Personalizing the Appearance of Content Packages

Alexander Kröner and Hideo Sato
German Research Center for Artificial Intelligence GmbH
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
kroener@dfki.de

Abstract

The personalized delivery of content is a way of making online content portals more attractive to end users, and consequently, nowadays many online portals provide varying ways of personalizing their service. In order to facilitate the creation of such personalized services, the IMAGEN toolset came into existence. IMAGEN aims at creating personalized content packages, where documents are selected from a repository with respect to the customer’s interests, and are bundled to a package. The appearance of packaged documents may be revised in order to take the layout preferences of authors, service providers, and customers into account. In this contribution, we describe varying ways of personalizing the layout style of content packages, ranging from a plain exchange of styles as suggested by the content authors to a constraint-based style unification. In the sequel we describe how service providers as well as customers may take influence on that revision process in order to obtain personalized layout styles.

1 Introduction

A key factor for the success of online portals is the presentation of the provided content. Ideally the presentation reflects the portal user's information needs and interests as well as the capabilities of the user's display device. Hence the operators of online portals ask for tools that facilitate setting up such personalized information services. Key requirements to such tools include the capability of using content that is already available and the application of existing standards. Thus selecting, packaging, and reformatting of existing media assets for the purpose of reuse are becoming primary tasks.

Within the EU funded project IMAGEN (Intelligent Multimedia Application Generator, IST-1999-13123) a platform has been developed for the set-up and operation of portal services that aim at customized publication and distribution of multimedia content.[1] Thus IMAGEN can be conceived as a mediator between content authors (1st-level users), publishers, content syndicators/owners, and administrators (2nd-level users), and content customers (3rd-level users).

The first pilot application of IMAGEN is a free entertainment/promotional application called MyArt. Its major business objective is to establish a communication channel between a large Italian publisher in the art domain and the community of English speaking internet users interested in Renaissance arts. Basic idea of this service is to provide 3rd-level users with magazine-like IMS[2] content packages, which are compiled on the fly from so-called *content units*. These are documents consisting of content related to Renaissance arts and some links related to a commercial Web service offered by the publisher. Content units are created by 1st-level users using an authoring tool such as the eXact Packager,[3] and are stored in a repository. When checked into the repository, content units are automatically classified based on their textual content.

