- WP2 - Common Software Tools

Particle detectors continuously evolve: future detectors will be bigger with higher granularity, will have higher particle rates, and include new detector types. Also technological advances - especially multicore CPUs - make new demands on the software. These developments require new software features and better software performance.

Many of the software requirements are common to all the future experiments. The goal is then to develop generic code, independant of which experiment it is to be used for, with high re-usability, high reliability, and using efficient algorithms. Making generic code allows a larger user-base which feeds back into higher quality code, while at the same time reducing the overall effort needed in development and maintenance.

Design of future detectors requires detailed simulation to find the expected performance of different options. GEANT4 is the universal modern tool for detailed simulation. It is mature software and no direct development of simulation tools is requested here. However, developments are needed in two areas of simulation. One is to facilitate putting in realistic misalignments of detectors, which the geometry package should handle. The other is to deal with high event rates - pile-up of many interactions simultaneously - giving high hit densities. In simulation this needs efficient use of memory and in reconstruction it needs efficient algorithms for track finding.

The raw event data stored by experiments has to be reconstructed to find the particles that were created in the event. This can be sub-divided into tracking for charged particles, and calorimetry for both charged and neutral particles. Much of the reconstruction software can be written in a detector-independent way, making it useful to a broad user base.

Both simulation and reconstruction need to know the shape, position, and materials of the detector: the detector geometry. Having a single geometry model that can feed all parts of the software is a major goal: it guarantees consistency and saves effort in maintenance and modifications.

Large computing resources are needed for the complex detectors. Computing hardware develops with time, taking advantage of technological advances. Software needs to evolve to take full advantage of these improvements. The current trend for computers is to go to multicore CPU's, with ever higher numbers of cores. Software needs to evolve to take full advantage of this, with multi-threading and memory management. This will benefit both simulation and reconstruction.

These then are the central tasks of the software networking work package: generic software for the geometry model and reconstruction; and optimising all software to take full advantage of multicore CPU's.

Work package number	WP2		**Start date or starting event:**				M1	
Work Package title	Common Software Tools							
Activity type	COORD							
Participant number	1	8						
Participant short name	CERN	CNRS	DESY	INFN	IFIC	USC	UNIBRIS	UBRUN
Person-months per participant	92	54	54	66	23	12	12	12
Participant number								
Participant short name	UCAM	UEDIN	UNIGLA	ULANC	UOXF	QMUL		
Person-months	24	10	10	8	12	4		

per participant								

Objectives:
Task1: General purpose detector geometry description package

- Develop a detector-independent geometry package, using best practice from the current packages available, and with sufficient flexibility to cope with the needs of the future detectors, including misalignments of detector elements
- Optimize it for representation in memory to allow fast access to detector parameters, including alignment and calibration constants
- Develop database tools to store and retrieve alignment and calibration constants
- Create interfaces to produce geometries for simulation (Geant4, Fluka), digitization, reconstruction, alignment, and event display programs.

Task 2: Reconstruction software

- Develop a toolkit based on best available practice for charged-particle tracking and calorimetry.
- Develop software to handle high particle-multiplicities: Optimise algorithms, memory management, data-base access, and file handling.
- Develop general-purpose alignment software.
- Improve existing persistency software for long term storage of information; adapt these methods to the testbeam DAQ systems in WP4, WP10, and WP11.
- Develop an event display using the geometry package, with flexibility to cover in particular the various testbeam detector setups.

Task 3 Parallelisation of software frameworks to exploit multi-core processors

- Explore thread synchronisation and memory management techniques. Select the most promising and develop them.
- Optimise the major particle physics frameworks to run on multicore processors, using the techniques developed above.

Description of work:
Task 1. General purpose detector geometry description package
Usually experiments want the geometry described in one place. This is efficient and eases maintenance during the life of an experiment: changes are only required in one place, guaranteeing consistency everywhere else. This central geometry then needs to be interfaced to all the other software packages, which often have very different requirements on level of detail needed etc.: the optimum level of detail needs to be accessible in each package. Covering several future experiments requires a comprehensive and flexible generic model. Methods for dealing with real detector layouts with imperfections - mainly mis-alignments and calibrations - which are time dependent also need to be developed, including databases with fast access to large data sets and efficient memory use to cache the information.

Work plan: Review geometry systems used by current experiments and select the best elements for a widely applicable package. Pay particular attention to efficient memory storage to allow for the future large detectors. Allow for mis-alignments as occur in real detectors. Develop efficient data base tools for the storage of mis-alignment and calibration constants. Enable geometries to be mis-aligned for simulation, including tools to detect or prevent clashes between detector volumes. Develop interfaces between the geometry package and client software. This will include the ability to help simplify the geometry to provide the optimum level of detail needed by a client (CERN, CNRS, DESY, UEDIN, UNIGLA, QMUL).

Task 2 Reconstruction software
The raw data - hits - from the detectors have to be reconstructed into information on the tracks of the particles that were created. The large variations between detectors and the complexity of the data mean it would be too ambitious to try to develop a unified general-purpose reconstruction software package. However, many tasks within such a package are common to all detectors, such as parts of charged particle tracking and calorimetry and particle flow, combining the two. A detector-independant

toolkit should be developed using the geometry package of task 1, to ease the process of writing high-quality reliable software for these tasks.

