alpha-Adaptive: Evolutionary Workflow Metadata in Distributed Document-Oriented Process Management

Christoph P. Neumann, Peter K. Schwab, Andreas M. Wahl, and Richard Lenz

Friedrich-Alexander University,
Erlangen-Nuremberg, Germany,
{christoph.neumann|richard.lenz}@cs.fau.de

Abstract. The α-Flow project enables process support in heterogeneous and inter-institutional scenarios in healthcare. α-Flow provides a distributed case file and represents workflow schemas as documents which are shared coequally to content documents. The activity progress and data flow is controlled by process-related metadata. A use case will motivate user-defined and demand-driven status attributes that are not known at design-time. α-Adaptive demonstrates how to apply the EAV data design approach and prototype-based programming concepts in order to provide an adaptive-evolutionary status attribute model for document-oriented processes.

Topics: Process-oriented system architectures in healthcare, facilitating knowledge-acquisition of healthcare processes, deferred systems design, case handling

1 Introduction & Objectives

Medical treatment of patients is increasingly evolving from a series of isolated episodes towards a continuous process, incorporating multiple organizationally independent institutions and different healthcare professions. One characteristic of this process is that both the order of treatment steps and the amount of involved parties are usually not known in advance as they are largely dependent on the preceding course of the treatment. Evolutionary workflow approaches are required that enable cooperation and coordination among the participants. It is essential to deal with the semantic and technical heterogeneity of the systems at the participating sites because different information systems and internal workflows are used.

In the case handling paradigm [1], the flow of a patient between healthcare professionals is considered as a workflow—with activities that include all kinds of diagnostic or therapeutic treatments. The workflow is considered as a case, and workflow management in healthcare is to handle these cases.

2 Background

Case handling is a new paradigm for process support. Unlike workflow management it is aimed at supporting a team of cooperating process participants in their decisions rather than predefining process steps. The core features that are defined by the case handling paradigm [1] are: (a) provide all information available, i.e. present the case as a whole rather than showing bits and pieces, (b) decide about activities on the basis of the information available rather than the activities already executed, (c) separate work distribution from authorization and allow for additional types of roles, not just the execute role, and (d) allow workers to view and add/modify data before or after the corresponding activities have been executed. Yet, on the framework level, contemporary case handling focuses on hospital (single institution) scenarios and technologically on a centralized case handling system.

α-Flow Conception: The α-Flow approach, as it is described in [2] and [3], aims to provide case handling in distributed environments and emphasizes on document-oriented systems integration. α-Flow is considered as an implementation of distributed document-oriented process management (dDPM). The document-oriented integration style supporting inter-institutional environments was motivated in a-Flow predecessor DEUS [4]. Basically, the traditional paper based interaction paradigm, that uses signed forms for communication, is imitated and extended to exploit the potential of electronic communication. The α-Doc is our notion of a distributed case file that contains all case related information to be shared among multiple participants.

An α-Doc is decomposed in α-Cards that are units of organizational accountability and of validation as well as subject to atomic synchronization actions. Each α-Doc represents an entire case which we also name an α-Episode. There is a one-to-one relation between α-Doc and α-Episode: The term α-Doc emphasizes on the artifact dimension, whereas the term α-Episode emphasizes on the implicit workflow dimension with tasks which are the treatment steps. Each task is planned by creating an α-Card descriptor, and it is fulfilled by providing its result report. The treatment process and its state will progress with the creation or change of α-Cards, which we elaborated in [5].

α-Flow Artifact Context of Adornments: For α-Adaptive, the focus lies on the structure of an α-Card that is outlined in Fig. 1. An *α-Card* consists of a descriptor and a payload. The *α-Card descriptor* consists of several *α-Adornments*. The general term "adornment" is borrowed from the Unified Modeling Language: an adornment adds to the meaning and/or semantics of the element to which it pertains and has a textual or graphical representation. In α-Flow, adornments are process-relevant status attributes and represent certain aspects of an α-Card's life-cycle and process state. Adornments either classify α-Cards passively or an adornment status change can actively act as an event trigger that implies process change.

The basic *α-Adornment model* for α-Cards has been discussed in [2] and consists basically of adornments for: contributor and object under consideration (OC), validity and visibility, version and variant, fundamental semantic payload type, syntactic payload type and domain-specific semantic payload type. The *payload* of an α-Card contains an arbitrary electronic medical document, contributed by a process participant.