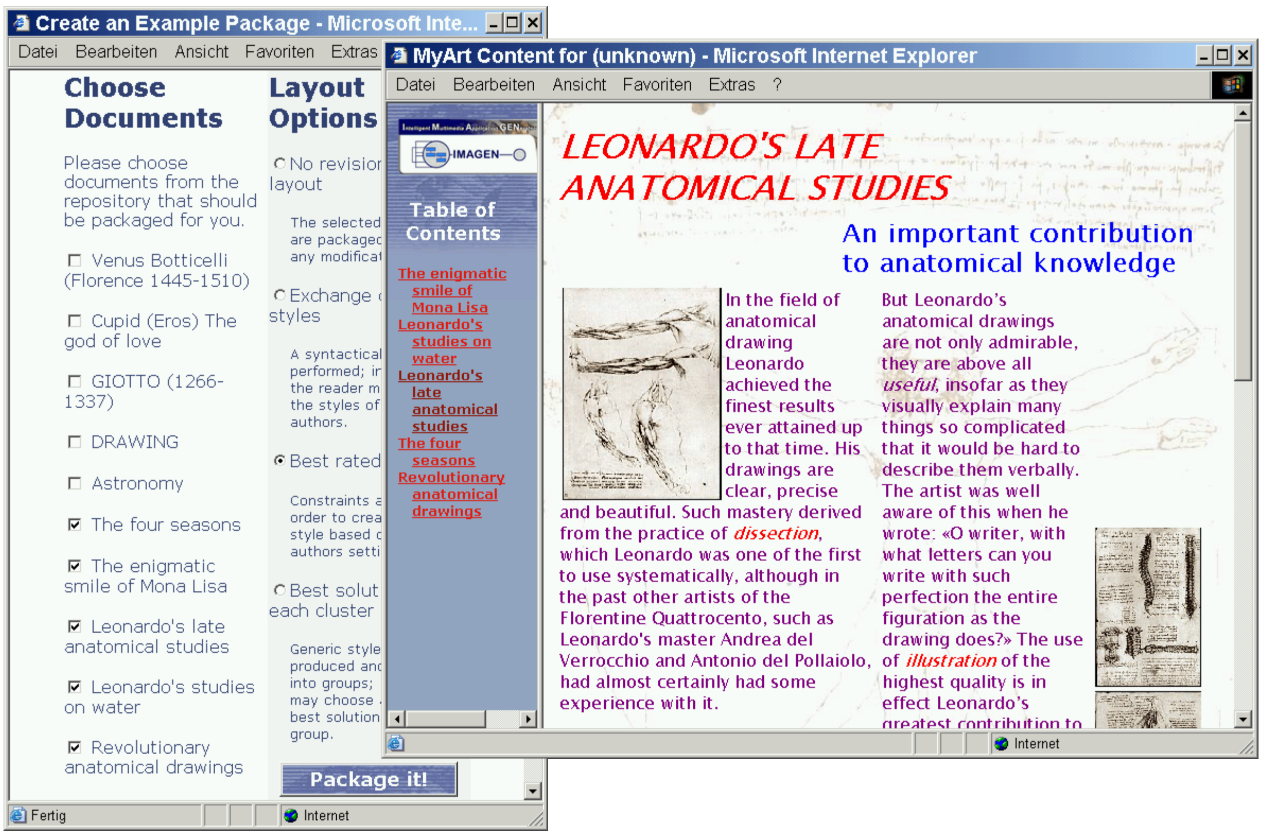
The workflow of the online service is divided into three major steps: selection and packaging of content units, revision of the content units’ appearance, and delivery. In response to a 3rd-level user request, the so-called *content manager* selects content units from the repository according to the selection method chosen by the 3rd-level user. In the case of *Implicit Profiling* (see [CK02]), content is chosen fully automatically by matching the user’s interest profile to content from the repository. After delivery, the interest profile is updated with respect to the content units from the package which have been actually accessed. An alternative approach for selecting content is *Explicit Profiling* (see [BV03]). There the user is asked to specify explicitly the performance of the desired content with respect to classification dimensions, e.g., *joyous* vs. *melancholic*. In addition, the user may choose if the selection should take into account the opinions of other 3rd-level users. In that case, the system applies *collaborative filtering* (cf. [SM95]) in order to refine the search. After delivery, the user may then express her own opinion about the delivered content using a feedback form. The input received this way is then used by the system to adjust the collaborative filtering mechanism.

When the content selection is finished, the layout manager performs optionally a revision of the package’s appearance, and creates HTML access structures based on the package’s organization to enable package browsing without need for a specific IMS browser. Afterwards the package is provided to the 3rd-level user for direct access or download. This process is under control of the *transaction manager*, which provides

[1] *http://www.imagenweb.org/*

[2] *http://www.imsproject.org/*

[3] *http://www.learnexact.com/*


Create an Example Package - Microsoft Inte...
Datei Bearbeiten Ansicht Favoriten Extras
Fertig
Internet
MyArt Content for (unknown) - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
IMAGEN
Internet

Choose Documents

Please choose documents from the repository that should be packaged for you.

☐ Venus Botticelli (Florence 1445-1510)

☐ Cupid (Eros) The god of love

☐ GIOTTO (1266-1337)

☐ DRAWING

☐ Astronomy

[✓ The four seasons]

[✓ The enigmatic smile of Mona Lisa]

[✓ Leonardo's late anatomical studies]

[✓ Leonardo's studies on water]

[✓ Revolutionary anatomical drawings]

Layout Options

○ No revision layout

The selected are packaged any modificat

○ Exchange styles

A syntactical performed; ir the reader m the styles of authors.

[✓ Best rated]

Constraints a order to crea style based c authors setti

○ Best solut each cluster

Generic style produced anc into groups; may choose best solution group.

Package it!

