus Christ is our Lord and will rise from the dead. Finally, fact $f_{4.3}$ expresses that the statement in 1Cor 15:3 that Christ died for our sins according to the scriptures, concerns the statement in Acts 26:22 about the prophets and Moses, the statement in Rom 1:2 about the promise of the gospel by the prophets, and the statement in Eph 2:20 about the foundation of the apostles and prophets.

Note that the above instances only partially reflect the expressive power of fact type f_4. As an example, in facts $f_{4.1}$, $f_{4.2}$, and $f_{4.3}$, the instances of *performance* are restricted to instances of fact type f_1. According to the definition of *performance*, facts of type f_4 may also contain instances of fact types f_2 and f_3.

5. Modelling Bible readers

In this section we consider models of Bible readers. Here we enter the area of *user modelling*. This research area has already received a lot of attention and deals with major challenges for future usage of manual as well as computerized information (see e.g. [6], [7], and [8]). For us, the main aim of reader modelling is to allow for the composition of *personalized versions* of the Bible.

In our project, we propose a multi-dimensional levelled approach, where dimensions such as knowledge, belief, and experience are defined. These dimensions are dealt with from a static perspective (i.e. their values) as well as from a dynamic perspective (i.e. changing values). A typical example of the dynamic perspective is *growing belief*.

5.1. Knowing, believing, and experiencing

A data item may be known or unknown to a reader. A first enhancement is to recognize levels here. Then a reader may know this data item on a certain level, where levels are defined linearly or as a more complex (e.g. partial) order.

A linear knowledge dimension may be a scale between a given minimum and maximum. A simple example model would be to have a scale from 1 to 3, where 1 is interpreted as *I don't know this*, and 2 means *I know a little bit about this*, and 3 is *I know this*. A more complex example in which growing knowledge is reflected, could be as follows:

1. I don't know this.
2. I don't know this, but I would like to know it.
3. I know this a little bit.
4. I know this a little bit, and I would like to know more about it.
5. I know this.
6. I know this, and I would like to find more text about this topic.

In the case of Bible information, we need further enhancements. A reader may know a given data item on a certain level, while he does not believe in it at all. This is modelled by adding another dimension, called *belief*. Now the situation where this reader does not believe, is expressed by the lowest level of belief. This leads us to a multi-dimensional information model, containing level specifications for e.g. knowledge, belief, and experience. Clearly these different dimensions can be realized in parallel: a specific reader may know a data item on a high level, while he believes it on a lower level and experiences it on the lowest level.

Recall that such Bible reader models do not particularly aim at computerized Bible access, although this automated access will benefit a lot from the modelling techniques we use. Manual Bible access (i.e. normal reading) will also benefit, including the application in courses.

5.2. Multiple dimensions versus mixed dimension

In the multi-dimensional model, several properties of the reader are recorded. As an example, when a person A knows a lot about Jesus, we say that he has a high level of knowledge (e.g. 80%). At the same time, this person may have a relatively low level of belief, say 10%. In other words, person A knows a lot, but believes a little.

In quite a different situation, we have a person B with a relatively low knowledge level, say 40%. Suppose this person believes everything he knows (belief level 100%). Then person A knows more about Jesus, while person B believes more.

It is also possible to have a *mixed form of dimension*. Then we do not have multiple separate dimensions for knowledge, belief, and experience. Rather, these three dimensions are mapped into an (integrated) single one. This may for example be done as follows:

1. I don't know this.
2. I don't know this, but I would like to know it.
3. I know this, but I don't believe it.
4. I know this, I don't believe it, but I would like to believe it.
5. I know this and I believe it, but I don't experience it.
6. I know this and I believe it, I don't experience it, but I would like to experience it.
7. I know this, I believe it, and I experience it.

Note that the above mixed dimension also contains the aspect of growth explicitly. In a simpler version, this aspect can be ignored by omitting all parts containing *I would like to*.

5.3. Basic reader model

Let I be the set of data items which may be presented to a reader. Furthermore, let K, B, E define knowledge levels, belief levels, and experience levels, respectively. Then, our basic reader model consists of the following functions:

$k : I \rightarrow K$ where $k(i)=x$ is read as: the reader knows item i at level $x \in K$

$b : I \rightarrow B$ where $b(i)=y$ is read as: the reader believes item i at level $y \in B$

$e : I \rightarrow E$ where $e(i)=z$ is read as: the reader experiences item i at level $z \in E$

In cases where multiple readers are to be considered, these functions are further parameterised with readers. As an example, the call $k(r,i)$ then yields the knowledge level of reader r with respect to item i.

This leads to our underlying data model for readers. In this model, the scores of different readers in different dimensions are recorded. The model is given in Figure 5.

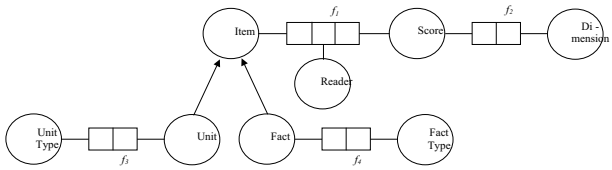

Figure 5 Underlying data model for readers

In Figure 5, we see that f_1 and f_2 are used to record items, readers, scores, and dimensions. In *dimension*, we have knowledge, belief, and experience. Note that the set of dimensions may be larger in cases where a higher granularity of reader modelling is desirable. As an example, additional dimensions may deal with the questions whether a reader *accepts* an item and / or *applies* it.

Next we address the question what data items have to be considered in Bible reader modelling. In Figure 5, a data item is a unit or a fact. Units come from the navigation model, as it was presented in section 2.