One exemplary adornment usage is given from [2]: visibility and validity. An α-Card represents an open task if there is only the descriptor but no payload. It represents a fulfilled task if there is a payload with visibility set to "public" and validity set to "valid". α-Cards with a contributed payload but still with its visibility or validity adornments set to incipient states (e.g. "private" or "invalid") represent work in progress. To

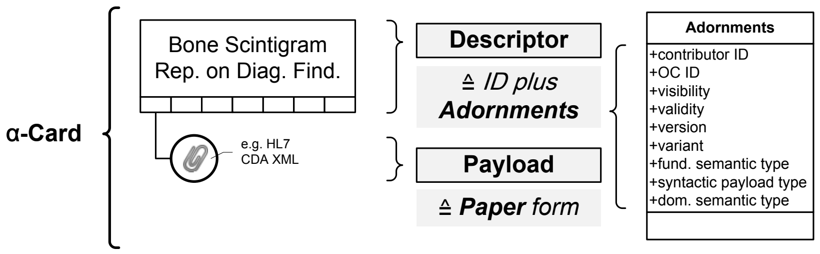


Fig. 1. Artifact structure: an α-Card with its descriptor & payload

share preliminary information, that is not yet validated by a human signature, is common in healthcare, especially for reports on diagnostic findings.

α-Flow Operative Outline: The "α" in all our terms implicates "active", in analogy to the underlying concept of active documents with active properties [6], and "α-Doc" essentially means "active document". The idea is to technically form the collective case dossier into a single self-managing file unit, that can be handled as passive files like a PDF or Word file, containing both the case data and the dDPM enactment engine. One appeal of such a solution is as follows: If we provide a technical platform for such eccentric artifacts as our active documents for dDPM purposes, each human actor becomes participant by handing him or her a copy of the α-Doc—which is basically the same interaction as making them participants by handing over referral vouchers.

From an operative embedding perspective, the α-Flow approach minimizes the initial work for establishing an information exchange between different process participants. From a technological perspective, no pre-installed system components are required to interact with an α-Doc. Thus, the α-Doc is an instantly available tool that needs no administration.

An α-Doc provides a functional fusion of a shared work list editor with instant messaging with version control with access restrictions. Furthermore, the α-Doc embeds an α-Props subsystem [7] which is a rule engine that guards the adornment changes and executes active properties as the kernel of the active document. Workflow benefits are process planning, process history, and participant management as well as template creation for process structure and process-required roles. An α-Doc supports all core features (a) to (d) of case handling. Further details about the α-Flow mechanics must be skipped at this point.

3 Motivation and Objectives

This paper focuses only on the α-Flow adornment model. It does not provide in-depth explanations for the over-arching artifact structures. It will not be necessary to know the overall α-Flow operative embedding in order to understand the α-Adaptive concerns. The first part of the paper provides a use case which is result of our studies and motivates user-defined adornments by example. The method section then outlines two general state of the art methods to achieve run-time adaptability in information systems. The last part of the paper discusses the application of our selected methods on our adornment model in order to achieve adaptive process adornments for healthcare artifacts.

The appeal of an adaptive attribute metadata model is that it allows for continuous adaptability of adornments as the process status attributes of artifacts. The general system architecture shall enable the users themselves to adapt adornments according to their demands at run-time. We need adornments, in addition to the payload documents, because we allow arbitrary payload file formats. The motive behind augmenting payloads with descriptors/adornments is to avoid upfront system integration efforts.

Status attributes for the artifacts are necessary such that actions can be defined upon their status change and automated by an active property. The users ultimately decide if the efforts to maintain a specific status attribute gains any benefit for cooperation. The use case scenario will motivate domain-specific status attributes whose exact specification cannot or should not be fixed at the design-time of a distributed process infrastructure because they ultimately are subject to semantic consensus finding between actors, institutions and domains.

4 Use Case Scenario

This section provides an example for user-defined status attributes and their utilization during treatment episodes. The use case description is independent of our framework—it is based on paper-based working practice in healthcare. This section extends our former description of breast cancer treatment [5].