Table of Contents

The enigmatic smile of Mona Lisa
Leonardo's studies on water
Leonardo's late anatomical studies
The four seasons
Revolutionary anatomical drawings

LEONARDO'S LATE ANATOMICAL STUDIES

An important contribution to anatomical knowledge

In the field of anatomical drawing Leonardo achieved the finest results ever attained up to that time. His drawings are clear, precise and beautiful. Such mastery derived from the practice of *dissection*, which Leonardo was one of the first to use systematically, although in the past other artists of the Florentine Quattrocento, such as Leonardo's master Andrea del Verrocchio and Antonio del Pollaiolo, had almost certainly had some experience with it.

But Leonardo's anatomical drawings are not only admirable, they are above all *useful*, insofar as they visually explain many things so complicated that it would be hard to describe them verbally. The artist was well aware of this when he wrote: «O writer, with what letters can you write with such perfection the entire figuration as the drawing does?» The use of *illustration* of the highest quality is in effect Leonardo's greatest contribution to

Figure 1: Screenshots of the IMAGEN layout manager showcase. The screen on the left-hand side enables manual choice of documents and the layout revision mode. After processing and HTML wrapping, the 3rd-level user may browse the created package as shown on the right-hand side.

typical features of digital rights management, such as water marking and transaction tracking (see also [BKZ98]).

In the following we focus on the layout revision process performed by the layout manager. First, we describe in Section 2 the idea of the revision process, and the varying revision methods provided by the layout manager. In Section 3 we continue with an description of the underlying constraint approach, followed by a description of how the constraint solving process may be customized. We begin in Section 3.1 with a brief description of how 2nd-level users may influence the revision outcome by adjusting fine-grained preferential constraints. Since the approach applied there might exceed the knowledge of 3rd-level users about layout, we continue in Section 3.2 with an approach, where constraints are aggregated to groups, which may be customized by a simple Web interface in order to adapt the package's appearance to the 3rd-level users current preferences.

2 Revising Layout Styles

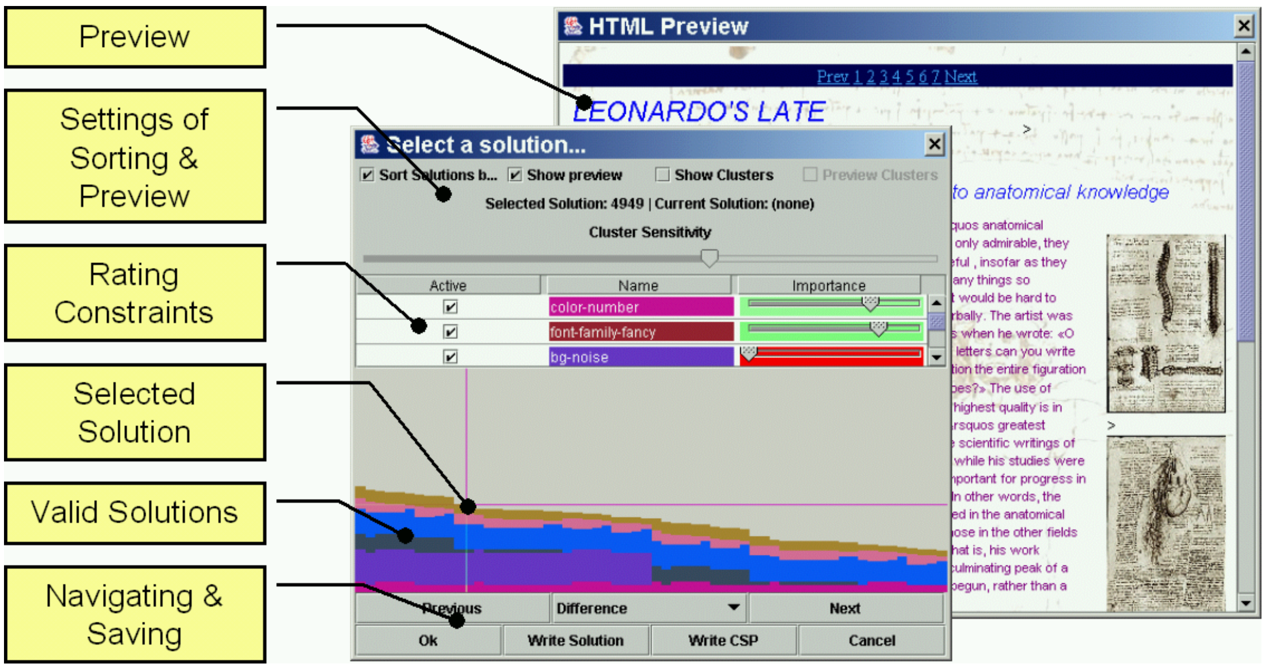
The content repository may contain content units from many different authors – with potentially widely different ideas of an appealing style. Thus the appearance of the single documents selected by the content manager for a content package may vary and make the package appear heterogeneous. For more convenient reading, 3rd-level users might therefore be interested in a unification of the document styles. Such a unification should also take into account the ideas of the contributing 1st-level users as well as the preferences of the 2nd-level user (e.g., to emphasize the corporate identity).