In f_3 the connection between units and their types is recorded. This connection may also be expressed as follows:

$$\text{Unit} = \cup_{x \in U} \text{Inst}_U(x)$$

Facts come from the kernel model as it was presented in section 4. Here we have the union of all facts of any type. This definition is analogous with the definition of units given above. Note that in Figure 5 units and facts are *subtypes* of items, which is graphically represented as usual with arrows (see e.g. [10], [1]). We use the concept of specialization, although generalization is also possible. More details on specialization and generalization may be found in [14] and [12]. Background information in this area is found in [9].

5.4. Dealing with dimensions from the dynamic perspective

When we deal with the dynamic perspective in reader modelling, we consider the way in which the reader properties change over time. This enables us to describe structuring mechanisms, for example in contexts where growing belief is relevant.

Consider a person having certain levels of knowledge and belief. Clearly, there may be known facts outside the scope of belief of this person. Suppose this person knows facts $X \subseteq$ Fact, but only believes the facts X-Y. So, the facts in $Y \subseteq X$ are knowledge but not belief. When a Bible (course) is presented to this person, it will present the facts in X in such a way that those facts from Y that are estimated to be closest to the current belief level are highlighted. These nearest facts may be candidates for a growth in belief. Note that such support is always based on estimations, and can never be applied in a prescriptive manner. So, flexibility is required here.

6. Conclusions

In this paper we considered modelling of Bible information. Besides structure modelling, we considered navigation models and contents models. In order to allow for the definition of personalized versions of specific Bible parts, we addressed the topic of reader modelling.

The models discussed in this paper, can be applied as structuring mechanisms. Our focus is on the basic definition of these structuring mechanisms. When a specific manual or computerized Bible application is to be implemented, attention has to be given to the structuring *process*. In that process, concrete Bible parts are mapped into our structuring mechanisms. Here, it can be helpful to take the results of other Bible projects as a starting-point. For example, so-called text hierarchies can be used as input to the mapping process. Then the actual derivation of text hierarchies is a preprocessing to the further structuring and presentation of concrete Bible parts. Experiences with the derivation of text hierarchies has been reported in e.g. [4] and [5].

Since so-called belief functions (e.g. Dempster-Shafer) have been successfully applied to various problems in systems engineering (see e.g. [1]), we may expect that these functions are applicable in Bible information modelling as well. However, the interpretation of belief functions in terms of biblical belief is not trivial at all. We are planning to examine how this could be embedded in our approach to Bible reader modelling.

We have noted that the use of different kinds of cross-reference mechanisms can be very helpful in models of complex documents. This will result in a refinement of our navigation model from section 2 and 3. Here, a distinction can be made between syntactic and semantic links, and between local and global links (see e.g. [2], [6].

References

van Bommel, P. ter Hofstede, A.H.M. & van der Weide, Th.P. (1991). Semantics and Verification of Object-Role Models, *Information Systems* 16(5) 471-495.

Bible Concordance.(1998). USA : Thomas Nelson Inc.

Gabel, J.B, Wheeler, C.B & York, A.D(2000). The Bible as Literature. London: Oxford University Press.

Talstra, E (1997). A Hierarchy of Clauses in Biblical Hebrew Narrative. In: E.J. van Wolde (ed). *Biblical Interpretation Series*, 29. 85-118. Leiden: Brill.

Talstra, E (1997). Workshop: Clause Types, Textual Hierarchy, Translation in Exodus 19, 20 and 24. In: *Biblical Interpretation Series.* E.J. van Wolde (ed). 29. 119-132, Leiden: Brill.

van Bommel, P. (ed)(2003). Information Modeling for Internet Applications. USA: Idea Group Publications.

van Bommel, P, Frederiks, P.J.M & Th.P. van der Weide(1996). Object-Oriented Modeling based on Logbooks, *The Computer Journal* 39(9) 793-799.

(2001)The Holy Bible. Authorized King James Version. USA. Thomas Nelson Inc.

(1990) The Holy Bible. New King James Version. USA. Thomas Nelson Inc.

Halpin, T (2001). Information Modeling and Relational Databases. USA: Morgan Kaufmann Publishers.

Chen, P (1976). The Entity-Relationship Model: Toward a Unified View of Data, *ACM Transactions on Database Systems* 1(1) 9-36.

ter Hofstede, A.H.M., Proper, H.A & van der Weide, Th.P (1993) Formal Definition of a Conceptual Language for the Description and Manipulation of Information Models, *Information Systems.* 18(7).

Buneman, P, Fernandez, M & Suciu, D (2000). UnQL: a Query Language and Algebra for Semistructured Data based on Structural Recursion, *VLDB Journal* 9(1) 76-110.

ter Hofstede, A.H.M. & van der Weide. Th.P(1993). Expressiveness in Conceptual Data Modelling, *Data & Knowledge Engineering* 10(1) 6 5-100.

Fink, J, Kobsa, A.(2002) User Modeling for Personalized City Tours, *Artificial Intelligence Review* 18(1) 33-74.

Fink, F.(1999) Transactional Consistency in User Modeling Systems. *Proceedings of the Seventh International Conference on User Modeling.* Springer-Verlag.

Bull. S, & McCalla, G (2002). Modelling Cognitive Style in a Peer Help Network, *Instructional Science* 30(6):497-528.

ter Hofstede, A.H.M. & van der Weide. Th.P(1992). Formalisation of Techniques: Chopping down the Methodology Jungle, *Information and Software Technology* 34(1) 57-65.

Ryan, M.J (2002). Violations of Belief Persistence in Dempster-Shafer Equilibrium, *Games and Economic Behavior* 39(1) 167-174.

Calado, (P) et al (2003). Local versus Global Link Information in the Web, *ACM Transactions on Information Systems.* 21(1) 42-63..