Condition Indicator: The exemplary classifier *condition indicator* can be of use in situations where patients are under periodic medical examination. Consensus finding must happen outside our system (the process platform can only foster it by supporting ad-hoc definitions of adornments as well as changes to the value range by the actors at any time). For the sake of our example, the process participants already have a consensus and we assume that they agreed upon a value range of *normal*, *guarded*, and *serious* for the condition levels. Such status can be attributed to any report and indicates the patient condition at the corresponding time and in regard to the diagnostic context.

After the primary therapy [5], i.e. removal of the tumor, the post-operative care and the adjuvant therapy run in parallel for the first six months. The adjuvant therapy (with chemo therapy, radio therapy and hormonal therapy) is not described in this paper as aggravation is mainly discovered during post-operative care. Post-operative care will continue for about five years. In contrast to primary therapy, the treatments during post-operative care are ambulant. The following use case illustrates how aggravation of a patient's condition spontaneously changes the course of treatment by requiring participation of additional healthcare professionals.

If no health problems arise, the post-operative care will follow a common schema (Fig. 2): Every three months the patient must undergo a clinical examination at her gynecologist (Gyn^A). Semi-annually she is referred to a radiologist (Rad^A) for a mammography (RV$_M$). Initially, Gyn^A supplies a detailed anamnesis documentation to briefly summarize the preceding treatment. After each examination the radiologist creates a report about the diagnostic findings and makes it available to Gyn^A again.

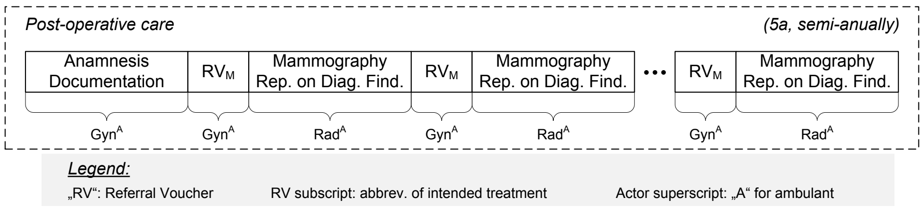

Post-operative care
(5a, semi-anually)
Anamnesis Documentation
RV_M
Mammography Rep. on Diag. Find.
Gyn^A
Rad^A
Legend:
„RV“: Referral Voucher
RV subscript: abbrev. of intended treatment
Actor superscript: „A“ for ambulant

Fig. 2. Breast cancer: post-operative care episode; no unclear symptoms

Because this is a periodic monitoring the doctors want to indicate normal and exceptional conditions. Anytime during the five years of post-operative care there is the possibility that the patient reports unclear symptoms or her gynecologist makes a suspicious finding that indicates metastases. Thus, the *condition indicator* is designated as a diagnostic report status attribute.

An Incidence Occurs: If, for example, the patient at some point complains about pain in her upper abdomen and/or a yellowish pigmentation of her skin, the gynecologist must find the reason for these symptoms as they may be caused by liver metastases. Fig. 3 illustrates the modified episode. The gynecologist sets the *condition indicator* of an exceptionally created anamnesis report to *guarded* and refers the patient to an internist (Int^A) for an upper abdomen sonography (RV$_{AS}$).

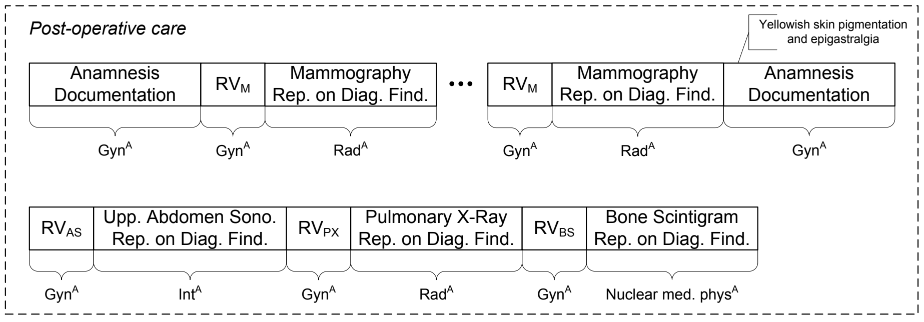

Post-operative care
Yellowish skin pigmentation and epigastralgia
Anamnesis Documentation
RV_M
Mammography Rep. on Diag. Find.
Gyn^A
Rad^A
RV_AS
Upp. Abdomen Sono. Rep. on Diag. Find.
RV_PX
Pulmonary X-Ray Rep. on Diag. Find.
RV_BS
Bone Scintigram Rep. on Diag. Find.
Int^A
Nuclear med. phys^A

Fig. 3. Breast cancer: post-operative care episode; classification of unclear symptoms

The internist might conclude in his report on diagnostic findings that the occurred symptoms are caused by a gallstone. In this case, the *condition indicator* of the sonography would also be set to *guarded* because the participants' consensus is that higher escalations are reserved for metastases. Of course, the patient is treated by the internist against gallstone but this forms another treatment episode.