In IMAGEN, this task is performed by a set of modules forming the *layout manager*. A showcase of the layout manager has been made available at *http://www2.dfki.de:8080/IMAGEN/*. In contrast to the actual MyArt service, this showcase enables a manual selection of content units from a subset of MyArt's content repository. The manual selection enables controlled experiments, and facilitates exploring the varying ways of style computations provided by the software.

The revision of layout styles is a popular issue due to its importance for personalization and accessibility. It is addressed in manifold ways not only in research, but also by existing applications. For instance, Yahoo![4] enables the user to choose among predefined styles. This technique is easy to handle for the end user, but limits her to the settings defined by the service provider. Other services enable the user to define styles manually and then to apply these to documents stored in some repository (e.g., as proposed by Visual Friendly[5]). Here the end user may set up her very own style – but neither the service provider's nor content authors' preferences will be taken into account.

In order to let all user groups participate in the style selection process, the layout manager provides several different layout revision modes. The left-hand side of Figure 1 shows the entrance screen of our showcase,

[4] *http://www.yahoo.com/*
[5] *http://www.visualfriendly.com/*



Selected Solution: 4949 | Current Solution: (none)

Active	Name	Importance
✓	color-number	
✓	font-family-fancy	
✓	bg-noise	

Figure 2: The user interface of the Visual Solution Space Explorer, a tool for configuring rating constraints

where the user may choose documents for packaging, and one of the included layout revision modes.

"Exchange of author styles" lets the layout manager organize the styles of the packaged documents in a way, which enables the 3rd-level user to choose (even offline) among these styles the one she likes most while browsing the package. The chosen style is applied to all documents contained in the package.[6] That process leaves the 1st-level users' styles untouched, but is performed without verifying if the chosen style is actually appropriate for the document currently displayed – and may therefore produce unpleasant results.

"Best rated solution" performs a constraint-based unification of the packaged documents' styles. This process is described in more detail in the next section. In brief, the system tries to generate a new style based on the preferences of 1st- and 2nd-level users, and general style requirements, e.g., about readability. Since this process may produce several solutions, the system computes a rating for each solution and chooses the one with the best rating.

"Best solution of each cluster" is built upon the constraint-based unification. The method identifies clusters of similar solutions, chooses the best solution of each cluster, and stores the styles received this way similarly to the mode "Exchange of author styles" so that the 3rd-level user may perform the final decision about the best style.

"Manually" enables the manual exploration of the unification results. This feature is provided only to 2nd-level users, and is provided in the first place in order to assist these users in the task of administrating the layout constraints applied during the unification process.

[6] The style exchange is achieved by using a JavaScript style switcher as proposed by Paul Sowden, see *http://alistapart.com/articles/alternate/*.

3 Constraining Layout Styles of Content Packages

As mentioned, the layout manager applies style constraints in order to obtain a unified style for the documents of a package. Constraints are a popular technique for solving layout problems (cf. [Graf97] and [Jacobs et al. 03]), and this holds also for their application in the Web context. For instance, Alan Borning's group provided several proposals about how constraints could extend cascading style sheets (CSS) as applied in Web page design (see [BLM00]). Here the motivation is to make a Web page's style adapt to constraints imposed from the display environment (e.g., window size). Similarly, in [Krö01] not only the style, but also the content is adapted in part using constraints and templates in order to optimize a Web page's appearance for a given environment.