As beam intensities increase, more interactions occur alongside the interactions of interest, known as pile-up events. These increase the amount of data to be handled. More efficient ways to handle the high particle multiplicity need to be developed, especially in pattern recognition for tracking. Memory management is also important here for efficiency. The toolkits should handle high-multiplicity events efficiently.

Reconstruction needs accurate information on detector positions. This information is usually best obtained from the data itself, in a process known as alignment. High quality generic alignment software could considerably reduce the effort typically needed for detector alignment.

Longer-term storage of information (“persistency”) - from the raw data to the results of high-level analysis - needs to be developed. This is urgent for handling test-beam data, and in the longer term will be vital for future experiments. Different storage times and access frequencies require different solutions for storage and re-use. The current experiments have solutions, which can be improved on and generalised.

Visualisation of both the detector set-up and events is important to understand what happened in an event, and also for debugging and developing the software. Flexible tools are needed in testbeams where the geometry changes frequently. Development of an event display for testbeams will both help in debugging and developing the geometry and reconstruction packages, and in analysis of test-beam data.

Work plan: Develop toolkits for pattern recognition and fitting of charged tracks, suitable for broad use in testbeams and future experiments. Bring experts together from several experiments to find the best solutions. Optimise reconstruction and simulation for the very high particle rates expected at the SLHC. Produce a generic alignment package initially for testbeam detectors without magnetic fields, and extending it later to cover current and future experiments with magnetic fields. Improve and extend existing persistency software for storage of raw data and apply it to the integrated testbeams. Produce an event display based on the geometry package. Use it to help develop the geometry and reconstruction software, as well as for analysis of testbeam data. (CNRS, DESY, INFN, IFIC, USC, UBRUN, UCAM, UNIGLA, ULANC, UOXF)

Task 3 Parallelisation of software frameworks to exploit multi-core processors
With current CPU's containing four cores and next-generation ones expecting up to 64, many particle physics applications - in simulation, reconstruction, and online-triggering - need considerable adaptation to optimise use of the extra processing power. Event parallelism can be exploited by launching as many processes (threads) as there are cores, provided thread synchronisation techniques are available. Simulation, reconstruction and triggering are all memory-intensive but it is inefficient and costly to increase the memory in line with the number of cores. Much of the memory - geometry and calibration constants, and magnetic field for example - is common to all threads, which can be exploited by developing techniques for sharing memory between many cores.

Once the best techniques for thread synchronisation and memory management have been identified, implementations for the software frameworks currently used in particle physics will be developed. These include Geant4, Root, ILC reconstruction, and the ATLAS, CMS and LHCb frameworks Athena, CMSSW, and Gaudi. Adapting the framework to exploit multicore architectures will automatically benefit the applications built on them, with no - or only minimal - changes needed to specific algorithmic code.

Work plan: Evaluate current and future multicore architectures, selecting and developing tools to measure performance and identify bottle-necks. Try prototype solutions to remove the bottle-necks. Apply solutions initially to the LHC data analysis frameworks, and later to other major particle physics software. By developing the overall packages and frameworks to make optimum use of the multicore

CPU's, the vast amount of algorithmic code already developed by the broad community will automatically benefit. (CERN, UNIVBRIS)

Deliverables of Tasks	**Description/title**	**Nature**[1]	**Delivery Month**[2]
2.1	Initial geometry package ready suitable for simulation and reconstruction.	O	M24
2.1	Geometry package with efficient memory management and allowing for mis-alignments.	O	M36
2.1	Final geometry package with interfaces to relevant software applications.	O	M48
2.2	Event display available for testbeams	O	M24
2.2	Alignment package without magnetic field suitable for use in testbeam data analysis	O	M24
2.2	Initial release of software for tracking, calorimetry and particle flow analysis with persistency software suitable for use in testbeam data analysis	O	M30
2.2	Tracking and calorimetry optimised for high pile-up	O	M36
2.2	Final release of alignment package suitable for experiments with magnetic fields	O	M48
2.2	Final release of persistency, tracking, calorimetry and particle flow analysis tools suitable for experiments	O	M48
2.3	LHC software libraries adapted to multi-core CPU's	O	M36
2.3	Software libraries for remaining applications adapted to multi-core CPU's	O	M48

Milestones	**Task**	**Description/title**	**Expected date**[2]	**Comment**
2.1	2.1	Geometry package Software Design Document based on current models and requirements of the detectors	M10	Report forms basis for decisions on which solutions to follow
2.2	2.1	Running prototype of geometry model with limited functionality to demonstrate applicability	M18	Quantitative evaluation of processing speed and memory use
2.3	2.2	Reconstruction Software Design Document based on review of current software and future needs	M10	Report forms basis for decisions on which solutions to follow
2.4	2.2	Tracking, calorimetry and particle flow analysis prototype software	M22	Quantitative evaluation of speed and memory needs to check solutions
2.5	2.3	Report surveying multicore architectures and tools to measure performance	M10	Report forms basis for decisions on which solutions to follow