For another patient, the initial suspicion could be strengthened by the upper abdomen sonography and liver metastases are now indicated. Consequently, the internist sets the *condition indicator* of his report to *serious*. The gynecologist will then instruct further examinations for potential lung or bone metastases: he refers the patient to a radiologist (Rad^A) for a pulmonary x-ray to check for lung metastases (RV$_{PX}$). A report on the x-ray results is written. The *condition indicator* would indicate the condition based on the x-ray, indicating exceptional lung condition with *normal* (“without pathological findings”) to *serious* (“lung metastases”). In parallel, a referral to a nuclear medical physician takes place, in order to conduct a bone scintigram in search for any signs of bone-related metastases (RV$_{BS}$).

For breast cancer, any suspicion of metastases (i.e. indicator value *guarded*) in one of the domains will always trigger the referral to both other domains (in the ternary set of liver, lung, and bones). Any affirmed suspicion (i.e. indicator value *serious*) will trigger a vital treatment. Treating the metastases will form an episode itself, besides the modified post-operative care. It will require a breast cancer center, an oncologist, and further surgical or chemo-therapeutic measures.

Benefits & Future Work: As far as described above, the user-defined attributes only record process-relevant states of the underlying reports. It would be possible to use the indicators as triggers for coordination actions.

Within the scenario, a modification of the *condition indicator* adornment into a *serious* state could trigger special notifications, e.g. notify epidemiological cancer registries which form a hierarchical national organization in Germany and complement the German cancer treatment centers. It would even be possible to offer users some means to define process templates for an escalation process plan. In case of a notable condition indication, the embedded rule engine, α-Props as mentioned above, could automatically extend the episode's process structure with the process steps from the escalation plan.

The rule engine could be extended in the future to dynamically support domain-specific rules that are not known at α-Flow design-time. Success would depend on providing an intuitive rule editor for end-users which is currently not implemented.

Further Adornment Example & Consensus Scopes: Another adornment could be *diagnosis certainty* with exemplary levels from *absolute* and *high* over *moderate* to *low*. In some situations it may not be feasible for physicians to make an authoritative diagnosis. Cooperative treatments of unclear symptoms or multimorbid patients require an intensified exchange of expert opinions. To indicate a limited certainty provides new participants with orientation while they gain an overview of the shared files.

Following the initial breast cancer classification episode (cf. [5]), the gynecologist creates a *diagnosis certainty* attribute for his initial report and sets the certainty of his own report to *low*. The radiologist later on provides a report on mammography and sets the certainty to *moderate* or *high*, according to the BI-RADS[1] indicator of the mammography. Finally in this specific episode, the pathologist contributes his diagnosis based on the biopsy with an authoritative certainty, so he sets the indicator to *absolute*.

Even if it seems possible to specify such adornments at design-time, there will always exist various conceptions of indicators both in name and value range. We propose that consensus finding can either be done ad-hoc during an episode or it can be provided by an institutional standard or a domain standard. An example for an indicator that is standardized for a domain is the BI-RADS score factor for mammographies as mentioned above. It would be perfectly conceivable, if users decide that they want the BI-RADS value directly available as a status attribute for mammography reports in breast

[1] Breast Imaging – Reporting and Data System

cancer episodes. The document-oriented process platform should allow for different consensus scopes and distinguish episode-, institution-, or domain-specific indicators.

5 Methods & State of the Art

One of the basic aspects for evolutionary systems is deferred design [8], i.e. to defer decisions from design-time to run-time. In order to achieve continuous adaptability [9], we need to be able to provide user-defined attributes at run-time. Thus, we need concepts to change behavior of program objects in regard to computing and persistence. Common methods are prototype-based programming and the EAV data design approach. In α-Adaptive we apply these concepts in order to find out how far they fit our purpose.