In IMAGEN, a finite domain constraint solver enables the processing of binary constraints about attribute values of XML document nodes. This constraint solver is applied by the layout manager to constrain the style attributes of content units included in a package. To each attribute a variable domain is assigned, which is initialized with the values suggested by the 1st-level users who contributed to the package. The variables constructed this way are constrained according to a constraint set specified by the 2nd-level user. Thus the authors' style preferences serve as input for a layout revision under control of the service provider.

The constraint set applied in that process consists of two different classes of constraints, which reflect different stages of the constraint solving process. At the beginning, this process is under control of the so-called *required constraints* which express layout preferences that are essential for an appropriate layout. For instance, required constraints encode conditions such as "*The title font size should be larger than the size of* *regular text*" or "*Background and foreground color should not be equal*".

Styles satisfying the required constraints count as *valid solutions*. The primary goal of the constraint solving is to find at least one valid solution – but this process may fail. In order to reduce the impact of such a failure, the solver analyses in advance dependencies in the given constraint problem. For instance, there might be no constraint connecting the variables of foreground color and font family. Then the solution of the constraints on the color variable won't affect the solution the font family. Consequently, the solver creates two separated constraint problems, which are solved on their own. If the solution of one of these sub problems fails, then the solution of the other problem is not affected, and thus the overall problem may be solved at least in part. Variables of unsolved sub problems stay unchanged, which means that in the worst case where all sub problems have failed, the styles will stay as originally defined by the 1st-level users.

However, in general we expect to receive not only a single one, but a set of valid solutions, of which only the *best* ones should be presented to the 3rd-level user. Thus we have to assign a measure of quality to the valid solutions, which reflects the performance of valid solution with respect to soft layout preferences such as "*Prefer less noisy background images*" and "*Maximize the font size of regular text*".

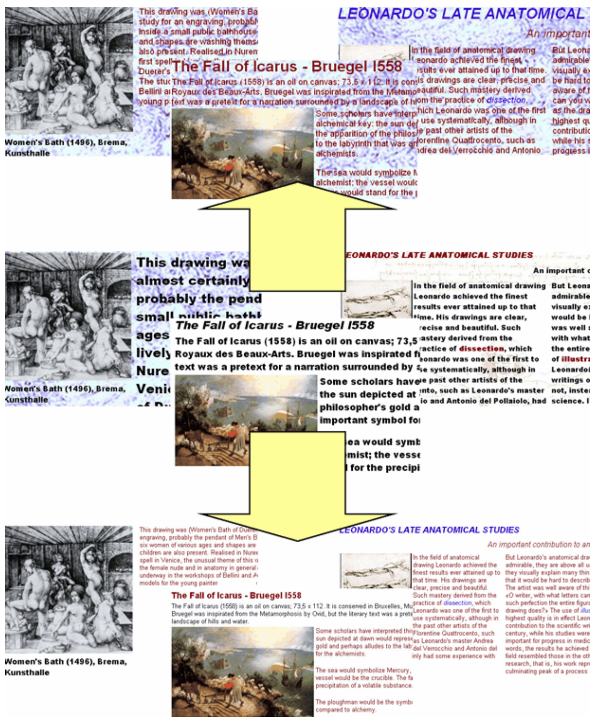
This rating process is performed by *rating constraints*. A rating constraint assigns a rating value to each valid solution, which is calculated by a solution-independent rating operator from the constraint variable's value in the rated solution, and is modified by a weight. The latter one ranges from "–1" to "1", where "–1" means "to be highly avoided", and "1" means "to be highly preferred". The overall rating of a solution is sum of the rating results of all rating constraints defined in the given constraint problem. The system tries to optimize the outcome of the rating process, but there is no threshold where a rating constraint would count as "failed".

Required as well as rating constraints are represented using an XML binding, which is described in more detail in [KBR02]. The constraint solver itself was implemented using Java, XML, and XSLT. It consists of a kernel that relies on the Java Constraint Library (JCL, see *http://liawww.epfl.ch/JCL/*), a set of wrapper classes that provide independence from the actual solver implementation, and several extensions – in the first place, the rating constraint solver, and the XML representation of constraints.

