The core components of the modeling framework has been designed and implemented to facilitate an easy integration with the inference and learning modules developed as part of Work packages 3 and 4. Figure 1 shows a high-level overview of the key component in the part of the AMIDST software tool that is directly related to learning; these learning-related components are connected to the framework core (see above) through the *Inference* component and .... (highlighted using boxes without rounded corners). The color coding in the figure summarizes the implementation status: blue boxes represent software components that have been implemented in the AMIDST toolbox and green boxes represent components that are part of the software design specification, but which has not yet been implemented.

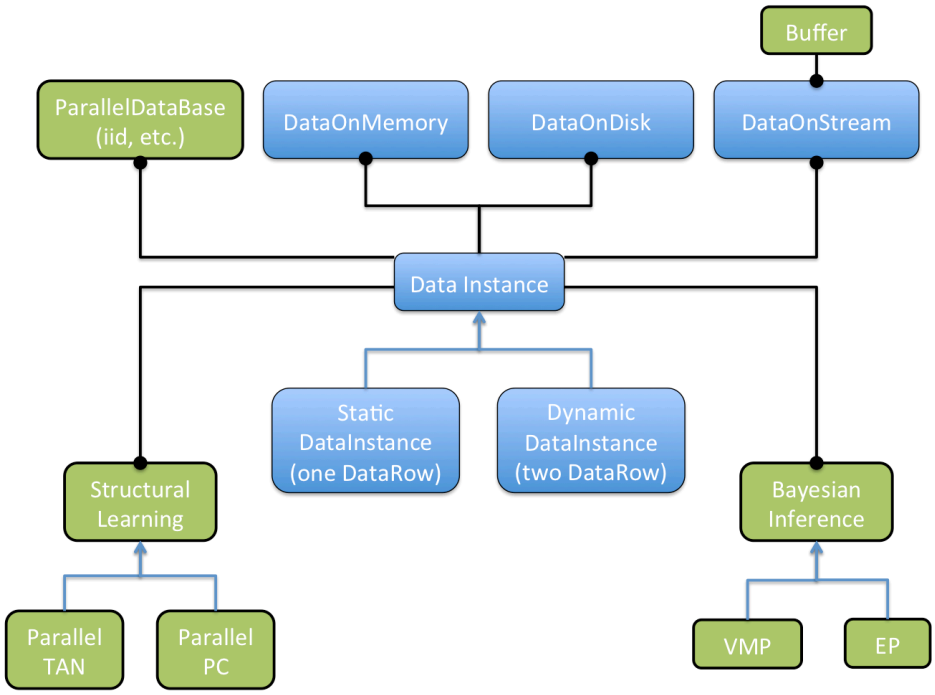

Figure 1: Illustration of the design of the software components related to model learning. Nomenclature: The boxes in the figure represent software components (sets, possibly singletons, of classes), a rounded-arc going from X to Y indicate that Y 'uses/references' X, and a regular arc from X to Y implies inheritance.

As described in Section **??** we pursue a fully Bayesian approach for doing parameter learning in the AMIDST framework. This, in turn, means

that parameter learning reduces to the task of inference for which we plan to consider two approaches: variational message passing (positioned in a variational Bayes framework) and expectation propagation. Particular efficient implementations of these methods can be realized when the distribution families of the models are conjugate-exponential. In this case the inference operations can be further supported by specifying the exponential distributions using their natural parameters, cf. Section **??**. These improvement are realized through tailored exponential family implementations of the standard distributions that are part of the AMIDST framework (such as conditional linear Gaussian distributions).

Central to the part of the design pertaining to learning is the data management components, which, in the current design, is connected to (or used by) the learning components *Structural Learning* and *Bayesian Inference* through the *Data Instance* component. This component consists of a single class representing a particular evidence configuration, such as the observed values of a collection of variables at time t or a particular row in a database. The *Data Instance* component is furthermore used by the components implementing the streaming (*DataOnStream*) and database functionality (*DataOnMemory*, *DataOnDisk*, and *ParallelDatabase*) of the framework:

- *DataOnMemory* implements database functionality for data sets that can be loaded into main memory.
- *DataOnDisk* provides functionality for handling datasets too larger to be loaded in main memory.
- *ParallelDatabase* implements a distributed database.
- *DataOnStream* implements functionality for handling data streams, where data is being *pushed* to the AMIDST system (in contrast to a pull-approach that is standard when dealing with static database). To allow for variation in the run-time performance of the system a buffer component (*Buffer*) is needed for storing the most recent unprocessed cases.

The data architecture design not only allows for ...

FiXme: *What about variable selection - component support for this task?*

Implementations of structural learning algorithms will be realized through the *Structural learning* component and its specialized sub-classes. The design includes components for supporting PC and TAN learning in a parallel

setting (cf. Task 4.1), but, currently, the implementation only supports standard PC learning and parallel TAN learning by interfacing to the Hugin API and invoking the corresponding functions.

FiXme: *Should this be reflected in the design?*

The Bayesian parameter learning approach will be realized through *Bayesian inference*. Specifically, we plan to consider both variational message passing (implemented in the *VMP* component) and expectation propagation (implemented in the *EP*) component; both of which connects to the core components of the framework through the *Bayesian inference* component (see Section **??**). These two parameter learning methods will support the activities in Task 4.2 and Task 4.4.

The high-level design description above provides an overview of the current design of the AMIDST software framework, including core components as well as components directly related to model learning. The current status of the software implementation in relation to the design can be summarized as follows:

- ...