Prototype-based Programming: In class-based programming abstract classes are used to describe the common properties and behavior of concrete objects [10]. These objects are created by instantiation of the classes. In order to get an object with different properties or behavior a separate class has to be modeled. So the semantic decisions for the object are defined during the conceptual design of a system. This restrains the flexibility in the application core, because revising semantic decisions cannot be performed at run-time.

In prototype-based languages there are no classes but only objects. Abstract classes are substituted by *prototype objects*. A new object is created by copying an existing prototype object, which is also called *cloning* with the prototype as a *clone base*. This process supports the concept of inheritance in form of a dangling reference to the clone base: Every time a prototype is modified, all its derived clones are automatically updated. Both the prototype and its clones can be modified at run-time in schema and in value. Prototype changes are propagated to clones but if clones deviate from their parent their specific value remains. Thus, a mechanism is required to determine the difference in structure and in values between a prototype and one of its clones. By avoiding the use of abstract classes, the semantic decisions for the objects in such languages can be deferred from design-time to run-time.

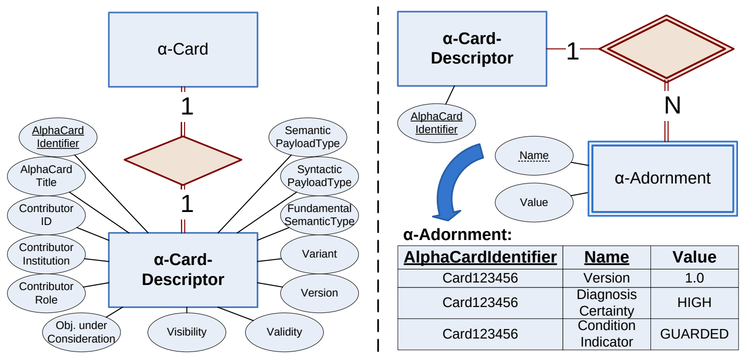
Entity-Attribute-Value Data Model: We must allow persisting data that was not known at design-time or deploy-time. Thus, the same flexibility that prototypes provide for the application core is also needed for persistence. Traditional database schema design freezes semantic decisions at design-time just like classes in programming do. It is not feasible to perform database schema alterations at run-time because schema-derived data access layers in dependent application systems would be disrupted. An update will also always affect all tuples, thus, historic tuples end up with many null values.

Entity-Attribute-Value (EAV) schema design [11] is a generalization of row modeling. EAV is based on association lists that originated in artificial intelligence. In contrast to the traditional schema design, the EAV design proposes a generic table with three columns: 1) the ID of an entity, 2) the name or identifier of an associated attribute, and 3) the corresponding attribute value for the entity. Thus, semantic decisions for an object are decoupled from altering the database schema because an arbitrary number of attribute-value pairs can be added at run-time.

6 The α-Adaptive Approach

The α-Adaptive approach focuses on the design of an evolutionary α-Adornment model to manage arbitrary α-Card status attributes. In a first step, we will demonstrate how we apply EAV in order to arrange an adaptive attribute schema. Subsequently, we extend traditional EAV for dDPM purposes and apply concepts of prototype-based programming to provide an attribute template that serves as a clone base for α-Card descriptors.

Creation of an adaptive α-Adornment schema: The first step towards an adaptive-evolutionary metadata model is to arrange an adaptive schema for the α-Adornments. The transformation of a static schema into an EAV schema is illustrated by the Entity/Relationship diagrams in Fig. 4. The statical design on the left does not support the extension of the α-Card descriptor with domain-specific adornments at run-time. The basic transformation into an EAV design results in an descriptor that contains no more fixed attributes, but a list of attribute-value pairs representing the α-Adornments.