3.1 Administrating Style Constraints

Which style is selected for delivery depends on the rating constraints' weights. Hence 2nd-level users have to express their preferences by assigning weights to rating constraints in order to instruct the layout manager to choose solutions meeting their ideas of an appropriate style.

The complexity of that configuration task depends on the number of applied rating constraints. For instance, in MyArt about ten rating constraints are applied in order to decide about the style that should be delivered. Since these constraints may compete with each other,



LEONARDO'S LATE ANATOMICAL
The Fall of Icarus - Bruegel I558
Women's Bath (1496), Brema, Kunsthalle
LEONARDO'S LATE ANATOMICAL STUDIES
The Fall of Icarus - Bruegel I558
Women's Bath (1496), Brema, Kunsthalle
LEONARDO'S LATE ANATOMICAL STUDIES
The Fall of Icarus - Bruegel I558
Women's Bath (1496), Brema, Kunsthalle

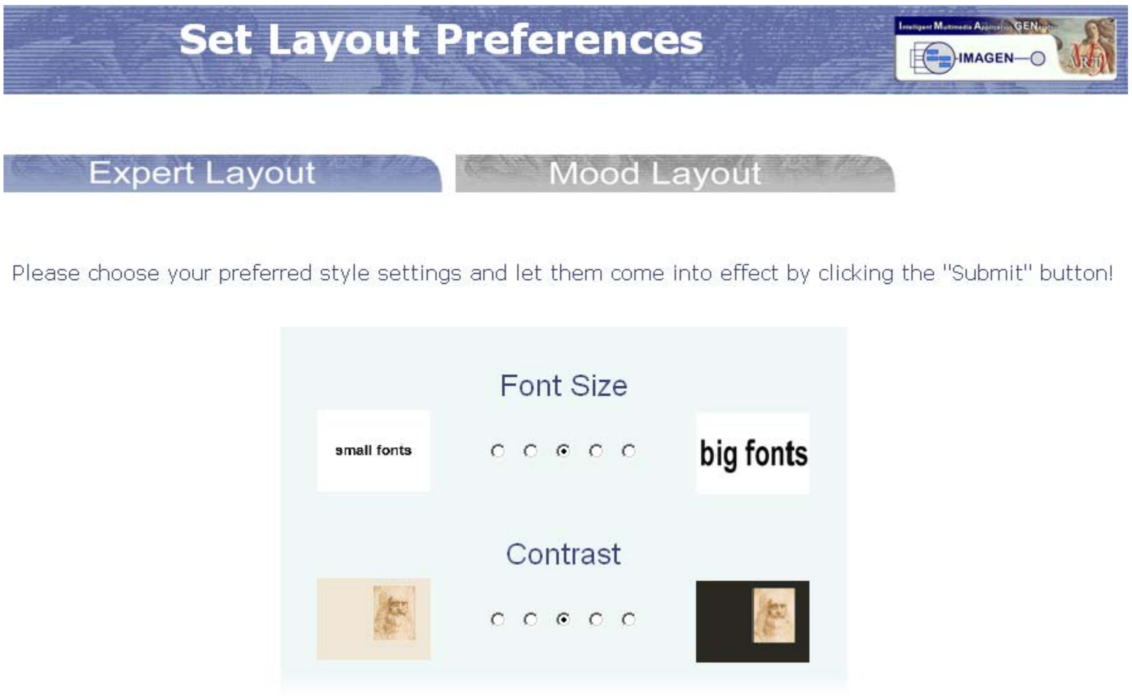
Figure 3: The impact of the cumulative constraint "loud": above maximized, in the middle unchanged, and below minimized.

finding the "right" combination of weights is sometimes a cumbersome task. Therefore we created the so-called Visual Solution Space Explorer (VSSE, see [KBR02]), which provides 2nd-level users with an overview of the valid solutions, the ratings assigned to these solutions, and the option to adjust the weights of the involved rating constraints by means of a graphical user interface.

Figure 2 illustrates the VSSE's user interface. Most important are the table filled with rating constraints and the bar chart. The table enables adjusting each rating constraint's weight by means of a slider. The resulting weights are visible in the bar chart: each bar represents a single valid solution, the bar's height represents the solution's overall rating, and the varying colors within a bar encode the contribution of the varying rating constraints to that rating.