AlphaCardIdentifier	Name	Value
Card123456	Version	1.0
Card123456	Diagnosis Certainty	HIGH
Card123456	Condition Indicator	GUARDED

Fig. 4. From static E/R design to adaptive EAV design

The first EAV extension concerns user-centric data types. In the original EAV, the physical data type of the attributes is a generic data type like String. There is no data type information included and data type transformations are commissioned to the application. Yet, adornments are user-centric and we require a slender type set from which a user might select a type for his or her adornment. Most data type sets in computer science are system-centric, e. g. primitive types in programming languages[2] or the ones in XML schema as a platform neutral superset. These data types are only comprehensible for programmers and are not adequate to fulfill an end-user's plain idea of data types. As a standard for user-centric types, we use the *Requirements Interchange Format*[3] (ReqIF) as a reference because requirements management is highly user-centric and ReqIF provides a slender type set. Thus, the data types implemented for α-Adaptive are: *String*, *Integer* (e.g. BI-RADS), *Timestamp* (e.g. due dates), *Enumeration* (e.g. our

[2] For example, in C++ a programmer in order to create an arbitrary integer variable must choose between types {short int, int, long int} crossed with {signed, unsigned} semantics.
[3] `http://www.omg.org/spec/ReqIF/1.0.1/11-04-02.pdf`

indicators) and *TextBlock* (e.g. Post-it notes). We extend the EAV schema by adding an additional attribute to store the user-centric data type restriction.

The second EAV extension concerns the consensus scopes, as we motivated them during the use case section. We again extend the EAV-entity schema with an attribute that specifies the consensus scope for each adornment. Currently, four scopes are implemented: users can choose between values *episode-specific*, *institution-specific* and *domain-specific* – the value *generic* is reserved and indicates α-Adornments that are used to grant the α-Flow platform functionality.

A third extension to the EAV schema is the instance attribute. α-Card descriptors with adaptive adornment sets solve only the first half of the α-Adaptive requirements. As discussed in the methods section, we need descriptors to provide prototype-oriented semantics, i.e. one α-Card descriptor becomes the template for others. Thus, the instance attribute is necessary as a flag and will be explained in the next section.

In conclusion, we can fulfill our data persistence requirements by adapting the traditional EAV approach. All our extensions to the basic EAV design are of general interest, in the context of attribute annotations of process artifacts in dDPM. The result is an attribute schema that is able to persist α-Adornments that can be adapted at run-time. The E/R diagram in Fig. 5 illustrates the resulting EAV:dDPM schema.

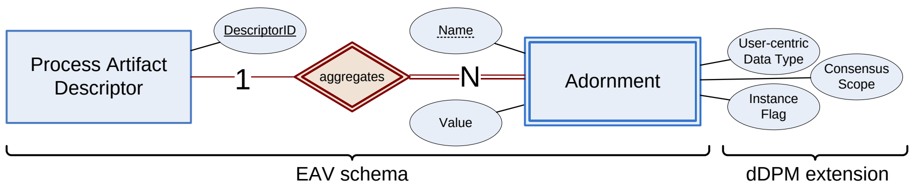


Fig. 5. The EAV:dDPM schema

Administration of the adaptive α-Adornment schema: Up to now, it would be possible to manage every single α-Card descriptor as a unique EAV-based object. Yet, the definition of adornments (at least within the episode if not within institutions or domains) is subject to a shared consensus of the episode's participants. Thus, we need a shared prototype within the α-Doc that serves as a template for all its descriptors. An individual descriptor will normally use only a subset of the prototyped adornments, e.g. BI-RADS will only be used for mammography reports and *diagnosis certainty* will not be used for reports on therapeutic measures.

To fulfill these requirements, the α-Adornment model is managed within an episode in form of the so called *Adornment Prototype Artifact* (APA). The APA enables a shared administration of the adornments and serves as a prototype for all α-Card descriptors that are generated by cloning the APA within one α-Doc. Each descriptor is allowed to use only a subset of the APA-defined adornments. To provide subset semantics, the α-Adaptive approach distinguishes between the adornment schema and the adornment instances of an α-Card descriptor. Figure 6 illustrates the correlation between the APA

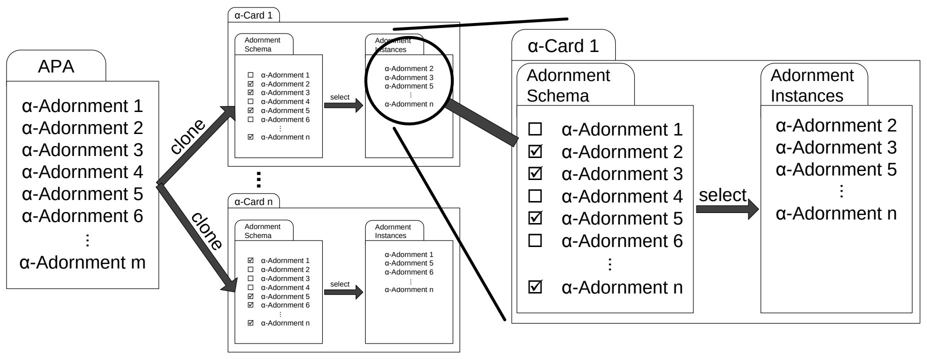


Fig. 6. Clone and Select: correlation between the APA prototype and the individual α-Cards' adornment-schema & -instances

as a prototype in contrast to the α-Adornment schema and α-Adornment instances of each derived α-Card descriptor.

The adornment schema of an α-Card descriptor contains all adornments that were inherited from the APA. The adornment instances, however, are the subset of adornments from the schema that the user actually selects to use for the individual α-Card. Thus, the *instance flag* was implemented for adornments as part of the EAV:dDPM schema. The setting of this *instance flag* means that the related adornment has been selected as an instance member of the corresponding α-Card descriptor.

Prototype Implementation: The functionality of α-Adaptive is provided by a prototype implementation in Java. The related classes are based upon the E/R diagram of Fig. 5. For cloning new α-Card descriptors a deep copy of the dynamic APA object structure is required. We implemented general-purpose deep cloning in Java by temporarily serializing the APA into a memory buffer and deserializing it. The values cloned from the APA provide default values for the descriptors.

Changes to the APA are propagated to the existing α-Card descriptors without overwriting individual adornment values in the descriptors as in prototype-based inheritance. For APA update propagation, every APA modification requires a delta check: The difference quantity between the set of α-Adornments in the APA and every α-Card descriptor within the α-Doc is determined and the descriptors are adapted to the APA's model without affecting the adornments that are part of the intersection between APA and α-Card descriptor. Changes like renaming adornments, switching consensus scope, changing default values, or changing Enumeration-based value ranges are transparently allowed and propagated, without disrupting existing descriptors. The prototype contains an embedded editor for visualization and editing of adornments: There are different screens for the APA, the Adornment schema and the Adornment instances of an α-Card descriptor.

7 Related Work

Content-oriented workflows (e.g. "object-aware" [12], "artifact-centric" [13], or "data-driven" [14] approaches) provide process execution based on data dependencies. The main characteristic in content-orientation is to separate the data structure from the process structure, and to support formal bindings between data state and process enactment, thus it contrasts to activity-orientation with its focus on control flow. Case handling is orthogonal to both. We consider α-Flow as a content-oriented workflow approach for case handling in distributed inter-institutional environments.

Adaptiveness in Activity-oriented Approaches: A modern approach to activity-oriented workflows is Proclets[4] by van der Aalst et al. [15, 16]. Proclets are interacting processes that exchange messages, named *performatives*, via *channels*. The Proclets approach proposes a shift in focus from control flow to communication in order to reduce control flow complexity. The approach is similar to conversation and choreography diagrams in the BPMN[5] 2.0 standard. Neither Proclets nor BPMN support adaptive change of their data flow objects or message structures.

In contrast, workflow adaption is discussed for ADEPT$_{flex}$ [17] by Reichert and Dadam. ADEPT$_{flex}$ is based on block-structured process description. Change operations in ADEPT$_{flex}$ consider only the control flow. Data flow, as an addendum to the control flow, is addressed for checking correctness of control flow change operations, which is possible because the exchange of data between tasks is based on global variables. Data elements are derived from input/output parameters of tasks. Users can extend the data structure not directly but by inserting new tasks with according parameters or by replacing tasks. This raises a variety of challenging issues with respect to dynamic parameter mapping and leaves significant complexity to the user.

Adaptiveness in Content-oriented Approaches: Content-oriented approaches commonly rely on fixed content schemas and status triggers to drive workflow automation. They do not consider run-time adoption of content schema, life-cycle configuration, or artifact status attributes. A state-of-the-art approach to content-oriented workflows is PHILharmonicFlows [18] by Künzle and Reichert. In PHILharmonicFlows data is managed based on object types. At run-time, the number of object instances and links may vary but the types and their structure is statically defined at workflow design-time. An adaptive artifact attribute model, as we propose in α-Adaptive, allows demand-driven data extensions to evolve the process status description at run-time by the human actors.