3.2 Personalizing Style Constraints

While the VSSE provides an abstraction from the weight computation mechanism and provides immediate feedback to weight manipulations, it still requires a certain understanding of the adjusted rating constraints' impact on layout to achieve an appropriate weight configuration in a reasonable time. Especially difficult to handle are configuration tasks, where several constraints have to be adjusted at once in order to achieve the desired result. For instance, if one is interested in a colorful style in MyArt, one has to increase not only the rating of color numbers, but also the rating of the foreground and the background color saturation. Consequently knowledge about the constraint set is required which a 2nd-level user *might* have, but for sure not a 3rd-level user.



Please choose your preferred style settings and let them come into effect by clicking the "Submit" button!

Figure 4: Personalizing the appearance in the layout manager showcase: two revision modes – Expert and Mood can be reached via tabs. In the centre, the weight controls of cumulative constraints specified for the selected mode are shown together with example thumbnails.

3.2.1 Grouping Constraints

In order to provide 3rd-level users – and 2nd-level users as well – with a convenient means of specifying style preferences on the fly, a further step of abstraction from the underlying technology was required. Therefore we enabled grouping of related rating constraints to more complex, but also more abstract constraints, an approach similar to the *cumulative constraints* as discussed in [HG96].

Like rating constraints, these groups possess a weight indicating the importance of the layout aspect incorporated by the group. Thus, by carefully combing rating constraints to groups, constraint authors (typically 2nd-level expert users) may create constraints, which constitute a more abstract kind of rating constraints. This way, the need for understanding the individual rating constraints meaning is reduced to an intuitive handling of fewer cumulative constraints. For instance, in the previously mentioned example one could capture all constraints relevant for the colorfulness of a style in a single cumulative constraint, and thus let the 3rd-level user adjust the colorfulness by assigning a weight to that single constraint instead of performing that action again and again for each involved rating constraint.

These constraint groups extend IMAGEN's original constraint definition. Their structure is straightforward – each cumulative constraint consists of a set of references to regular rating constraints. A single rating constraint may be referenced by several cumulative constraints. Additionally the references carry a weight indicating the importance of the constraint *within* the group and a multiplier which indicates the extend modifications of the group weight have on that particular constraint. This structure is illustrated in the following example.

```
<cs:cumulative
  name="loud" descr="Loudness">
  <cs:affectedRC
     name="size-heading-maximize"
     weight="1" mult="1"/>
   <cs:affectedRC
     name="size-around"
     weight="-0.25" mult="-0.75"/>
  <cs:affectedRC
     name="bg-image-noise"
     weight="0" mult="1"/>
</cs:cumulative>
```

The constraint shown in this example rates the "loudness" of a style. It makes reference to rating constraints related to the font size (`size-heading-maximize`, `size-around`) in order to let larger fonts appear "more loud", and to `bg-image-noise`, a rating constraint which judges the pixel noise of the background texture.

The impact of this constraint on the style selection is shown in Figure 3. In the middle, the original documents are shown before the constraint-based unification has taken place. On top, the same documents are shown after a revision, which included a maximization of the constraint's weight. Thereupon a colorful and noisy background texture has been selected in combination with large, bold fonts. The results on the bottom have been achieved after minimizing the constraint's weight. A plain white background has been chosen, and a regular choice of font settings.

3.2.2 Revision Modes

Cumulative constraints may help 2nd-level users to provide 3rd-level users with an abstract means of customizing the layout style. However, there is still the question which of these constraints are the best ones for a given application. For instance, 3rd-level users with a precise idea of the desired style might prefer cumulative constraints addressing typical style characteristics such as font sizes, background colors, and so on. But there might be other users thinking in terms such as “warmth” and “silence” instead of numbers of colors and background noise. Since the cumulative constraints may map arbitrary style aspects to the underlying rating constraints, one could address the ideas of these users by additional cumulative constraints as well – but would end up with a larger and thus more difficult to control constraint set.

Thus we decided to enable the definition of so-called *revision modes*. Each of these modes is a group of cumulative constraints, which provides a kind of view on the overall set of rating constraints applied by the system. The constraint solver receives the settings of only one mode as additional input, and adjusts the rating constraint weights with respect to these settings.

For 2nd-level users, these modes provide a means of addressing the ideas of different groups of 3rd-level users by different views on the same underlying set of rating constraints.

3.2.3 Extending the User Interface

In order to let 3rd-level users benefit from cumulative constraints and revision modes, we extended the layout revision in our showcase with a new screen shown in Figure 4, which is evoked after the content selection. It enables the 3rd-level user to choose among the revision modes – here “Expert” and “Mood” – and to adjust the weights of rating constraints contained in the selected mode. Here the mode “Expert” provides constraints which are still close to the actual constraint variables, whereas the mode “Mood” provides a completely different and less technical view on the constraint set.

After completion of the layout revision, the user may browse (and download, if desired) the result. If the user is not confident with the result, she may return directly to the layout preferences screen and adjust her style preferences while keeping the content selection unchanged.

4 Conclusion and Outlook

In this contribution we presented work conducted in the project IMAGEN about personalizing layout styles of content packages and illustrated varying approaches ranging from manual exchange of styles to a constraint-based style computation. The latter revision method leads to a configuration problem where the user has to express her style preferences by assigning weights to so-called rating constraints. Going further, we showed how administrators can be assisted in that task, and finally, we showed how grouping of rating constraints may help to assist also end users in the task of specifying layout preferences without knowledge about the underlying constraint set.

At this point, potential next steps include the extension of the VSSE software with a means of visually creating groups of rating constraints in order to assist 2nd-level users in the definition of cumulative constraints. For 3rd-level users, a more sophisticated means of providing feedback about the received layout would be a valuable feature, whose outcome could be applied in turn by the system in order to provide the user with proposals about adjustments that might help to improve the individual result.

Acknowledgements

This work was conducted in the context of the EU-founded project IMAGEN (IST-1999-13123). We like to thank our project partners for their valuable support, and are especially grateful to Giunti-ILABS for providing us with the art content shown in our examples.

References

[BKZ98] S. Burgett, E. Koch, and J. Zhao: Copyright Labeling of Digitised Image Data. IEEE Communications Magazine, Vol. 36 (3), pp 94-100, 1998.

[BLM00] A. Borning, R. Lin, and K. Marriott: Constraint-Based Document Layout for the Web. In: Multimedia Systems 8.3, pp 177-189, 2000.

[BV03] J. de Bo and L. Vervenne: Benefits of explicit profiling in the IMAGEN project. Technical Report 03, STAR Lab, Brussels, 2003.

[CK02] M. Chalamish and S. Kraus: Learning Users Interests for Providing Relevant Information. In: Proceedings of the ABIS-Workshop 2002: Personalization for the mobile World, pp 59-66, Hannover, Germany, 2002.

[Graf97] W. H. Graf: Constraint-based graphical Layout of Multimodal Presentations. In M. T. Maybury and W. Wahlster (publ.), editors, Readings in Intelligent User Interfaces, Los Altos, CA, 1997.

[HG96] W. Hower and W. H. Graf. A bibliographical survey of constraint-based approaches to cad, graphics, layout, visualization, and related topics. Knowledge Based Systems, 9(7):449--464, December 1996.

[Jacobs et al. 03] C. E. Jacobs, W. Li, E. Schrier, D. Bargeron, and D. Salesin: Adaptive grid-based document layout. ACM Trans. Graph. 22(3): 838-847 (2003).

[KBR02] A. Kröner, P. Brandmeier, and T. Rist: Managing Layout Constraints in a Platform for Customized Multimedia Content Packaging. In: Proceedings of the Working Conference on Advanced Visual Interfaces AVI 2002 pp. 89-93, Trento, Italy, May 22-24, 2002.

[Krö01] A. Kröner: Adaptive Layout of Dynamic Web Pages. In: DISKI - Dissertationen zur künstlichen Intelligenz 248, infix (2001), ISBN 3-89838-248-6.

[SM95] U. Shardanand and P. Maes: Social information filtering: Algorithms for automating “word of mouth”. In: Proceedings of CHI'95 – Human Factors in Computing Systems, pp. 210-217, 1995.