Acknowledgements

The first author wants to thank Dr. med. Helmut Neumann who as a gynecologist explained breast cancer treatment to me and Rita M. Neumann who survived breast cancer and familiarized me with the patient perspective.

[4] From an implementation perspective, Proclets had been based on Petri nets and later on YAWL.

[5] Business Process Model and Notation. Int'l standard by the Object Management Group.

References

1. W. M. P. van der Aalst, M. Weske, and D. Grünbauer. Case handling: a new paradigm for business process support. *Data & Knowledge Engineering*, 53(2):129–162, 2005.
2. C. P. Neumann and R. Lenz. alpha-Flow: A Document-based Approach to Inter-Institutional Process Support in Healthcare. In *Proc of the 3rd Int'l Workshop on Process-oriented Information Systems in Healthcare (ProHealth'09)*, Ulm, Germany, September 2009.
3. C. P. Neumann and R. Lenz. The alpha-Flow Approach to Inter-Institutional Process Support in Healthcare. *Int'l Journal of Knowledge-Based Organizations*, 2(3), 2012. Accepted for publication.
4. C. P. Neumann, F. Rampp, M. Daum, and R. Lenz. A Mediated Publish-Subscribe System for Inter-Institutional Process Support in Healthcare. In *Proc of the 3rd ACM Int'l Conf on Distributed Event-Based Systems (DEBS 2009)*, Nashville, TN, USA, July 2009.
5. C. P. Neumann and R. Lenz. The alpha-Flow Use-Case of Breast Cancer Treatment – Modeling Inter-Institutional Healthcare Workflows by Active Documents. In *Proc of the 8th Int'l Workshop on Agent-based Computing for Enterprise Collaboration (ACEC)*, Larissa, Greece, June 2010.
6. A. LaMarca, W. K. Edwards, P. Dourish, J. Lamping, I. Smith, and J. Thornton. Taking the work out of workflow: mechanisms for document-centered collaboration. In *6th European Conf on Computer Supported Cooperative Work*, pages 1–20. Kluwer Academic Publishers Norwell, USA, 1999.
7. A. Todorova and C. P. Neumann. alpha-Props: A Rule-Based Approach to 'Active Properties' for Document-Oriented Process Support in Inter-Institutional Environments. In Ludger Porada, editor, *Lecture Notes in Informatics (LNI) Seminars 10*. Gesellschaft für Informatik e.V. (GI), March 2011.
8. N. V. Patel. *Adaptive Evolutionary Information Systems*. Idea Group Inc, 2002.
9. R. Lenz. Information Systems in Healthcare – State and Steps towards Sustainability. *IMIA Yearbook 2009 – Yearbook of Medical Informatics as a supplement of Methods of Information in Medicine*, pages 63–70, 2009.
10. A. H. Borning. Classes versus prototypes in object-oriented languages. In *Proc of 1986 ACM Fall joint computer conference*, pages 36–40. IEEE Computer Society Press, 1986.
11. P. M. Nadkarni. Data extraction and ad hoc query of an entity-attribute-value database. *Journal of the American Medical Informatics Association*, 5(6):511, 1998.
12. V. Künzle and M. Reichert. Towards Object-Aware Process Management Systems: Issues, Challenges, Benefits. *Lecture Notes in Business Information Processing*, 29:197–210, 2009.
13. David Cohn and Richard Hull. Business Artifacts: A Data-centric Approach to Modeling Business Operations and Processes. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, September 2009.
14. D. Müller, M. Reichert, and J. Herbst. Data-driven modeling and coordination of large process structures. *Lecture Notes in Computer Science*, 4803:131, 2007.
15. W. M. P. Van Der Aalst, P. Barthelmess, C. A. Eliis, and J. Wainer. Proclets: A framework for lightweight interacting workflow processes. *Int'l Journal of Cooperative Information Systems*, 10(4):443–482, 2001.
16. RS Mans, NC Russell, WMP van der Aalst, PJM Bakker, AJ Moleman, and MWM Jaspers. Proclets in healthcare. *Journal of Biomedical Informatics*, 43(4):632–649, 2010.
17. M. Reichert and P. Dadam. ADEPTflex – supporting dynamic changes of workflows without losing control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.
18. V. Künzle and M. Reichert. PHILharmonicFlows: towards a framework for object-aware process management. *Journal of Software Maintenance and Evolution: Research and Practice*, 23(4):205–244, 2011.